UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIENCIA DA COMPUTAÇÃO

HENRIQUE DE NONI

ANÁLISE DE DESEMPENHO UTILIZANDO BENCHMARKS ENTRE LINUX RASPBIAN E WINDOWS 10 IOT COM USO DO RASPBERRY PI

CRICIUMA

2016

HENRIQUE DE NONI

ANÁLISE DE DESEMPENHO UTILIZANDO BENCHMARKS ENTRE LINUX RASPBIAN E WINDOWS 10 IOT COM USO DO RASPBERRY PI

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Sérgio Coral

CRICIUMA

2016

HENRIQUE DE NONI

ANÁLISE DE DESEMPENHO UTILIZANDO BENCHMARKS ENTRE LINUX RASPBIAN E WINDOWS 10 COM USO DO RASPBERRY PI

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Sistema Operacional.

Criciúma, 29 de Novembro de 2016.

BANCA EXAMINADORA

Prof. Esp. Sérgio Coral - Orientador

Prof. MSc. Paulo João Martins - UNESC

Prof. Esp. Valter Blauth Junior - UNESC

Dedico este trabalho aos meus pais que sempre me apoiaram e a todos os amigos que conheci na faculdade que de alguma forma sempre me ajudaram.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem ele não estaria aqui, por sempre me dar forças nas horas difíceis e por nunca desistir dos meus objetivos.

Agradecer aos meus pais por me darem o dom da vida, por sempre terem me dado educação e terem me ensinado a ser uma pessoa boa.

Agradeço ao meu orientador Sérgio Coral por me aceitar ser seu orientando e pela paciência em me ensinar e orientar.

Agradecimento especial ao professor Kristian Madeira por me ajudar na escolha do método estatístico e a escolha do programa SPSS usado para a validação dos dados.

Agradecer aos amigos que conheci na faculdade Afonso Vieira, Rafael Schimitz de Oliveira, Guilherme Cruz da Cunha, Alisson Zecchinel, Larissa Gomes da Rosa, Ronaldo Borges Vicente, Sulivan Borges Brasil, Laudecir Martins Hasckel, Ederson Macedo de Oliveira, Jose Silvestre Correia, Álvaro Ambriz Domingos, Gustavo Pasquali Moretti, Willian Jeferson Meurer, José Vitor Morona Souza e também outros que não coloquei o nome, mas estão em minha memória, obrigado por sempre me animarem e nunca me fazerem desistir dos meus objetivos.

“Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar aonde a maioria não chega, faça o que a maioria não faz. ”

Bill Gates

RESUMO

Neste trabalho realizou-se um teste de desempenho entre o sistema operacional Linux Raspbian e o Windows 10 IoT para avaliar qual sistema operacional se adapta melhor ao uso da plataforma Raspberry Pi. Para realizar os testes foram utilizados alguns *benchmarks online* e alguns *benchmarks desktop*. Para processar e analisar os dados dos testes e fazer os comparativos foi utilizado tabelas e gráficos do Excel e o software SPSS da IBM. Com os testes realizados o Windows 10 IoT conseguiu se adaptar melhor ao Raspberry devido as médias melhores em relação ao Linux Raspbian. Apesar de o Windows 10 IoT ter se saído melhor, em ambos os sistemas operacionais apresentaram um certa instabilidade ao usa-lo.

Palavras-chave: Raspberry Pi. Linux Raspbian. Windows 10 IoT. *Benchmarking*.

ABSTRACT

In this work a performance test was performed between the Linux operating system Raspbian and Windows 10 IoT to evaluate which operating system is best adapted to the use of the Raspberry Pi platform. To perform the tests we used some online benchmarks and some desktop benchmarks. In order to process and analyze the data of the tests and to make the comparatives was used tables and graphs of Excel and the software SPSS of IBM. With the tests carried out, Windows 10 IoT was able to adapt better to Raspberry due to the better means compared to Linux Raspbian. Although Windows 10 IoT has fared better, both operating systems show some instability when using it.

Keywords: Raspberry Pi. Linux Raspbian. Windows 10 IoT. Benchmarking.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AT&T	American Telephone and Telegraf
ARM	Advanced RISC Machines
ARPANET	Advanced Research Projects Agency Network
BESYS	Bell Operating System
CSI	Camera Serial Interface
DARPA	Defense Advanced Research Projects Agency
DHCP	Dynamic Host Configuration Protocol
DSI	Display Serial Interface
DVI	Digital Visual Interface
FSF	Free Software Fundation
FTP	File Transfer Protocol
GPL	General Public Licence
HD	High Definition
HDMI	High Definition Multimedia Interface
IDE	Integrated Drive Eletronics
IoT	Internet of Things
ISO	International Organization for Standardization
MIT	Massachusetts Institute of Tecnology
MS-DOS	Microsoft Disk Operating System
Multics	Multiplexed Information and Computing Service
RCA	Radio Corporation of America
RISC	Reduced Instruction Set Computer
SD	Secure Digital
SDHC	Secure Digital High Capacity
SDXC	Secure Digital Xtended Capacity
SDK	Software Development Kit
SPSS	Statistical Package for Social Sciences
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNESC	Universidade do Extremo Sul Catarinense
Unics	Uniplexed Information and Computing Service
USB	Universal Serial Bus

VGA Video Graphics Array

SUMÁRIO

1 INTRODUÇÃO

Os setores de TI e da eletrônica, são dois setores que estão aumentando muito espaço nestes últimos anos, essas duas áreas se completam e se estendem em muitos tipos de atividades, na área de desenvolvimento, por exemplo, muitas vezes é preciso soluções rápidas e de baixo custo, o ideal seria plataformas eletrônicas de prototipação pequenas e com pouca programação embutidas, uma dessas plataformas de prototipação é o Raspberry Pi (SABER ELETRÔNICA, 2013).

O *kernel* Linux é um sistema operacional desenvolvido por milhares de programadores e colaboradores de todo o mundo, com pacotes estáveis para todos os tipos de usuários e até para empresas, ele é o sistema operacional que está sendo cada vez mais encontrado em desktops e que possui o núcleo mais usado em servidores de grande porte no mundo por ser muito estável e livre de ameaças como vírus (ALENCAR, 2005).

O Windows é um sistema operacional que opera através de interfaces gráficas em formas de janelas, daí o nome Windows, fundado por Bill Gates e Paul Allen, sua última versão o Windows 10 é multiplataforma, ou seja, funciona em computadores, celulares e tablets, o diferencial é que ele funciona em plataformas menores como o Raspberry Pi.

Para avaliar o desempenho de qualquer sistema é necessário um software de *benchmarking*, ou seja, são *softwares* específicos para avaliar o desempenho de determinados sistemas, para realizar a metodologia de benchmarking é necessário definir as métricas de desempenho para então poder ser avaliado.

Atualmente existem poucos trabalhos na área de desempenho de sistemas operacionais, pois é uma área com certo grau de dificuldade para medir desempenho, pois dependendo do tipo de *hardware* utilizado, varia muito os resultados, no caso deste projeto, como o *hardware* do Raspberry é bem específico, ou seja, possui apenas um tipo de configuração, o resultado é bem verídico e com grande grau de confiabilidade nos seus resultados.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é utilizar métricas de desempenho com Benchmarks no Raspberry Pi utilizando Linux Raspbian e Windows 10.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos cogitados para a elaboração deste projeto são:

a) compreender o funcionamento de um Raspberry Pi;
b) entender o funcionamento do Linux Raspbian no Raspberry Pi.
c) entender o uso do Windows 10 em um Raspberry PI;
d) compreender como utilizar métricas de desempenho através de *Benchmarks* para sistemas operacionais;
e) aplicar o uso do Windows 10 em Raspberry Pi's;
f) aplicar o uso do Linux Raspbian em Raspberry Pi's;
g) utilizar as métricas de desempenho no Linux Raspbian e no Windows 10 IoT que serão: Desempenho de processamento (Renderização, Cálculos Matemáticos e Algoritmos Complexos), desempenho de vídeo (Análise de gráficos, imagens e jogos), desempenho de navegadores Web (avaliação de código fonte, *frameworks*).

1.3 JUSTIFICATIVA

Com a utilização de métricas de desempenho é possível conseguir melhorias e resultados satisfatórios no uso dos sistemas operacionais e do Raspberry, pois ajuda a corrigir falhas e erros e também auxilia onde deve ser melhorado para próximos projetos de sistemas operacionais onde o foco principal é a satisfação do cliente (GUARIZZO, 2008).

O Raspberry Pi além de ser pequeno, ter baixo consumo energético e ser de baixo custo pode ser usado de várias maneiras diferentes, ele pode ser usado como um computador normal, algumas aplicações são até melhores de usar no Raspberry do que em desktops, pode ser usado para assistir filmes, navegar na web

e até para jogos. Ele é uma plataforma para divertimento, para serviços e para criar coisas novas, nele é possível à instalação de vários softwares livres, também é possível programar nele em várias linguagens como C, *Python, Ruby* e *Java*. Ele se diferencia de um computador normal não apenas pelo preço ou pelo seu tamanho, ele também consegue integrar a eletrônica com projetos de arduino na construção de micro controladores (RICHARDSON; WALLACE, 2012).

O Windows 10 tem o mesmo layout em desktops, celulares e tablets para não dificultar muito a interface com o usuário, e têm muitas funcionalidades novas em relação a sua antiga versão o Windows 8.1. Primeiramente a volta de o menu iniciar dividida em duas partes, a primeira metade do menu com a mesma interface do menu do Windows 7 e a outra metade com interface em blocos com a mesma interface do Windows 8, o Windows explorer ficou também mais dinâmico mostrando uma lista dos arquivos usados recentemente, vem também com um novo navegador substituindo o Internet Explorer, o Microsoft Edge, mas o que mais chama atenção é o desempenho, a versão de 32 bits funciona com apenas 1GB RAM e a versão de 64 bits funciona com apenas 2GB RAM tornando o sistema muito rápido e eficaz (BRITO, 2014).

O Linux é o software *Open Source* mais importante que foi produzido, ele é um sistema operacional completo em todos os requisitos tanto para uso pessoal, como para negócios e está crescendo muito seu uso dentro da eletrônica digital. As empresas instalam o Linux para gerenciar controles financeiros, ambientes hospitalares, ambientes de computação paralela e distribuída e empresas de telecomunicações. Ele é diferente dos outros sistemas operacionais, primeiro porque ele é totalmente gratuito e ainda ele é desenvolvido por vários voluntários de todo o mundo, onde eles compartilham os códigos na Internet, corrigem erros e melhoram cada vez mais o sistema operacional que cresce o número de usuários todos os anos (SOTO; SILVA; MENEZES, 2002).

A vantagem de usar *benchmark* é primeiramente atender a todas as necessidades do cliente, também é importante cumprir a todas as metas e objetivos propostos e também com os seus processos tentam descobrir um método melhor e eficaz para resolver determinado problema e não reinventar de novo a partir do zero, seu principal objetivo é a fixação de metas (WAQUED, 2002).

Este trabalho fez uma análise para ver qual o sistema operacional que se

adapta melhor para o uso em Raspberrys, será realizada uma análise de várias métricas de desempenho em cima dos dois sistemas operacionais, como por exemplo: Desempenho de processamento (Renderização, Cálculos Matemáticos e Algoritmos Complexos), desempenho de vídeo (Análise de gráficos, imagens e jogos), desempenho de navegadores Web (avaliação de código fonte, frameworks). Cada um desses itens será comparado através de vários softwares de desempenho baixados da internet.

Depois de feito todas estas análises, elas serão comparadas e irão verificar qual o sistema operacional que se adapta melhor ao Raspberry Pi.

1.4 ESTRUTURA DO TRABALHO

Este projeto é composto por nove capítulos, o primeiro capítulo mostra uma pequena apresentação do tema, seguindo do objetivo geral e dos objetivos específico do projeto, depois a justificativa e a estrutura do projeto.

No capítulo dois é falado da plataforma de prototipação Raspberry Pi, onde mostra toda a sua história, o funcionamento de seus componentes e sua arquitetura separadamente e onde ela é aplicada em seus mais diversos projetos.

No capítulo três é começado a falar dos dois sistemas operacionais que vão operar no Raspberry Pi, o primeiro é o Linux, onde será discutida toda a sua história, sua forma de licenciamento e uma breve introdução das versões mais importantes do Linux, inclusive as que serão utilizadas no projeto que é a versão destinada ao Raspberry Pi, chamado Linux Raspbian, baseado no Linux Debian.

No capítulo quatro é discutido sobre o outro sistema operacional que será utilizado no projeto, o Windows da empresa Microsoft, neste capítulo será mostrada toda a sua história e serão faladas de suas versões, suas formas de licenciamento e será falado da sua última versão lançada no ano de 2015, que também conta com uma versão para o Raspberry Pi, o Windows 10.

No capítulo cinco é retratada sobre as métricas de desempenho, sua definição, como elas funcionam, seus tipos e onde elas podem ser aplicadas nas mais diversas atividades, será falado sobre *benchmarking*, também será falado quais métricas de desempenho serão utilizadas para medir o desempenho nos dois sistemas operacionais dentro do Raspberry Pi. E também serão discutidos os

métodos estatísticos para o processamento e análise dos resultados.

Então no capítulo seis foi demonstrado alguns trabalhos correlatos para análise de desempenho em diversos tipos de sistemas utilizando benchmarks, no capítulo sete será demonstrado o que será feito neste projeto, contendo toda a metodologia e os recursos utilizados, no capítulo oito será demonstrado como feito o processamento e a análise destes dados incluindo os seus resultados e no capítulo nove foi mostrada a conclusão do projeto e os seus trabalhos futuros.

2 RASPBERRY PI

O Raspberry Pi ficou popularmente conhecido no mundo no ano de 2011, ele é uma placa de prototipação de mais ou menos do tamanho de um cartão de crédito que custa cerca de 35 dólares baseada na arquitetura de processadores *Advanced RISC Machines* (ARM), esta plataforma foi bem aceita por desenvolvedores e por profissionais da área de eletrônica (SABER ELETRÔNICA, 2013).

2.1 PLATAFORMAS DE PROTOTIPAÇÃO

A palavra prototipagem designa um conjunto de técnicas para se produzir objetos físicos a partir de uma fonte de dados gerenciada por um computador, a prototipagem oferece vantagens em várias aplicações em comparação aos métodos de produção tradicional como fresamento e torneamento (GORNI, 2001).

Existem várias plataformas de prototipagem, uma das mais conhecidas é o arduíno que é uma família de micro controladores desenvolvidos na Itália que é totalmente *Open Source*, ou seja, seu hardware é completamente aberto para os usuários desenvolverem suas aplicações, diferente do Raspberry Pi que possui peças proprietárias como é o caso de seu processador *Broadcom* (WARNER, 2014).

O diferencial do arduíno são suas entradas analógicas e *shields*, que podem ser usados para fazer vários tipos de sensores, como por exemplo sensores de temperatura ou de volume, convertendo valores digitais em processamento, interagindo o arduíno como o mundo real (WARNER, 2014).

O arduíno é um microcontrolador *single board* (placa única) soldado em um único circuito, ele possui uma pequena memória e pode comunicar-se a outros aparelhos periféricos, seus pontos fracos seriam que eles não possuem um sistema operacional e por não ter um controlador de vídeo, sua programação precisa ser feita por um computador externo conectado a ele, já o Raspberry Pi possui GPU para interface gráfica e consegue rodar diferentes sistemas operacionais (WARNER, 2014).

Outra plataforma de prototipação é o *beaglebone*, que é o maior concorrente do Raspberry Pi, pois ele possui o processador ARM Cortex-A8 e o

Raspberry Pi possui o ARM-Cortex-A7, no caso do desempenho gráfico fica melhor no *beaglebone* pois tem o processador melhor, mas o Raspberry Pi possui mais memória RAM, o Raspberry Pi possui 1GB RAM e o *beaglebone* apenas 256MB, o *beaglebone* funciona também Android e Ubuntu (BEAGLEBOARD, 2016, tradução nossa).

2.2 ARQUITETURA ARM

A arquitetura *Advanced RISC Machines* (ARM) é um conjunto de instruções de processadores de 32 bits baseada em *Reduced Instruction Set Computer* (RISC) com uso geral de 16 registradores, no total são 31 registradores, mas são somente visíveis 16, tem grande escalabilidade em sistemas embarcados como, por exemplo, o arduíno e o Raspberry (GOMES; LEITE; CAETANO, 2012).

Ela foi desenvolvida pela *Arcon* Computadores em 1983 na Inglaterra, originalmente ela foi desenvolvida para realizar operações de processamento enxutas, mas sem se desviar de seu alto desempenho, isso é possível graças as suas instruções simples, núcleo reduzido e baixo consumo de energia (GOMES; LEITE; CAETANO, 2012).

A arquitetura ARM é encontrada em mais de 75% dos processadores de 32 bits encontrados atualmente e é usada em diversos tipos de aplicações, pois apresenta uma arquitetura multifuncional e está sempre em constante atualização. Apesar de ser pouco conhecida pelo usuário final a arquitetura ARM está disponível nos processadores da Intel e da AMD e está disponível nos processadores *Broadcom* do Raspberry Pi (GOMES; LEITE; CAETANO, 2012).

As características de uma arquitetura de processadores ARM são baixo consumo energético e a potência de processamento, como falado anteriormente ele é um dos instrumentos chaves para a computação embarcada e estão substituindo a utilização de microcontroladores.

2.3 HISTÓRIA

A história do Raspberry Pi iniciou-se em meados de 2005 na Inglaterra, quando Eben Upton observou que alunos anteriores da década de 2000 chegavam à

universidade sabendo várias linguagens de programação e um grande conhecimento na área de hardware. E os acadêmicos da década de 2000 para frente só tinham conhecimento em linguagens para a Internet (SABER ELETRÔNICA, 2013).

Eben Upton decidiu que para aumentar o conhecimento dos acadêmicos que estavam entrando, queria criar algum hardware que fosse usado como instrumento de aprendizagem e de baixo custo para que os acadêmicos pudessem ter um conhecimento melhor de outras linguagens de programação e de hardware. Desenvolveu então uma plataforma de prototipagem que mais tarde se chamaria Raspberry Pi (SABER ELETRÔNICA, 2013).

Com o passar do tempo o projeto foi aumentando e foi crescendo o número de colaboradores envolvidos na construção da placa, decidiram criar então a instituição *Raspberry Pi Foundation*, aí ele deixou de ser apenas uma ferramenta de aprendizagem e se tornou uma plataforma de prototipagem eletrônica que poderia ser usada em todo o mundo (SABER ELETRÔNICA, 2013).

Em agosto de 2011 as primeiras 50 placas *Alfa* foram produzidas e com algumas modificações em dezembro de 2011 o Raspberry Pi já rodava vídeos de alta resolução em 1080p graças ao processamento gráfico do *Video Core IV* presente no seu processador (RASPBERRY PI, 2012, tradução nossa).

Em janeiro de 2012 foram leiloados alguns Raspberrys no *E-Bay* e anunciaram que os primeiros 10.000 Raspberrys estavam sendo produzidos na China, depois de alguns problemas e contratempos o primeiro Raspberry Pi teve seu lançamento oficial em 29 de fevereiro de 2012 (RASPBERRY PI, 2012, tradução nossa).

Um dos primeiros nomes que Upton deu ao seu projeto foi “ABC Micro”, mas ele foi recebendo idéias de outros colaboradores durante todo o processo de fabricação e implementação até que o nome foi mudado, o nome Raspberry significa Framboesa, mas não é a única empresa com o nome de fruta, pode-se citar também a *Apple* (Maçã). E Pi refere-se a linguagem de programação *Python* que é a linguagem de programação que é recomendada para o Raspberry (SABER ELETRÔNICA, 2013).

2.4 APLICAÇÕES

Existe uma vasta quantidade de projetos feitos com Raspberry Pi, tais como alarmes, centros de usinagem, contadores, sensores, *joysticks*, totens, robôs autônomos e até *clusters* usando vários Raspberrys. Também está ganhando cenário na criação de jogos e integração com outros jogos, por exemplo, o *Minecraft*. A maior parte deles desenvolvido na linguagem de programação *Python* que é a linguagem recomendada do Raspberry Pi (RASPBERRY PI, 2015, tradução nossa).

Os projetos envolvendo o Raspberry Pi não param de crescer todos os anos e segundo a revista oficial do da empresa tem até competições todos os anos para ver quais os melhores projetos são criados envolvendo o Raspberry Pi. Vale ressaltar que alguns projetos desenvolvidos estão em uma escala um pouco maior, por exemplo, a criação de placas de energia fotovoltaicas gerenciadas por um Raspberry ou até mesmo sensores de furacões para determinar a velocidade dos ventos para fazer futuros estudos.

O Raspberry Pi é uma placa de prototipação que foi desenvolvida para ser usado por pessoas de todas as idades, pois sua interface é simples e de fácil entendimento, ela é utilizada em diversas aplicações desde a criação de brinquedos, integração com jogos e até monitoramento de furacões, monitoramento de sinais cardíacos e gerenciamento de energia solar.

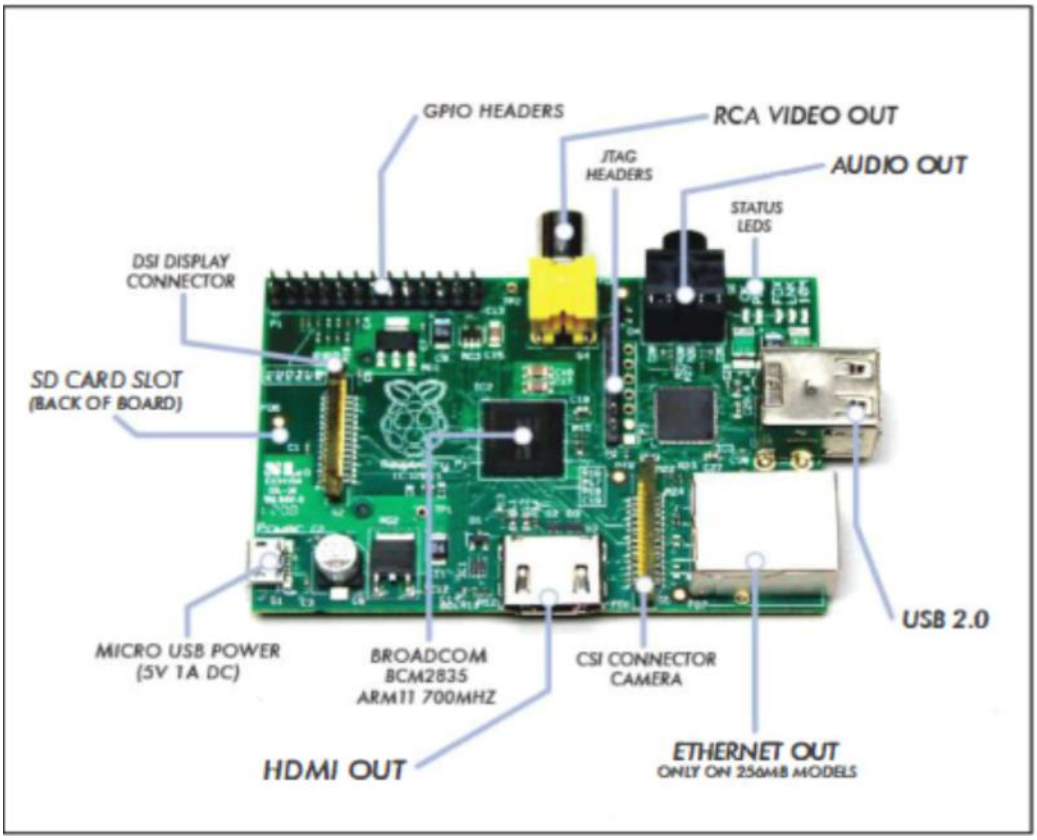
2.5 COMPONENTES

Na figura 1 é mostrado os componentes do Raspberry Pi. Para a alimentação de energia, o Raspberry Pi usa uma micro entrada *Universal Serial Bus* (USB) igual aos dos *smartphones* usados hoje em dia, a fonte de alimentação precisa fornecer no mínimo 5 volts como 700 micro amperes ou 0,7 amperes. Inclusive o Raspberry tem uma tolerância de 0,75 volts, qualquer coisa fora deste intervalo é interessante rever a parte elétrica, cabos ou fonte de alimentação, pois pode danificar o aparelho (GAY, 2014).

A entrada micro USB só pode ser usado para alimentação do aparelho, se for usada para transferência de dados de algum outro aparelho ela não irá funcionar. Apesar de o modelo B do Raspberry ser o mais usado o modelo A ainda é

encontrado em alguns distribuidores, ele funciona com 500 micro amperes ou 0,5 amperes e 3,5 volts. Embora o Raspberry precise de cinco volts de alimentação sua placa interna usa apenas 3,3 volts, isto ocorre por causa de seu regulador de tensão e do seu condensador presentes internamente no Raspberry (WARNER, 2014).

Figura 1 – Raspberry Pi



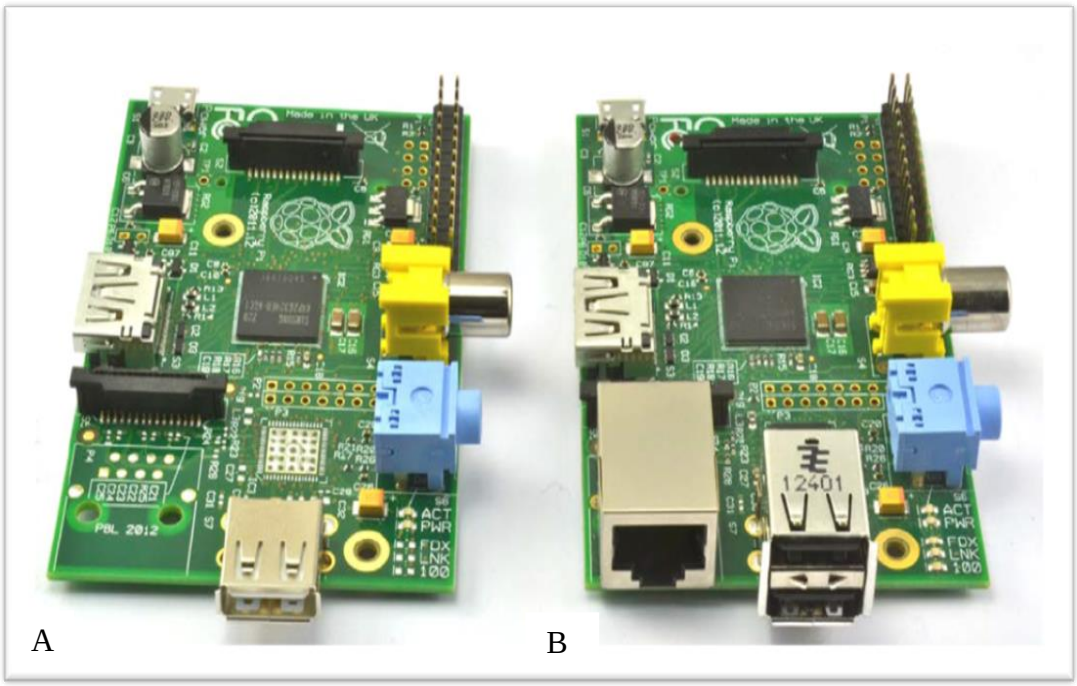
Fonte: Gupta, Patchava e Menezes (2015).

Dependendo da quantidade de periféricos conectados no Raspberry sua corrente pode chegar até 1000 micro amperes, se ultrapassar isso, os periféricos terão que ser alimentados em um *hub* USB externamente (RASPBERRY PI, 2016, tradução nossa).

A porta *High Definition Multimedia Interface* (HDMI) usa 50 micro amperes, os pinos *General Purpose Input/Output* (GPIO) podem chegar até 50 micro amperes também, mas cada pino separadamente pode chegar até 16 micro amperes, a câmera precisa 250 micro amperes para poder funcionar, e os *mouses* e teclados dependendo do modelo podem ir de 100 micro amperes até 1000 micro amperes, dependendo da potencia deles será preciso colocar em uma fonte externa de alimentação (RASPBERRY PI, 2016, tradução nossa).

A peça chave do Raspberry Pi modelo B é um processador da Broadcom: BCM 2836, sua arquitetura de 32 bits é baseada num núcleo ARM 11, modelo Cortex-A7 com 4 núcleos, é o mesmo processador que encontraríamos num Iphone 3G, este processador possui 700Mhz de processamento e 512MB de memória, entretanto os modelos mais recentes já vem com 1GB. O modelo A do Raspberry tem o processador BCM 2835 e apenas 256MB de memória, como é mostrado na figura 2 (RICHARDSON; WALLACE, 2012).

Figura 2 – Raspberry Pi: Modelos A e B



Fonte: Monk (2013).

O núcleo ARM 11 é construído baseada na arquitetura ARMv6 e tem 3 conjuntos de instruções. O primeiro é o principal e é onde a maioria do código fonte vai rodar, o segundo conjunto de instruções chamado de polegar possui 16 bits, ele é um subconjunto da primeira parte de instruções principais e ele é mais voltado para sistemas embarcados, e o terceiro conjunto refere-se ao conjunto de instruções *Java* que roda *hardware* e *softwares* na plataforma *Java* (HORAN, 2013).

O Raspberry possui um processador gráfico denominado GPU *Video Core* IV, capaz de reproduzir vídeos em *bluray* e acelerar a codificação de vídeos em 1080p no formato do *codec* de vídeo H.264, se comparar o desempenho de vídeo

deste processador gráfico seria equivalente ao primeiro *Xbox* (SABER ELETRÔNICA, 2013).

O Raspberry Pi modelo A é equipado com apenas uma entrada USB enquanto que o modelo B possui 2 entradas USB, as portas USB servem para ligar dispositivos periféricos como por exemplo *mouses*, teclados, *webcams*, caixas de som que aumentam a funcionalidade do Raspberry (RASPBERRY PI, 2016, tradução nossa).

Como as portas USB do Raspberry são ligadas diretamente ao processador dele, as portas USB da placa têm funcionalidades mais simples do que a de um computador normal, no geral ele aceita quase todos os tipos de aparelhos, mas para fornecer um nível adequado de funcionamento o sistema operacional utiliza mais processamento (RASPBERRY PI, 2016, tradução nossa).

Todos os dispositivos suportados no Linux irão funcionar no Raspberry Pi, pois o Linux tem o sistema de drivers mais completo para *hardware* do que qualquer outro sistema operacional. As portas USB na placa têm apenas uma raiz, ou seja, todo o tráfego dos periféricos é colocado em cima do processador que opera numa velocidade máxima de 480Mbps (RASPBERRY PI, 2016, tradução nossa).

A especificação USB define três tipos de velocidade para os equipamentos: Baixo, completo e alto. Os periféricos como *mouses* e teclados são de baixa velocidade, os dispositivos de áudio são de velocidade completa e as câmeras e os dispositivos de vídeo são de alta velocidade (RASPBERRY PI, 2016, tradução nossa).

Os periféricos anunciam sua própria necessidade de energia quando ligados pela primeira vez na placa, as portas USB no Raspberry tem em média uma alimentação de 100 micro amperes cada uma, mais que o suficiente para alimentar *mouses* e teclados. Dispositivos como caixas de som, câmeras e *pen drivers* precisam ser ligados em um *hub* externo com sua própria fonte de alimentação, pois consomem mais energia (RASPBERRY PI, 2016) , tradução nossa.

A plataforma suporta três tipos diferentes de entradas de vídeo: O vídeo HDMI, o vídeo composto e o vídeo *Display Serial Interface* (DSI). O mais recomendado seria utilizar o HDMI, pois fornece uma alta velocidade totalmente digital além de recursos de áudio e vídeo em *High Definition* (HD), na resolução de 1080p (UPTON; HALFACREE, 2012).

Se não tiver um monitor que não tenha HDMI não tem problema, pois a placa ainda identifica também a entrada *Digital Visual Interface* (DVI), como consequência não vai ter os recursos de áudio do HDMI e será preciso comprar um conversor de DVI para HDMI, pois o Raspberry não tem conexão DVI (UPTON; HALFACREE, 2012).

O problema é se o monitor for *Video Graphics Array* (VGA), pois a conexão VGA é analógica e não digital, além de ser já considerada antiga e a plataforma não reconhece esta conexão (UPTON; HALFACREE, 2012).

O conector redondo amarelo que se encontra na placa é chamado de conexão *Radio Corporation of America* (RCA) ou vídeo composto, já não é mais utilizada esta conexão, ela servia para conectar a plaquinha a televisores antigos e não há recursos para áudio nesta conexão (WARNER, 2014).

O modelo B do Raspberry inclui uma terceira interface de vídeo chamada DSI, bem pouco utilizada, ela conecta a plataforma a tablets e celulares, mas esta interface não tem total acesso a GPU da placa ainda (WARNER, 2014).

O Raspberry Pi modelo B possui uma entrada de rede LAN9512 SMSC, a capacidade de se conectar na Internet através de cabos é um dos principais diferenciais do modelo B, a vantagem disso é que pode-se fazer vários tipos de atividades com ele, por exemplo, serviços *web* e monitoramento de redes, já o modelo A não tem entrada de rede mas tem um módulo de rede sem fio USB que pode ser instalado nele, mas a configuração do módulo USB e a configuração da placa precisará ser feita toda remotamente (MONK, 2013).

Se observar atentamente a entrada de rede do Raspberry Pi é integrada com as outras 2 USB’s, é como se tivesse 3 USB’s, esta “terceira” porta USB é conectada internamente no chip da internet lhe adquirindo funções de rede, usaram isto para economizar espaço na placa e para que ele não ficasse mais caro, por esta razão também a plataforma não pode ser rede giga, apenas rede mega (10/100), pois o USB tem uma vazão máxima de 60Mb’s o que não é suficiente para uma rede giga (HORAN, 2013).

Sua conexão de rede ainda pode desempenhar várias funções tais como funcionamento dos protocolos *Transmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP), funciona também o *Dynamic Host Configuration Protocol* (DHCP) e ainda faz o cálculo do *checksum* dos protocolos TCP e UDP, ou seja,

verifica a integridade dos dados transmitidos em determinado canal em vez de deixar esta tarefa para a CPU (HORAN, 2013).

Para armazenamento e instalação do sistema operacional o Raspberry Pi precisa de um cartão *micro Secure Digital* (SD), pois a plaquinha não tem armazenamento interno, no mínimo o cartão SD precisa possuir espaço de 4GB (WARNER, 2014).

Os cartões de memória são classificados em classes, que é um indicador para leitura e gravação de dados, o cartão de memória é dividido nas classes 2, 4, 6, 8 e 10, a classe 2 executa dados a 2MB por segundo, a classe 4 a 4MB por segundo, a classe 6 a 6MB por segundo, a classe 8 a 8MB por segundo e a classe 10 a 10MB por segundo. Sobre o tamanho do cartão eles são divididos em duas variedades: *Secure Digital High Capacity* (SDHC) e *Secure Digital Extended Capacity* (SDXC). A principal diferença é que os cartões SDHC vão até 32GB, e os Cartões SDXC vão até 2TB (WARNER, 2014).

O cartão micro SD possui um controlador interno, conhecido como um processador de armazenamento *flash*. O cartão SD gerencia os dados com um tamanho de setor de 512 bytes, é este tamanho para a compatibilidade com os sistemas operacionais existentes (GAY, 2014).

A taxa de erros dos cartões SD é muito menor do que em discos magnéticos e quando acontecem erros no cartão SD o próprio cartão solta um protocolo corrigindo os problemas para prevenir problemas futuros irreparáveis, estas operações são rodadas em segundo plano invisíveis ao usuário (GAY, 2014).

Existe também no Raspberry Pi uma conexão de áudio analógica 3,5 mm, ele é o mesmo conector de áudio de fones de ouvido e microfones do computador, mas tem baixa qualidade comparada a entrada HDMI do Raspberry que também fornece áudio, a entrada de áudio do Raspberry também funciona as caixas de som do computador, para o usuário que precisar de alguma entrada de som mais fixa (UPTON; HALFACREE, 2012).

O Raspberry Pi modelo B possui uma conexão serial para a instalação de uma *webcam*, o modelo é uma Omnivision OV5647 de 5 megapixels que captura imagens com uma resolução de pixels de até 2592x1944 e registros com resolução de até 1080p. O conector da câmera é uma *Camera Serial Interface* (CSI) que é um

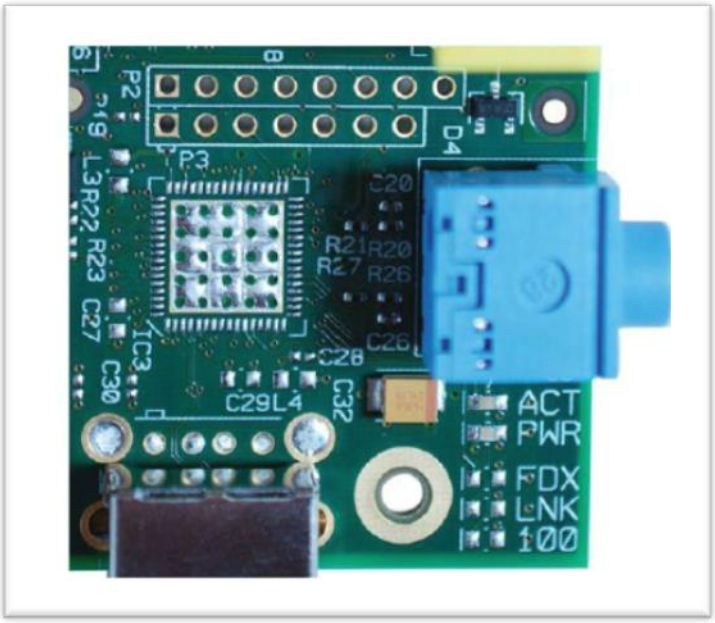
cabo de fita flexível de 15 pinos parecido com um cabo *Integrated Drive Eletronics* (IDE) de *Hard Disks* (HD's) mais antigos (WARNER, 2014).

Por último, o Raspberry Pi possui uma conexão de 26 pinos que serve para expandir o Raspberry com outros *hardwares*, microcontroladores e robôs, a conexão é denominada de *General Purpose Input/Output* (GPIO) (WARNER, 2014).

A conexão GPIO pode fazer o Raspberry ser uma conexão de entrada, de saída ou funcionar como uma porta, isso torna o Raspberry a se adaptar com várias plataformas, mas os pinos são frágeis à eletricidade estática e suas placas de expansão não podem ultrapassar os 50 micro amperes suportados pela conexão, e como serão adicionados *hardwares* extras, acaba aumentando o custo também (GAY, 2014).

Quando o Raspberry é ligado dependendo do que está conectado nele, ele acende alguns *leds* em sua placa. Se o *led* ACT estiver verde quer dizer que o cartão micro SD está instalado, se o *led* PWR estiver vermelho quer dizer que o Raspberry está sendo alimentado a 3,3 volts, se o *led* FDX também da cor verde estiver aceso que dizer que o cabo de rede está conectado, se o *led* LNK estiver aceso em verde também quer dizer que está trafegando dados, e se o *led* 100 da cor amarela estiver ligado quer dizer que a rede está conectada em 100Mbps, na figura 3 é mostrado onde são encontrados estes *leds* (WARNER, 2014).

Figura 3 – Leds Raspberry Pi



Fonte: Warner (2014).

O Raspberry Pi pode ser utilizado por vários sistemas operacionais, a grade parte deles na plataforma Linux. O Ubuntu Mate pode ser executado no Raspberry Pi, a versão do Ubuntu utilizado nos desktops pode ser usada também, mas recebe o nome de *Ubuntu Snappy Core*, a própria Microsoft entrou na área de computação embarcada e lançou o Windows 10 *Internet Of Things* (IoT – Internet das Coisas) destinado aos Raspberrys, outros sistemas operacionais como o OSMC, o OPENELEC e o Risc OS também estão disponíveis para o Raspberry Pi (RASPBERRY PI, 2016, tradução nossa).

A própria Raspberry lançou uma versão de sistema operacional mais destinada ao público infantil chamada PiNet, mas a própria Raspberry Pi recomenda seu sistema operacional Linux chamado Raspbian baseado no Linux Debian (RASPBERRY PI, 2016, tradução nossa).

O objetivo do trabalho é demonstrar como esta placa de prototipação de baixo custo se comporta e como ela se adapta aos sistemas operacionais Linux Raspbian e Windows 10 IoT para Raspberry Pi.

3 LINUX

O termo Linux é designado para sistemas operacionais que utilizam o kernel Linux, que foi desenvolvido por Linus Torvalds com base no sistema operacional *Minix,* sua licença é livre, ou seja, é possível usa-la, estuda-la, modifica-la livremente de acordo com os termos regidos por ela. O Linux é apoiado por grandes empresas, tais como a *Hewlett-Packard, Red Hat, Oracle, Google, Mandriva* e *Canonical* (ALENCAR, 2005).

3.1 HISTÓRIA

Um sistema operacional é um componente de software que fornece a interface e a comunicação entre os programas e os periféricos do computador (*mouses*, teclados, impressoras), gerenciados por um *Kernel* que é a principal parte de um sistema operacional onde se encontra todas as ferramentas e tarefas de gerenciamento (CAMPOS, 2003).

Durante a criação da *Advanced Research Projects Agency Network* (ARPANET), onde foi o marco que iniciou para o aparecimento da Internet, os programadores dos laboratórios da *Bell* e da *American Telephone and Telegraf* (AT&T), sentiram a necessidade de um sistema operacional para gerenciar seus diversos projetos (PEREIRA, 2006).

Então a Bell desenvolveu seu sistema operacional *Bell Operating System* (BESYS) para ser utilizado no seu computador. No entanto, outras pessoas e empresas se interessaram pelo seu sistema operacional, então a *Bell* decidiu fornecer seu sistema de forma gratuita, mas sem oferecer nenhum tipo de suporte técnico (PEREIRA, 2006).

Na era da terceira geração de computadores em 1964, a *Bell* tinha que avaliar se utilizaria um sistema operacional criado por outras empresas ou se tinha que desenvolver o seu próprio sistema operacional. Então desenvolveram internamente o seu sistema operacional ao qual chamaram de *Multiplexed Information and Computing Service* (*Multics*), realizando parcerias com as empresas *General Electric* e com o MIT (*Massachusetts Institute of Tecnology*) (PEREIRA, 2006).

Então em 1969 a AT&T decidiu retirar-se do projeto, pois sabia que o desenvolvimento do *Multics* teria um custo relativamente alto e o tempo de execução era muito comprido e demorado (PEREIRA, 2006).

Depois de um tempo alguns dos participantes no projeto *Multics* com destaque para Ken Thompson e Ritchie, resolveram programar uma versão mais simples e menor do *Multics*. Programaram um sistema de arquivos e um interpretador de comandos digitados pela pessoa que iria utilizar o sistema, que foi designado de *Shel* (PEREIRA, 2006).

O tempo foi passando e aos poucos as coisas iam tomando um rumo melhor e mais evoluído. Em pouco tempo esse sistema ganhou o nome de *Uniplexed Information and Computing Service* (*Unics*) e posteriormente ganhava o nome de UNIX (PEREIRA, 2006).

No início da década de 80 foi criado um ambiente gráfico, o *X-Windows*, que tornava padrão em todas as distribuições UNIX e o protocolo TCP/IP desenvolvido pela *Defense Advanced Research Projects Agency* (DARPA), foi usado para a comunicação à distância entre sistemas UNIX, esse que mais tarde viria a se tornar o padrão de Internet utilizado até os dias de hoje (PEREIRA, 2006).

No ano de 1985, foi criada uma instituição designada de *Free Software* Fundation (FSF), seu objetivo era o desenvolvimento e a proteção do software livre. Como o objetivo desta fundação era proteger o software livre, foi criado um novo tipo de licença, chamada de *General Public Licence* (GPL), autorizando assim a qualquer pessoa ou empresa a permissão de modificar e melhorar todo o software protegido por ela (PEREIRA, 2006).

O molde que foi designado para o desenvolvimento do sistema operacional que a FSF queria desenvolver foi o UNIX. A partir do UNIX, surgiu a palavra GNU, que significa *GNU is Not Unix* (PEREIRA, 2006).

O Linux que é nada mais que um *kernel* foi escrito por Linus Torvalds, do Departamento de Ciência da Computação da Universidade de Helsinki, na Finlândia, no ano de 1991, com a ajuda de muitos programadores voluntários. Linus Torvalds começou a programação do *kernel* inspirado pelo seu interesse no *Minix*, um pequeno sistema UNIX desenvolvido por Andrew S. Tanenbaum (ALENCAR, 2005).

O termo Linux foi criado por Ari Lemmke, que era o dono do diretório *File Transfer Protocol* (FTP) na qual o núcleo Linux estava disponível. Anteriormente,

Linus havia nomeado o *kernel* de “Freax”. Em 5 de outubro de 1991, Linus Torvalds anunciou a primeira versão “oficial” do núcleo Linux, a versão 0.02 (ALENCAR, 2005).

Linus Torvalds comentou sobre a possibilidade da utilização de um pinguim como símbolo do Linux, pois ele era um de seus animais favoritos, na Finlândia existem muitos pinguins, alguns dizem que ele foi mordido por um deles. Batizou o pinguim de Tux, o nome vem das iniciais de Torvalds e da primeira e última letra da palavra Unix, daí o nome Tux (MELO, 2008).

3.2 LICENCIAMENTO

O *software* livre é um tipo de *software* que atende um determinado número de exigências, nas quais é necessário ter o livre acesso ao código fonte, à permissão de efetuar mudanças ao programa original, e a distribuição dessas alterações, sendo que a licença não poderá discriminar pessoas e outras áreas de pesquisa (CARDOSO, 2000).

O Linux é um sistema desenvolvido sob o modelo *Open Source*, tornando assim um software de utilização livre, para quem quiser utiliza-lo. Isto permite que qualquer pessoa tenha acesso ao seu código fonte e que qualquer um possa contribuir com o projeto, seja no seu desenvolvimento, correção de bugs, e em sua documentação, desde que seja de forma livre (PEREIRA, 2006).

Para que um programa seja *Open Source* não é apenas deixar o código fonte aberto, é necessário ter alguns requisitos como: Redistribuição livre, ou seja, é proibida a venda do *software*, respeitando as normas do *software* da empresa proprietária. Acesso ao código fonte: O programa deve incluir o código fonte e sua distribuição pode ser compilada ou não, e precisa estar em um formato legível para que futuros desenvolvedores possam utilizar e mexer no código (PEREIRA, 2006).

Também é necessário não haver nenhum tipo de discriminação contra nenhuma pessoa e nenhum tipo de grupo, e nem contra outras áreas de estudo ou de pesquisa. A licença precisa ser distribuída a todos do mesmo jeito sem a necessidade de nenhum outro tipo de licença adicional, a licença também não pode ser exclusiva de um único produto e ela tem que ser aberta a todos os tipos de tecnologias (PEREIRA, 2006).

As distribuições GNU/Linux possuem em comum o mesmo *kernel*, mas elas foram construídas por diferentes pessoas, grupos e empresas que possuem ideias comerciais e sugestões diferenciadas, além do objetivo de atender a sua própria necessidade e de outros fornecedores (MELO, 2008).

3.3 VERSÕES

Segundo Melo (2008), no Brasil as principais distribuições GNU/Linux mais usadas são estas:

a) a trindade *Slackware*, *Debian* e *Red Hat/Fedora*;

b) as variantes *Mandriva* e *OpenSuSE/SuSE;*

c) a meta-distribuição *Gentoo* (e *Debian*);

d) as especializadas *Ubuntu/Kubuntu* e *Kurumin*.

As distribuições *Slackware, Debian* e *Fedora* são as mais tradicionais e foram as primeiras para uso geral em diversas aplicações, são as distribuições mais antigas e ainda contam com um bom número de usuários no mundo, a maior parte das distribuições atualmente são derivadas desta trindade (MELO, 2008).

Para o uso em empresas, a *Red Hat* foi à distribuição que mais desempenhou bem o seu papel e é modelo de referência para muitas outras, o *Debian* e a *Slackware* são popularmente conhecidas por serem leves e flexíveis a vários tipos de aplicações, que propiciam um alto grau de customização (MELO, 2008).

A *Slackware* é a distribuição mais antiga e tradicional do Linux, desenvolvida por Patrick Volkerding, sendo ainda muito utilizada nos dias de hoje. A *Red Hat* é uma das principais distribuições em desenvolvimento para o uso em servidores, tendo atualmente um grande público neste mercado, a *Red Hat* torna-se muito simples pela configuração do sistema, aplicativos de instalação e configuração que buscam facilitar ao máximo o uso desta distribuição, já o *Fedora* apesar de ser parecido com a *Red Hat* seu uso é mais exclusivo em desktops e para usuários finais (MELO, 2008).

A distribuição *Debian* foi desenvolvida inicialmente por Ian Murdock em 1993, sua principal característica é de ser a distribuição Linux de grande lealdade aos conceitos de liberdade do *software* livre, contendo somente pacotes de código

aberto em sua distribuição. O *Debian* é a distribuição oficial do Projeto GNU e conta com algo em torno de 20.000 pacotes inclusos (MELO, 2008).

O *Debian* pode ser executado em quase todos os tipos de computadores, incluindo os mais antigos. Quase todo os tipos de *hardware* são suportados, inclusive na computação embarcada e em plataformas de prototipação como no caso do Raspberry Pi. O nome *Debian* vem do nome de seu criador, Ian Murdock, e sua esposa, Debra (DEBIAN, 2016, tradução nossa).

As distribuições variantes foram baseadas na antiga *Red Hat*, como exemplo delas são o *Mandriva* e *SuSE* / *OpenSuSE*. O *Mandriva*, ou melhor, a *Mandriva S.A*. é resultado da união de duas empresas de *software* livre ocorrida em 2005, a *MandrakeSoft* e a *Conectiva* (MANDRIVA, 2015, tradução nossa).

Suas características são: Comunidade forte e atuante em língua portuguesa tem um dos maiores repositórios de pacotes Linux; suporte a vários idiomas; atualizações constantes e lançamento de novas versões periodicamente (MANDRIVA, 2015, tradução nossa).

Criado em 1992, o *SUSE* é a distribuição empresarial do Linux mais utilizada na Europa e a plataforma com o maior número de operações para ambientes de computação. Dentre seus principais recursos, está o suporte a 12 idiomas diferentes (SUSE, 2016, tradução nossa).

Esta foi uma das primeiras distribuições a lançar seu *software* gravado tanto em CD-ROM quanto em DVD-ROM em um único pacote. Seu símbolo é representado por um camaleão, referente à adaptação e flexibilidade (SUSE, 2016).

Meta-distribuições são aquelas distribuições que possuem grande capacidade de adaptação para os mais diversos tipos de tarefas, as distribuições são *Debian* falado anteriormente e o *Gentoo* (MELO, 2008).

O *Gentoo* é uma meta-distribuição programada com a finalidade de ser extremamente otimizado, o nome deriva-se de uma raça de pinguins que tem como característica de serem pequenos, leves e ágeis. O *Gentoo* consegue obter um excelente desempenho, que realiza a operação apenas com a compilação do código fonte dos programas. Devido as suas características, é uma distribuição recomendado para usuários mais experientes (MELO, 2008).

As distribuições especializadas foram desenvolvidas para um grupo específico de usuários ou para um público exclusivo. O *Ubuntu*, que seu nome em

africano significa "humanidade aos outros", foi criado por Mark Shuttleworth em 2004, sua distribuição possui um repositório com mais de 16.000 pacotes e é patrocinado pela *Canonical* (MELO, 2008).

O *Kurumin* foi desenvolvido por Carlos E. Morimoto no ano de 2003, para ele funcionar, é preciso de poucos recursos de *software* e *hardware*, com o *Kurumin*, pode-se utilizar o sistema operacional inicializando-o diretamente pelo CD, tornando-se muito prático, e pode ser utilizado o *Kurumin* de qualquer computador, basta apenas rodá-lo direto do CD (MELO, 2008).

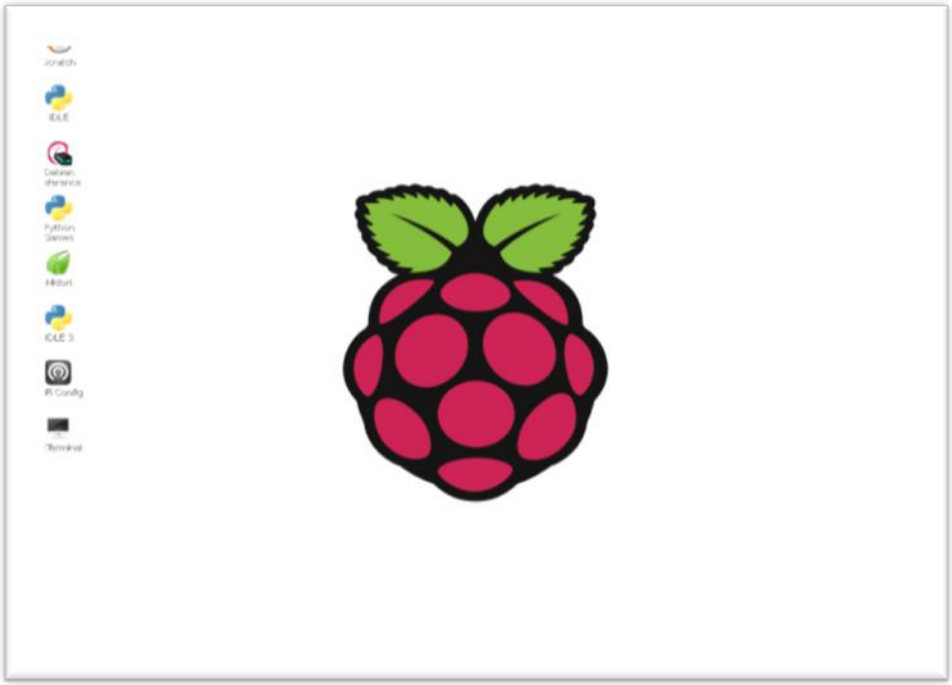
3.3.1 Linux Raspbian

O Raspbian é um sistema operacional gratuito baseado no Linux *Debian* e é um sistema operacional adaptado para o Raspberry Pi, ele vem com mais de 35.000 pacotes e softwares pré-compilados. O sistema ainda está em desenvolvimento para aperfeiçoar ainda mais o Raspberry Pi e ele foi desenvolvido por um grupo de fãs e admiradores do Raspberry. A versão atual do Raspbian é a versão Jessie (RASPBERRY, 2016, tradução nossa).

O sistema operacional é a distribuição ideal para quem possui menos conhecimentos dos sistemas Linux, pois já é um sistema que quando instalado vem praticamente pronto. O Raspbian é um sistema quase completo, já que vem com diversas aplicações pré-instaladas, os *drivers* mais importantes instalados e ferramentas para facilitar algumas configurações necessárias. Muitas aplicações e módulos dedicados à programação já vêm incluídos na imagem do SO, bastando apenas iniciar o sistema para acessá-los (PEREIRA; JUCÁ, 2015).

O Raspbian foi o sistema operacional escolhido pela *Raspberry Pi Foundation* como sistema padrão do Raspberry, pois inclui um dos gerenciadores de pacotes mais flexíveis. O *Debian* e o *Ubuntu* não suportam a arquitetura ARM encontrados no processador do Raspberry Pi, mas a *Raspberry Pi Foundation* modificou o *kernel* do *Debian* para acomodar o processador ARM do Raspberry Pi, na figura 4 é mostrada a interface gráfica do Raspbian (WARNER, 2014).

Figura 4 - Interface gráfica do Raspbian



Fonte: Getting Started with Raspberry Pi (2012).

4 WINDOWS

O Windows é um sistema operacional desenvolvido pela Microsoft, ele está presente em mais de 90% dos computadores no mundo, desenvolvido por Bill Gates e Paul Allen, sua interface gráfica é toda em janelas, dai o nome Windows.

4.1 HISTÓRIA / VERSÕES

A história da Microsoft inicia-se na década de 1970 quando dois jovens chamados Bill Gates e Paul Allen perceberam que o futuro da computação estava nos computadores pessoais. Em 1975 eles formaram uma parceria e fundaram a Microsoft, onde a visão deles era levar o computador para todas as pessoas usarem em suas próprias casas o que antes era impossível, pois os computadores ocupavam uma sala inteira (MICROSOFT, 2015).

Em 1980 Bill e Paul contrata um antigo amigo de universidade para ajudar eles a comandar a empresa, o nome dele era Steve Ballmer, depois disso a Microsoft dá enfoque a criação de um sistema operacional para gerenciar o *hardware* dos computadores que existiam na época, conseguiram de uma loja de computadores da época a base desse sistema, o nome da loja era *Seatle Computadores,* compraram por 50 mil dólares o sistema operacional (MICROSOFT, 2015).

Depois de algumas modificações em sua estrutura e em seu núcleo batizaram o novo sistema de MS-DOS que significa Sistema Operacional em Disco da Microsoft (*Microsoft Disk Operating System*) (MICROSOFT, 2015).

Em 1985 a Microsoft anuncia seu próprio sistema, mas baseado no *kernel* do MS-DOS, que não funcionava mais com comandos de terminal, mas sim possuía uma interface gráfica com botões e menus, o nome do sistema ia ficar *Interface Manager* (Gerenciador de Interfaces), mas o nome final ficou como Windows, pois a interface gráfica do sistema lembrava “janelas”, este ficou conhecido como Windows 1.0 (MICROSOFT, 2015).

Em 1987 a Microsoft anuncia a versão 2.0 do Windows, as melhorias dessa versão foram o início do aparecimento de ícones na área de trabalho e atalhos do teclado, aumentando a usabilidade do sistema e em 1988 a Microsoft se

torna a maior vendedora de computadores pessoais do mundo, nesta época muitas empresas começam a usar os computadores em suas atividades (MICROSOFT, 2015).

Na década de 1990 a Microsoft lança o Windows 3.0 e em 1992 já lança o Windows 3.1, dessa época em diante o Windows se torna o sistema operacional mais usado no mundo, a partir destas duas versões sua interface gráfica é a que mais se aproxima da dos dias de hoje, usando interface gráfica de 16 bits, nesta versão os destaques são o gerenciamento de impressão e a permissão para o usuário instalar outros programas (MICROSOFT, 2015).

No ano de 1993 a Microsoft anuncia o Windows NT que é a primeira versão do Windows construída a partir do zero, a diferença em relação às outras versões é a plataforma gráfica de 32 bits (MICROSOFT, 2015).

Em 1995 a Microsoft anuncia o Windows 1995, esta é a primeira versão que conta com o menu iniciar e recursos para a Internet e uma maior quantidade de *drivers* para a instalação de diversos periféricos, o maior diferencial desta versão é que cada janela aberta tem os botões de minimizar, maximizar e fechar e também a primeira versão do *Internet Explorer*, encontrado até hoje nas versões atuais (MICROSOFT, 2015).

No ano de 1998 a Microsoft lança o Windows 98, que é a última versão do Windows baseado no MS-DOS, as novidades desta versão são os recursos para utilização USB e para a leitura de CD’s e DVD’s (MICROSOFT, 2015).

O Windows ME lançado na década de 2000 foi um sistema mais voltado para o uso doméstico, suas maiores funcionalidades foram às aparições do aplicativo Restauração do Sistema, onde caso fosse feita alguma modificação no Windows que desse errado, era possível restaura-lo para um ponto anterior e também do aparecimento do *Windows Media Player* para reprodução de músicas e do *Windows Movie Maker* para edição de vídeos (MICROSOFT, 2015).

O Windows 2000 Professional foi uma atualização do Windows ME, com melhoramentos na parte de instalação de novos drivers e periféricos além de maior estabilidade na instalação de dispositivos USB e aparelhos de comunicação sem fio (MICROSOFT, 2015).

No ano de 2001 a Microsoft lança uma de suas versões mais estáveis e vendidas, o Windows XP com uma interface gráfica totalmente nova além de possuir

a versão para se usar em casa, que é a versão *Home Edition* que conta já com o *Windows Live Messenger*, e a versão para empresas, que é a versão *Professional.* As versões mais novas do XP possuem a versão de 64 bits (MICROSOFT, 2015).

Em 2006 a Microsoft investe no Windows Vista que é considerado uma das versões do Windows que possui a maior segurança para vírus de todas as versões anteriores, possuindo várias versões como Windows Vista Starter, Windows Vista Business (Para as empresas), Windows Vista Home Basic, Windows Vista Home Premium e Windows Vista Ultimate, na versão Ultimate o diferencial é a maior criptografia na unidade de dados que fornece uma melhor segurança aos dados do usuário (MICROSOFT, 2015).

O Windows 7 foi anunciado em 2009, ela ainda é uma das versões mais usadas até hoje por ser bem estável e por possuir uma interface gráfica bem intuitiva ao usuário, o destaque desta versão é o *Windows Touch*, onde é possível, se o monitor do computador tiver este recurso, o uso do Windows com o toque na tela como se fosse um celular (MICROSOFT, 2015).

Em 2012 a Microsoft remodela novamente todo o seu sistema operacional, e lança o Windows 8, sua interface gráfica é toda em blocos, também chamada de interface *Metro*, o que fica mais fácil o uso principalmente usando o toque na tela, mas ele ainda possui a interface da Área de Trabalho igual aos outros Windows, não foi muito aceito pelos usuários pela falta do menu iniciar (MICROSOFT, 2015).

No ano de 2013, é anunciado o Windows 8.1 que seria mais uma atualização do Windows 8, a Microsoft coloca de volta o menu iniciar, um dos diferenciais desta versão é que quando o Windows é inicializado, em vez de cair na tela de blocos, já entra direto na área de trabalho do Windows igual as outras versões (MICROSOFT, 2015).

4.2 LICENCIAMENTO

A Microsoft disponibiliza vários tipos de licenças em suas versões, as mais conhecidas são as licenças *Original Equipment Manufacturer* (OEM) e a *Full Packaged Product* (FPP), as licenças do tipo OEM vem junto com a máquina comprada, ou seja, na compra do computador a Microsoft já disponibiliza a licença

pré-instalada na máquina, e as licenças FPP vem separadas normalmente em embalagens com o CD de instalação do sistema e a chave de ativação, a chave de ativação serve para ver se a licença do Windows é original (MICROSOFT, 2016).

Existe também a *Open License* (VL) que as licenças do SO ficam armazenados no servidor da Microsoft para acessar as licenças é preciso ter uma conta da Microsoft, onde pode ser usada à mesma chave de ativação em várias máquinas, é muito comum este tipo de licença nas empresas ou órgãos do governo onde há um grande número de equipamentos (MICROSOFT, 2016).

É possível também a opção de *downgrade* da licença do Windows, ou seja, é adquirida uma licença do sistema operacional mais recente, mas tem-se a necessidade de utilizar uma versão do SO mais antiga, por motivos de incompatibilidade de programas ou outros recursos, ai pode-se usar a mesma chave de ativação do sistema mais novo para o Windows versão mais antiga também, mas nem todas as versões são compatíveis para *downgrade* (MICROSOFT, 2016).

4.3 WINDOWS 10

A última versão do Windows, chamada de Windows 10 foi lançada em 2015, o sistema operacional vem com uma interface totalmente modificada, dando mais enfoque a usuários mais antigos, o menu iniciar volta diferente e a interface do sistema inteiro volta mais intuitiva ao usuário, também tem a chegada de um novo navegador para a Internet o *Microsoft Edge* (MICROSOFT, 2015).

O Windows 10 é a primeira versão da Microsoft que foi disponibilizada gratuitamente ao público, essa atualização gratuita ficará no ar até o dia 29 de julho de 2016 completando um ano desde que a Microsoft lançou o novo sistema, depois disso terá que ser adquirida por algum revendedor ou pelo site da Microsoft, todas as outras versões são compradas. A licença do Windows 10 no site da Microsoft custa em torno de 469,99 reais.

4.3.1 Windows 10: Raspberry Pi

O Windows 10 também funciona em plataformas menores e de pouco processamento, a Microsoft chama de Windows 10 *Internet Of Things* (IoT – Internet

das coisas), esse SO funciona em dispositivos que possuem monitor ou não, ele roda no Raspberry Pi e também aceita os recursos do arduino (MICROSOFT, 2016, tradução nossa).

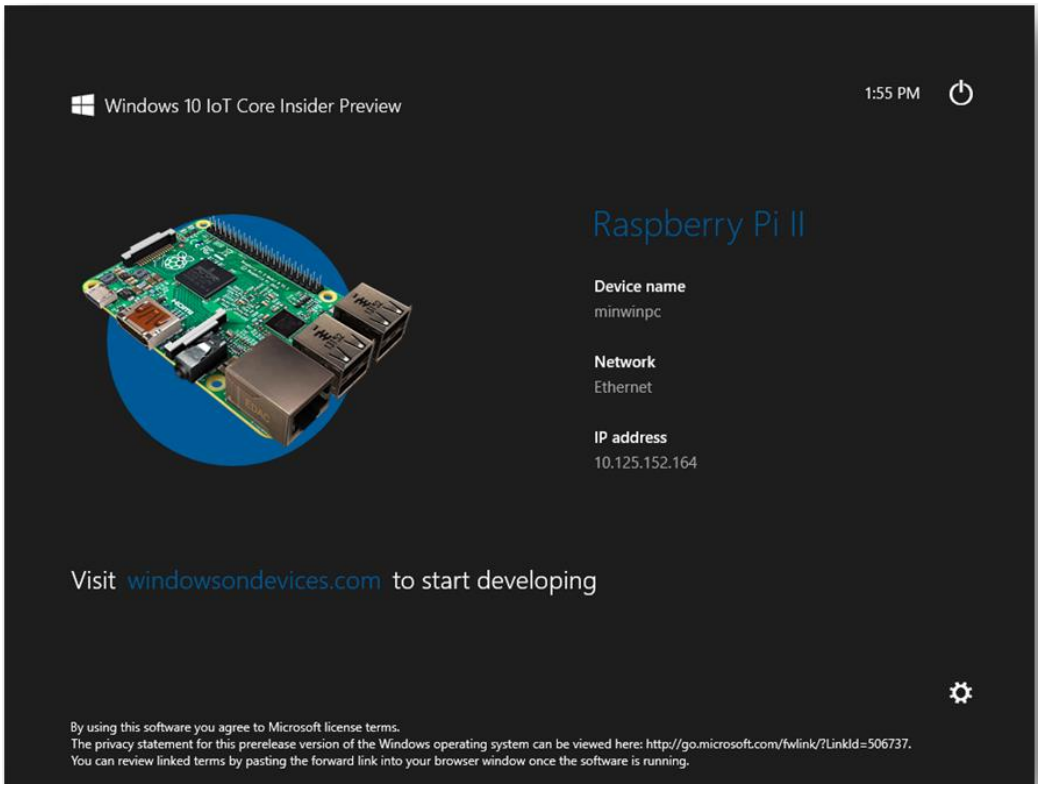
Para instalar o Windows 10 IoT nestes dispositivos precisam ter os seguintes requisitos de *hardware*: No mínimo 256 MB de memória RAM e 2 gb de armazenamento, o recomendado é 512 MB de memória, e o processador precisa ter no mínimo 400 MHz de processamento (MICROSOFT, 2016, tradução nossa).

A licença do Windows 10 IoT é gratuita para baixa-la é preciso somente ter uma conta da Microsoft, para instala-lo primeiramente é preciso colocar um cartão de memória em um computador normal ou *notebook* e colocar a imagem do Windows IoT dentro do cartão de memória e depois colocá-lo no Raspberry.

O SO é dividido em três versões: O *Windows 10 IoT for industry devices* para sistemas e máquinas com processamento e memória mais altos, o *Windows 10 IoT foi mobile devices* para celulares e tablets e o *Windows 10 IoT Core* para plataformas de prototipação como o Raspberry Pi, como mostrado na figura 5.

O sistema operacional que será utilizado neste projeto é o da versão IoT Core, pois é apropriado para plataformas de prototipação como o Raspberry Pi, e também para outras placas como por exemplo as placas Dragonboard e Minnowboard.

Figura 5 - Windows 10 IoT



Fonte: Microsoft (2016).

5 MÉTRICAS DE DESEMPENHO

As métricas de desempenho servem para avaliar a performance de todo um sistema ou parte dele, as métricas são definidas pela pessoa ou organização que irá realizar os estudos sobre determinado sistema ou ferramenta, algumas métricas podem avaliar a performance do sistema todo, algumas métricas apenas uma parte dele (IBM, 2016, tradução nossa).

5.1 DEFINIÇÃO

As métricas de desempenho são valores brutos que servem de base para algum tipo de pesquisa ou estudo, eles podem ser compostos por valores, pesos, volumes ou qualquer outro formato que expresse quantidade (ELIAS, 2016).

As métricas têm como objetivo principal as funções de coletas de dados de avaliação e desempenho, como elas usam valores quantitativos para a coleta de dados elas serão exibidas em formas de indicadores, ou seja, é o conjunto das métricas para a compreensão de um produto ou de alguma coisa. As métricas são divididas em três fases: a coleta dos dados, o cálculo destes dados e por último a sua análise (GUARIZZO, 2008).

5.2 BENCHMARKS

A metodologia de *benchmarks* é uma das técnicas mais utilizadas para avaliar métricas de qualidade nas empresas, essa metodologia procura melhorar os processos empresariais e aumentar a competição para com outras empresas, onde essa metodologia traz vários avanços para dentro das empresas (GOMES, 2001).

5.2.1 Definição

Há várias definições para a metodologia de *benchmarking*. Alguns tipos de autores definem que o *benchmarking* vem da palavra japonesa *dantotsu* que significa os processos de pesquisa e exploração dos pontos fortes das empresas concorrentes a fim de melhora-los (MENEGUELLI et al., 2007).

A primeira aplicação técnica da metodologia ocorreu na empresa americana Xerox em 1979, com o decorrer de mudanças dentro desta organização, os diretores da empresa procuraram uma forma de melhorar o desempenho de seus processos, em volta disso, eles desenvolveram um programa de *benchmarking*, que fornecia informações sobre o desempenho da organização para a diretoria da empresa para avaliar seus processos com as demais empresas concorrentes (MENEGUELLI et al., 2007).

O processo é definido como uma metodologia sistemática e de processo contínuo para realizar a avaliação de processos de trabalho ou de produtos a fim de melhorar o desempenho da organização (SPENDOLINI, 2003).

Para o chefe executivo da Xerox David T. Kearns o *benchmarking* é um processo que serve para medir o desempenho de produtos, serviços e processos para competir com outras empresas (LACOMBE, HEILBORN, 2003).

O propósito da metodologia é confrontar dados, processos, procedimentos e situações dentro de uma organização, avaliar estes dados e compara-los com experiências de outras empresas promovendo seu crescimento. Esta metodologia serve para ver o que outras empresas fazem corretamente e o que traz resultados positivos e ao mesmo tempo identifica problemas ou defeitos em seus processos de trabalho e em seus concorrentes (MENEGUELLI et al., 2007).

5.2.2 História

Os antecedentes históricos que se remetem ao uso das tecnologias de *benchmarking* vêm da época de Frederick Taylor, sua metodologia baseava-se na cronometragem e estudos de tempo das linhas de produção, para fazer possíveis estudos de como melhorar as tarefas e aumentar a produção com estudos de tempo (GOMES, 2001).

Também foram encontrados dados do uso deste processo na Segunda Guerra Mundial a fim de verificar padrões empresarias que comparassem empresas umas com as outras em termos de segurança, cargas de trabalhos entre outros fatores (GOMES, 2001).

Em 1979 a empresa Xerox começou originalmente a utilizar *benchmarks* competitivos, ou seja, para verificar e melhorar suas falhas e acertos no processo de

produção, e tentar compara-las com outras empresas para verificarem onde podiam melhorar sua produção interna e a qualidade de seus produtos (GOMES, 2001).

Após o uso na empresa Xerox muitas outras empresas começaram a empregar o uso desta metodologia, no começo apenas poucas unidades operacionais usufruíram do uso do *benchmarking*, mas no ano de 1981, ele foi difundido em todas as corporações para o aumento da qualidade do produto (GOMES, 2001).

5.2.3 Tipos

A metodologia de *benchmarking* se divide em três tipos: Informal, Semi-Formal e Formal, na tabela abaixo estão diferenciados os principais tipos de *benchmarking* e como funciona cada um deles:

Tabela 1 – Tipos de Benchmarking

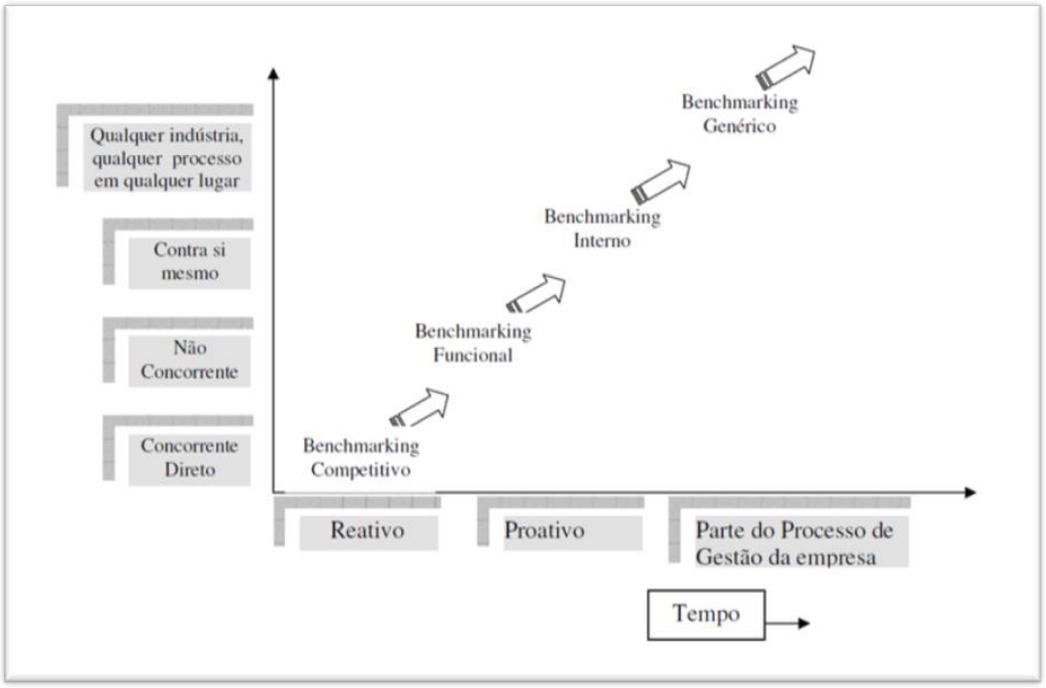
	Informal	Semi-Formal	Formal
Frequência	Irregular	Frequente	Contínuo
Organização	Individual ou grupos de trabalho	Equipes criadas por função	Equipes formais e especialistas
Relação com outros sistemas	Pequeno	Algum	Muito grande
Finanças	Sem previsão orçamentária	Depende do escopo do projeto e porte	Há previsão orçamentária e controle de projeto
Metodologia	Informal	Parcialmente desenvolvida	Bem desenvolvida
Fonte de Dados	Entrevistas informais e pesquisa na literatura	Regular coleta de dados	Pesquisa, coleta e análise de dados
Integração	Esporadicamente ligada a outras atividades	Parcialmente integrado	Integrado com o trabalho diário

Fonte: Drew (1997).

O *benchmarking* segundo Martins e Santos (2009) também é dividido em *benchmarking* competitivo, interno, funcional e genérico, a figura 6 mostra a evolução dos tipos de benchmarking.

a) competitivo: Esta metodologia se caracteriza unicamente para procurar e aprender sobre as tecnologias e as inovações das empresas concorrentes, a vantagem deste tipo de *benchmark* é sempre a posição de frente do mercado em relação às outras empresas;
b) interno: Este tipo de benchmarking se caracteriza para o uso dentro da própria empresa, este tipo de metodologia tem como o propósito a busca de falhas dentro da empresa com a busca de melhorias do processo de trabalho, e também para um autoconhecimento da própria empresa;
c) funcional/genérico: Esta metodologia não busca a concorrência para com outras empresas, o objetivo dela é buscas métodos e processos em empresas consideradas de alto nível ou multinacionais, verificar seus processos e tentar leva-los para a própria empresa, com isso pode-se ter uma vantagem competitiva para com outras empresas.

Figura 6 – Evolução dos benchmarkings



Fonte: Zairi (1995).

5.2.4 Aplicações

A técnica de benchmarking é utilizada nas mais diversas áreas e em quase todos os tipos de segmentos do mercado, muitas empresas ainda não sabem que existe a técnica de benchmarking, mas utilizam com outros nomes e formas.

Normalmente as técnicas de *benchmarking* são utilizadas no setor de Qualidade das empresas, onde eles buscam o melhoramento dos seus produtos e a melhoria dos processos de trabalho da empresa.

Também é bem utilizada no ramo da tecnologia e nos setores de TI das organizações, para avaliar determinados sistemas, máquinas e banco de dados, neste projeto serão utilizadas as técnicas de *benchmarks* para avaliar o desempenho de três sistemas operacionais, o Linux Raspbian, o Windows 10 IoT e o Linux Ubuntu Mate.

5.2.5 Métricas de desempenho no Raspberry Pi

Neste projeto serão avaliados o desempenho de dois sistemas operacionais com uso do Raspberry Pi, o Linux Raspbian e o Windows 10 IoT. Para avaliar qual sistema que se adapta melhor ao uso do Raspberry será avaliada uma série de métricas de desempenho dentro de cada sistema operacional.

Foi avaliado o gerenciamento de processamento que irá avaliar como o Raspberry processa as informações dentro de cada sistema operacional, como o processador trabalha com cálculos matemáticos e renderização.

No desempenho de vídeo será demonstrado como ele se comporta ao rodar jogos e analisar gráficos, no desempenho dos navegadores será demonstrado como o Raspberry faz criptografia e descriptografia dos dados, processamento de arrays e de código fonte.

Para fazer estas análises serão instalados vários softwares de *benchmarks* nos dois sistemas operacionais, e através das análises destes softwares serão comparadas as métricas através de um método estatístico para então dizer qual sistema que se adapta melhor ao uso do Raspberry Pi.

5.3 MÉTODOS ESTATÍSTICOS

Um método é um conjunto de passos ou rotinas para se chegar a algum objetivo específico, a estatística é a área da matemática que a abrange a coleta, organização e processamento de dados matemáticos a fim de possibilitar um estudo matemático de algum determinado assunto (CRESPO, 1999).

A estatística abrange várias áreas de estudo, e está presente em quase todas as áreas do conhecimento. As áreas que utilizam o método estatístico estarão mais preparadas na coleta, organização e análise de dados. É utilizando também os métodos estatísticos quando é feito um estudo sobre algum determinado assunto e é preciso compreender melhor os dados, ai é utilizado os métodos estatísticos, para fica mais fácil a compreensão dos dados (BATTISTI; BATTISTI, 2008).

O método estatístico analisa todas as causas e variações de algum determinado assunto possibilitando a comparação dos dados, e analisando quais formas são mais adequadas para se utilizar ou executar o estudo feito, analisando erros e acertos e chegando a uma conclusão mais absoluta e com menor taxa de erros (CRESPO, 1999).

O método estatístico é dividido em algumas fases de elaboração. A primeira delas é o planejamento, nesta etapa é feita a preparação das outras fases, como será feita a coleta e análise dos dados e os métodos que serão utilizados durante todo o processo. A segunda etapa é a coleta dos dados, estas coletas podem ser feitas por períodos de tempo (diária, semanal, mensal), ou *softwares* que façam a coleta destes dados, a apresentação dos dados e também a análise destes dados (CRESPO, 1999).

A terceira etapa é chamada de População e Amostra, nesta etapa é definido se serão amostras qualitativas ou quantitativas. A quarta etapa são as Séries estatísticas, onde serão agrupados os dados em tabelas ou outros meios, serão demonstrados também os índices, coeficientes e taxas, todos estes dados calculados e agrupados. E a última etapa é chamada de gráficos estatísticos, nesta etapa são feitos as conclusões por meios de gráficos onde serão mostrados todos os dados processados e comparados, os gráficos podem ser de linhas, colunas, de barras ou de setores (CRESPO, 1999).

Para a execução deste projeto foram utilizando alguns métodos estatísticos para a comparação e validação dos dados. Os métodos utilizados foram os seguintes: Shapiro-Wilk, U de Mann-Whitney, Test T de Student e o teste de Levene.

O teste de Shapiro Wilk é um teste para analisar a normalidade dos dados, este teste foi desenvolvido no ano de 1965, ele é eficiente para vários tipos de tamanhos de amostras, este teste tem como objetivo analisar a probabilidade das amostras para saber se os dados pertencem a uma distribuição de dados normal ou não. Ele considera se a probabilidade for menor do que 0,05 ou p<= 0,05, a distribuição é considerada não normal, e se a distribuição for maior do que 0,05 a distribuição é considerada normal (LOPES; BRANCO; SOARES, 2013).

O teste U de Mann-Whitney, foi um teste elaborado em 1945 por Frank Wilcoxon, ele é utilizado para testar duas amostras independentes, consideradas não normais pelo teste de Shapiro-Wilk, ele é considerado um teste não paramétrico, pois a amostra analisada possui um valor muito pequeno, se a amostra tiver a distribuição normal é preciso realizar o Teste T de Student (PEREIRA, 2010).

O teste T de Student foi desenvolvido no ano de 1908 por William Sealy Gosset, este teste é um dos mais utilizados para a comparação de dados estatísticos, esse teste analisa as probabilidades das distribuições consideradas normais pelo teste de Shapiro-Wilk. Ele é um teste de hipótese que usa conceitos estatísticos para rejeitar ou não uma amostra nula (SILVA, 2014).

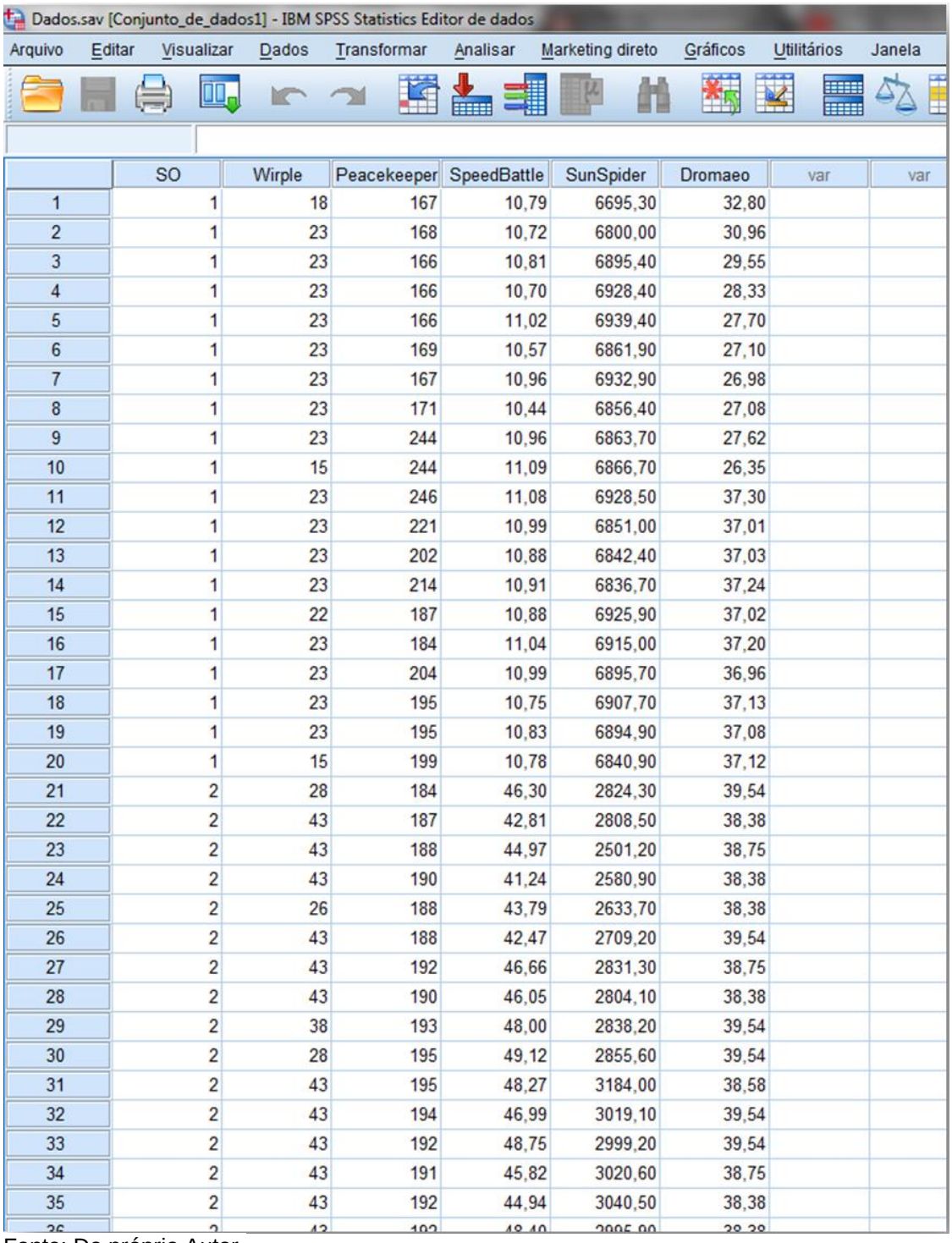
Ele é divido em alguns tipos, o primeiro tipo do teste T é o teste de hipóteses para uma amostra, é utilizado este teste quando a variância da população é desconhecida. O outro tipo é o teste de hipóteses para duas amostras independentes, este teste analisa dois grupos de variáveis numéricas que é desconhecido as suas variâncias. O terceiro tipo é o teste de hipóteses para duas amostras relacionadas, ele mede dois conjuntos de dados independentes, mas com valores semelhantes. Os testes T de Student e o teste U de Mann-Whitney analisam dados heterogêneos, como grande diferença de valores (SILVA, 2014).

O último teste é o teste de igualdade de variâncias, onde verifica se os dados são homogêneos, para verificar se os dados são homogêneos utiliza-se o teste de Levene. O teste de Levene foi desenvolvido em 1960, ele testa a igualdade de variâncias para realizar estudos balanceados de variáveis estatísticas, e utiliza

como análise um único fator, cada observação é substituída pelo desvio absoluto da variável em relação à média analisada (ALMEIDA; ELIAN; NOBRE, 2008).

A figura 7 mostra o ambiente de execução mostrando as variáveis cadastradas e os valores dos testes de *benchmark* com o software SPSS da IBM.

Figura 7 – Ambiente de testes com o SPSS



	SO	Wirple	Peacekeeper	SpeedBattle	SunSpider	Dromaeo	var	var
1	1	18	167	10,79	6695,30	32,80		
2	1	23	168	10,72	6800,00	30,96		
3	1	23	166	10,81	6895,40	29,55		
4	1	23	166	10,70	6928,40	28,33		
5	1	23	166	11,02	6939,40	27,70		
6	1	23	169	10,57	6861,90	27,10		
7	1	23	167	10,96	6932,90	26,98		
8	1	23	171	10,44	6856,40	27,08		
9	1	23	244	10,96	6863,70	27,62		
10	1	15	244	11,09	6866,70	26,35		
11	1	23	246	11,08	6928,50	37,30		
12	1	23	221	10,99	6851,00	37,01		
13	1	23	202	10,88	6842,40	37,03		
14	1	23	214	10,91	6836,70	37,24		
15	1	22	187	10,88	6925,90	37,02		
16	1	23	184	11,04	6915,00	37,20		
17	1	23	204	10,99	6895,70	36,96		
18	1	23	195	10,75	6907,70	37,13		
19	1	23	195	10,83	6894,90	37,08		
20	1	15	199	10,78	6840,90	37,12		
21	2	28	184	46,30	2824,30	39,54		
22	2	43	187	42,81	2808,50	38,38		
23	2	43	188	44,97	2501,20	38,75		
24	2	43	190	41,24	2580,90	38,38		
25	2	26	188	43,79	2633,70	38,38		
26	2	43	188	42,47	2709,20	39,54		
27	2	43	192	46,66	2831,30	38,75		
28	2	43	190	46,05	2804,10	38,38		
29	2	38	193	48,00	2838,20	39,54		
30	2	28	195	49,12	2855,60	39,54		
31	2	43	195	48,27	3184,00	38,58		
32	2	43	194	46,99	3019,10	39,54		
33	2	43	192	48,75	2999,20	39,54		
34	2	43	191	45,82	3020,60	38,75		
35	2	43	192	44,94	3040,50	38,38		

Fonte: Do próprio Autor.

6 TRABALHOS CORRELATOS

Para a pesquisa e elaboração deste projeto foram utilizados como base cientifica alguns trabalhos correlatos, estes trabalhos serão listados a seguir.

6.1 ANÁLISE DE SISTEMAS OPERACIONAIS UTILIZANDO PLATAFORMA EMBARCADA

Este artigo foi desenvolvido por Victor Gutemberg Santos Lima, Wanderson Roger Azevedo Dias, José Damião de Melo e Edwar David Moreno, no departamento de computação na Universidade Federal do Sergipe, esse artigo analisa o desempenho da plataforma de prototipação Raspberry Pi.

Este artigo faz uma análise de desempenho entre dois sistemas operacionais o Linux Raspbian e o Pidora, através de alguns *benchmarks*, e como resultado o Linux Raspbian se saiu mais adequada para uso.

Este foi o principal projeto que foi utilizado como base para este trabalho, foi utilizada como base cientifica o *hardware* utilizado, assim como os *softwares* de *benchmarking* e um dos sistemas operacionais.

6.2 ANÁLISE COMPARATIVA DE DESEMPENHO ENTRE OS CODIFICADORES DE VÍDEO H.264/AVC JM E H.264/SVC JSVM

Neste Trabalho de Conclusão de Curso desenvolvido por Lauro Aguirre Nascimento, no curso de Ciência da Computação na Universidade do Extremo Sul Catarinense, em Criciúma, Santa Catarina no ano de 2013, ele mostra o desempenho de dois tipos de *codecs* de vídeo.

Este projeto analisa alguns *benchmarks* para codificação de vídeos do padrão H.264AVC e H.264SVC. Lauro fez uso da análise de várias sequencias de vídeos para testar os dois padrões de vídeo, e com as métricas de desempenho ele pode obter qual codificador de vídeo se destacou.

Como resultado ele identificou que o padrão que tem suporte a escalabilidade é mais adequado para uso do que o codificador sem escalabilidade, vídeo destes dois *codecs*.

6.3 ANÁLISE DE DESEMPENHO DE MÁQUINAS VIRTUAIS USANDO BENCHMARK

Este projeto foi feito por Erelyn Luís Gonçalves Alves e João Paulo Ferreira dos Santos, no Curso de Ciência da Computação na Universidade da Amazônia, em Belém, no Pará no ano de 2011, ele mostra uma análise de desempenho em máquinas virtuais.

Este trabalho avalia o desempenho de dois sistemas operacionais, o Windows Server 2008 e o CentOS, como ferramenta para virtualização foi utilizado o Virtual Box e o Vmware Workstation. Ele avaliou o desempenho através dos gráficos gerados pelos *benchmarks* e como resultado o CentOS ficou mais adequado.

Foi utilizado como base cientifica neste projeto, o software de benchmark utilizado por eles e os tipos de análises de desempenho que eles utilizaram.

6.4 ANALYSIS OF VIRTUALIZATION TECHNOLOGIES FOR HIGH PERFORMANCE COMPUTING ENVIRONMENTS

Este artigo foi desenvolvido Andrew J. Younge, Robert Henschel, James T. Brown, Gregor von Laszewski, Judy Qiu, Geoffrey C. Fox, na Universidade de Indiana, em Bloomington, nos Estados Unidos, e retrata análises para virtualizações de máquinas que possuem alto desempenho.

O artigo realiza uma séria de testes em máquinas virtuais, realizando testes com o *HPCC benchmarking* que é um conjunto de *benchmarks* que fazem testes em servidores, no resultado deste projeto o servidor da KVM saiu na frente.

Neste artigo foi utilizado como base cientifica os tipos de *benchmarks* utilizados por eles para fazerem os testes de desempenho.

7 TRABALHO DESENVOLVIDO

Inicialmente foi realizado o levantamento bibliográfico sobre o Raspberry e os sistemas operacionais utilizados, após isso foi realizado o desenvolvimento da fundamentação teórica do projeto e após isso a realização da parte prática.

7.1 METODOLOGIA

A primeira parte prática realizada foi no Linux Raspbian, foi feita toda a sua instalação e realizada a instalação dos programas de *benchmark*, após isso feitos os testes de desempenho.

Depois foi feita a metodologia do Windows 10 IoT, onde foi realizada toda a instalação do sistema operacional e a instalação do navegador e após isso a execução dos testes de desempenho.

Todos os testes de desempenho realizados no Linux Raspbian e no Windows 10 IoT foram executados vinte vezes cada teste para se ter uma base sólida de testes para os resultados.

7.1 LINUX RASPBIAN

O primeiro sistema operacional que foi avaliado o desempenho foi o Linux Raspbian, para instalar o Linux primeiro é preciso baixar a imagem de seu sistema no site do Raspberry e ir à aba *downloads*, depois de acessar é preciso ir à opção *Linux Raspbian*, depois disso terá duas opções do sistema para baixar, a primeira que é a versão completa que se chama Linux Raspbian Jessie e a versão Linux Raspbian Jessie Lite, esta versão Lite é mais reduzida e possui apenas interface terminal por linhas de comando.

Selecionar a primeira versão que é a completa, e tem a opção de baixar por *torrent* usando algum aplicativo que aceite a extensão *torrent* ou pelo navegador que baixará a versão compactada, no caso deste projeto foi baixada a versão pelo navegador, depois de baixada, descompactar a imagem do sistema utilizando algum descompactador de arquivos (*Winrar,Winzip*), após descompactar surgirá a imagem do sistema no formato ISO, após isso é só gravar a imagem no cartão de memória.

Para gravar a imagem do Raspbian no cartão de memória do Raspberry é preciso baixar um *software* gratuito da Internet que a própria empresa do Raspberry solicita aos usuários que eles utilizem para gravar a imagem de seu sistema operacional, o nome do programa é *Win32DiskImager*, este programa é para gravar a imagem do Linux Raspbian para o cartão de memória, este programa é para usuários que utilizam Windows, como foi o caso deste projeto. Para outros sistemas operacionais, o processo de gravação da imagem do Raspbian é feito de maneira diferente.

Depois de baixar o programa Win32DiskImager, colocamos o cartão de memória no drive de cartão de memória do computador ou no slot de cartão de memória do notebook utilizando o seu adaptador que vem junto com o cartão. Com o programa *Win32DiskImager* aberto, ele seleciona automaticamente a unidade do computador que está inserido o cartão de memória, lembrando que o cartão de memória precisa estar formatado, como no caso deste projeto o cartão era novo, ele já estava formatado, depois é preciso selecionar onde está a imagem do Raspbian que foi feito *download*, depois é só ir ao programa e clicar em *Write*, que começará a gravação da imagem do Raspbian no cartão de memória.

Após o programa completar a gravação da imagem do Raspberry, é preciso ejetar o cartão do computador e encaixa-lo no compartimento abaixo do Raspberry, é a única entrada disponível abaixo dele, depois de encaixa-lo é preciso ligar o Raspberry na tomada ou no estabilizador lembrando que ele é *bivolt* (110~220V), e o sistema operacional dele irá dar *boot* sozinho.

Para a execução de todos os testes deste projeto foi utilizado um Raspberry Pi 2 Modelo B com 1GB de memória RAM, com um processador Broadcom BCM2836 de 900Mhz e possui uma GPU Broadcom VideoCore IV. Com o Linux Raspbian em execução, é preciso ir até na barra de tarefas do Linux Raspbian e ir ao aplicativo *Internet*, onde serão iniciados primeiramente os testes dos *benchmarkings online*.

O primeiro *benchmarking* testado foi o Wirple que é um benchmarking que realiza stress no processador gráfico e testes 3D. Na barra de tarefas do navegador digitar o link disponível em[1], na próxima tela abrirá uma tela onde pedirá para iniciar

[1] Disponível em: http://www.wirple.com/bmark/

o teste, clicar em *Start Test*, após isso este *benchmark* começará a fazer os testes, ele roda 4 testes.

No primeiro teste é feito um stress no processador gráfico onde mostram na tela vários blocos coloridos, e verifica o desempenho que o Raspberry leva para processar os blocos, no segundo teste é mostrado na tela duas esferas apenas com suas armações e contornos, esse teste mede a média do desempenho dos quadrados de cada esfera.

No terceiro teste, aparece na tela três globos e vários pixels coloridos lembrando focos de luz, esse teste mede o desempenho das partículas de luz, sombras e opacidade. No último teste é parecido com o terceiro, mas além de medir as partículas de luz, sombras e opacidade ele mede também reflexos e realismos as imagens mostradas na tela. No final dos testes o *benchmark* fornece uma pontuação total dos quatro testes executados.

O segundo *benchmarking* online testado foi o *Peacekeeper Futuremark*, para acessa-lo basta digitar o link que está disponível em[2], o *Peacekeeper* é um *benchmark Java Script*, onde ele roda uma série de testes, primeiramente ele roda alguns testes de renderização, ou seja, no primeiro teste ele roda na tela alguns blocos de cor amarela e mede como o navegador movimenta o fluxo dos blocos em vários cantos da tela, no segundo teste aparece na tela alguns desenhos de dentes-de-leão caindo em um gramado, esse teste mede a renderização física dos objetos em movimento na tela além de cálculos matemáticos.

O terceiro teste avalia se o navegador suporta *HTML5 WebGL*, ou seja, se o navegador suporta recursos 3D sem a intervenção de *plug-ins* externos, neste teste o navegador roda um cubo 3D transparente na tela sendo movimentado em vários ângulos diferentes, avaliando principalmente os efeitos de luz e sombra além do processamento gráfico no Raspberry. O quarto teste avalia o uso dos diferentes formatos de vídeos suportados pelo navegador, o quinto teste avalia a quantidade de coisas que o navegador consegue rodar ao mesmo tempo, ou seja, ele estressa o navegador ao rodar vários cálculos e imagens na tela.

O sexto teste ele roda um jogo 2D de aviões, onde ele avalia o desempenho do Raspberry ao rodar jogos além de simulações no navegador envolvendo o movimento dos aviões. O sétimo teste é o teste *Canvas,* ou seja, ela é

[2] Disponível em: http://peacekeeper.futuremark.com/run.action

uma tecnologia usada na *Web* para desenhar e manipular elementos gráficos sem a ajuda de *plug-ins*, neste teste aparece à imagem de uma cachoeira, onde ele mede a capacidade do navegador ao rodar pixels individuais.

No oitavo teste, ele faz uma análise através de matrizes de *Java Script* em uma página dinâmica, este teste analisa a capacidade de o navegador adicionar, modificar e remover dados armazenados em uma matriz. O teste nove é chamado de operações no modelo de objeto do documento, isso significa que ele pode emular métodos para criar páginas *Web* dinâmicos, esse teste nove é baseado no *framework JQuery*. O último teste faz a análise e simulação do uso de textos no navegador, como por exemplo, a utilização de filtros de palavras e validação de formulários, além de testes em loop de palavras. No final do teste este *benchmark* disponibiliza uma pontuação total de todos os testes executados.

O terceiro *benchmark* testado foi o *Speed-Battle,* para acessa-lo basta digitar o *link* disponível em[3], este teste faz uma análise geral em todo o *hardware* e *software* do Raspberry, tais como recursos de CPU, navegador, *plug-ins*, memória e processos do sistema rodando, este é um *benchmark* que não trás muitas informações a respeito dele, no final ele também fornece uma média da pontuação dos testes executados.

O quarto *benchmark online* testado foi o *Sun Spider JavaScript Benchmark*, para acessa-lo basta digitar o *link* no navegador que está disponível em[4] este teste *Java Script* analisa uma série de requisitos tais como criptografia e descriptografia de dados, descompressão de código fonte gerado pelo navegador, árvores binárias, processamento de strings, controle de fluxo de dados além de cálculos matemáticos.

O quinto e último *benchmarking online* testado foi o *Dromaeo: JavaScript Performance Testing*, para acessa-lo basta digitar o link disponível em[5], dos benchmarkings online testados este foi o que apresentou a maior quantidade de tempo para rodar os testes e o que apresentou informações mais detalhadas.

Na página do *benchmark* ir até a opção *Run Recommended Tests,* este teste analisa muitas variáveis tais como renderização 3D, avaliação de código, desempenho de arrays, *plug-ins* do Java como, por exemplo, *JQuery*, cálculos e

[3] Disponível em: http://www.speed-battle.com/speedtest_e.php
[4] Disponível em: https://webkit.org/perf/sunspider-1.0.2/sunspider-1.0.2/driver.html
[5] Disponível em: http://dromaeo.com/

expressões matemáticas, desempenho de *strings, hashing*, além de outros testes já citados como por exemplo criptografia e descriptografia de dados e árvores binárias.

Depois de testados os *benchmarkings online* foram instalados alguns *benchmarkings desktop* no Linux Raspbian, o primeiro *benchmark* instalado foi o software *GTKPerf*, para instala-lo é necessário abrir o *Terminal* do Linux e entrar como administrador pelo comando *sudo su* e digitar o comando *sudo apt-get-install gtkperf*, após isso o sistema operacional instalará o *software*, para acessa-lo basta ir no menu principal dele e ir em ferramentas do sistema e selecionar o programa ou ir pelo terminal e digitar *gtkperf*. Com o programa aberto é só ir na opção *"Run"*, ai o *software* rodará todo o processo.

Este teste é basicamente uma avaliação de processamento gráfico, ou seja, ele analisa uma sequencia de exames comuns como por exemplos abrir e fechar caixas de texto do sistema, abrir e fechar caixas de diálogo, cliques do mouse, desenha várias imagens e traços na tela em diversas cores e formatos diferentes, de modo a avaliar as capacidades gráficas do Raspberry.

O segundo *benchmarking* desktop instalado foi o *Hard Info*, este *benchmarking* engloba seis outros *benchmarks*, a maioria deles baseados em algoritmos complexos, para instala-lo é necessário abrir o terminal do Linux e entrar como administrador pelo comando *sudo su* e digitar o comando *sudo apt-get-install hardinfo*, após isso o sistema operacional instalará o *software*, para acessa-lo basta ir ao menu principal dele e ir em ferramentas do sistema e selecionar o programa ou ir pelo terminal e digitar *hardinfo,* ao abrir o software ele trará uma série de informações sobre todos os componentes do Raspberry, seu tipo de processador, interfaces de rede, placa gráfica, periféricos conectados, memória, disco e informações sobre o Linux Raspbian instalado.

Na parte inferior do programa, tem uma aba chamada *Benchmarks*, ao clicar nesta opção abrirão seis outros benchmarks mais destinados a testes do processador. O primeiro teste é o *CPU Blowfish*, este teste tem como base o algoritmo de *Blowfish*, ou seja, ele analisa o processador separando as informações em blocos de 32 bits, ele funciona da seguinte maneira, primeiro ele expande as chaves e as sub-chaves analisadas e depois faz a encriptação destes dados.

O segundo teste é o *CPU CryptoHash*, este teste analisa a integridade e a validação de dados que o processador executa, o terceiro teste é o *CPU Fibonacci*,

este teste analisa o tempo que o processador leva para executar a sequencia de *Fibonacci* até o 42º elemento. O quarto teste é o *CPU N-Queens*, este teste baseia-se no algoritmo complexo do N-Rainhas conhecido também como algoritmo das oito rainhas, ou seja, utiliza o cálculo exponencial combinatório para calcular a capacidade de utilização do processador.

O quinto teste é o *FPU-FFT*, este teste analisa cálculos matemáticos do processador utilizando como bases pontos flutuantes, e o último teste é o *FPU-Raytracing*, este teste sintetiza e analisa imagens tridimensionais executados pelo processador.

7.1 WINDOWS 10 IOT

A segunda parte da metodologia foi feita em cima do Windows 10 IoT, para instalar o Windows 10 IoT é necessário, um *notebook* ou *desktop* que tenha Windows 10. No computador com Windows 10 é preciso ir ao aplicativo *Loja* e instalar o aplicativo *Windows 10 IoT Core Dashboard*, para instalar este aplicativo é preciso entrar com uma conta da Microsoft, depois que fizer isso é só procurar no aplicativo *Loja* e instalar. Nesse momento pode-se colocar o cartão de memória no computador.

Depois de instalado abrir o programa e ir ao menu *Configurar um novo dispositivo*, será preciso colocar o *Tipo de dispositivo*, no caso deste projeto foi o Raspberry Pi 2, na *Compilação do SO* colocar Windows 10 IoT Core, na *Unidade* aparecerá automaticamente o cartão de memória inserido no computador, se não aparecer automaticamente, selecionar a unidade onde ele está alocado no computador. No nome do dispositivo foi deixado como padrão o nome *minwinpc*, mas pode-se colocar o nome da preferência do usuário.

Na opção *New Administrator password* não foi colocada senha, mas é opcional a colocação de senha de administrador do sistema, após isso é preciso aceitar os termos de licença da Microsoft e ir à opção *Baixar e Instalar*, ai irá demorar algumas horas para baixar e instalar o Windows 10 IoT no Raspberry, ele baixará e instalará direto no cartão de memória, após o programa terminar, retirar o cartão de memória do computador e colocar no Raspberry, após ainda alguns minutos atualizando o sistema operacional, o Windows 10 IoT se iniciará sozinho.

É possível acessar o Windows 10 IoT de várias maneiras, a primeira é pela interface gráfica dele, utilizando monitor com HDMI. A segunda é pelo navegador em alguma máquina que tenha Windows 10 e que possua o *Windows 10 IoT Core Dashboard.* Com o Raspberry ligado com o Windows 10 IoT, abrir o *Windows 10 IoT Core Dashboard* e esperar a placa carregar na tela do programa. Após isso clicar com o botão direito do mouse sobre ele e selecionar a opção *Abrir no Device Portal*, ele abrirá o navegador e pedirá usuário a senha.

O usuário padrão é *Administrator* e a senha é *raspberry*, aqui é possível acessar todas as configurações do Windows 10 IoT, instalar programas e acompanhar processos e performance do sistema. A terceira forma de acessar ele, é por outro aplicativo da *Loja* do Windows, chamado *Windows IoT Remote Client*, onde é possível acessar a tela dele sem precisar de monitor com HDMI.

Alguns problemas foram encontrados em relação ao Windows 10 IoT, primeiramente ele não possuía nenhum navegador instalado, para resolver este problema, foi encontrado um código fonte de navegador para Raspberry Pi no site da Microsoft chamado de *IoT Brownser Sample*, o código fonte era desenvolvido em C#, feito o *download* do código é preciso baixar e instalar o *Microsoft Visual Studio*, para poder funcionar o navegador, foi realizado o *download* da versão 2015 também disponibilizado no site da Microsoft. Depois de instalar o *Microsoft Visual Studio* é preciso baixar e instalar o pacote de atualização para trabalhar com o Raspberry Pi, chamado *Windows 10 Software Development Kit* (SDK), encontrado também no site da Microsoft.

Depois de instalar isso, é necessário abrir o código fonte do navegador e compilar remotamente para a plataforma, na opção *Remote Machine*. Para realizar isso o Raspberry precisa estar ligado e configurado corretamente, após isso ele irá compilar automaticamente lá na placa, se não será preciso colocar o IP e configurar ele manualmente.

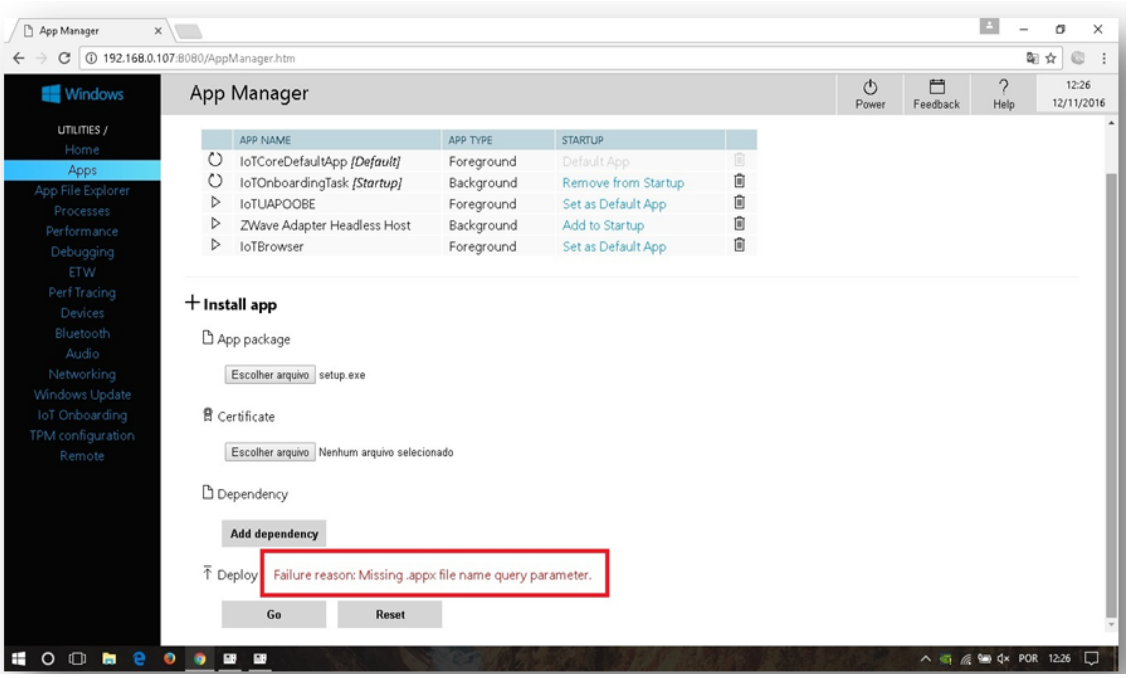
Foi utilizado o mesmo procedimento do Linux Raspbian no Windows 10 IoT em relação aos *benchmarkings online*, foram executados os testes dos *benchmarks* Wirple, Peacekeeper, SpeedBattle, SunSpider e Dromaeo, todos eles compilados remotamente através do navegador implementado em C#.

Outro problema que aconteceu foi que o Windows 10 IoT, não aceita nenhum programa *desktop* igual a qualquer Windows, ou seja ele não aceita

nenhum programa no formato *Exe* ou *Msi*, vale informar que o Windows 10 IoT não vem com nenhum aplicativo instalado, é uma plataforma sem nenhum aplicativo instalado. Ele roda apenas programas no formato *Appx*, como mostrado na figura 8. O *Appx* é o novo formato que a Microsoft utiliza para aplicativos do Windows Phone, a Microsoft chama esses aplicativos, de aplicativos universais, pois eles rodam em Windows Phones, Raspberrys, e no próprio Windows 10 para *desktops* e *notebooks*, mas como os *benchmarks* encontrados são antigos e não recentes como o Windows 10 IoT, não foi possível fazer a instalação deles no Raspberry, então foram apenas utilizados apenas os *benchmarks online* no Windows 10 IoT, os *desktops* não foram.

Foram encontrados vários *benchmarks* para utilizar no Windows, foram encontrados até mais do que no Linux, como por exemplo: *PcMark, CPU-M Benchmark, 3D Mark, Disk Benchmark, NovaBench, SisSoftware Sandra, HD Tune, FurMark* e *AIDA 64*. Não foi possível instala-los no Windows 10 IoT pois estes *benchmark*s eram no formato *Exe* e não possuíam versão em *Appx*, por isso foram testados apenas os *benchmarks online* no Windows 10 IoT.

Figura 8 – Windows 10 IoT – APPX



Fonte: Do próprio Autor

7.4 MÉTODO ESTATÍSTICO

Todos os resultados dos testes de *benchmarking* foram armazenados em uma planilha do Microsoft Office Excel. Para realizar a análise, processamento e comparação dos dados foi utilizado o *software Statistical Package for Social Sciences* (SPSS) versão 22 da empresa IBM, ele tem a versão paga e a versão gratuita, para este projeto foi utilizada a versão gratuita.

Este *software* está disponível para *download* no site da IBM, depois de feito o *download*, é preciso executar o *setup* e instalar o programa, antes de terminar de instalar o programa pedirá para ativar ou utilizar a versão gratuita, foi utilizado a versão gratuita. Depois é necessário abrir o programa, então exibirá uma janela onde é preciso colocar o arquivo temporário para a ativação gratuita do programa, o programa irá selecionar a pasta automaticamente, será apenas preciso selecionar o arquivo. Após isso é só reiniciar o computador e o SPSS estará pronto para uso.

Com o SPSS aberto, primeiramente é preciso cadastrar os sistemas operacionais e os *benchmarks*, ir à opção visualização da variável, e ir ao primeiro campo da tabela e digitar o nome do *benchmark*, após dar *enter* no teclado os outros campos vão se completar automaticamente, é preciso arrumar o campo casas decimais, rótulo e o campo medir, no campo medir para variáveis numéricas colocar a opção de escala e para variáveis palavras a opção nominal. Após preencher os *benchmarks* foi pego os dados da planilha e copiados para o SPSS, foram definidos como sistema operacional 1 o Linux e o sistema operacional 2 o Windows.

O primeiro teste que irá ser realizado vai nos trazer as informações de média, desvio padrão e o teste de Shapiro-Wilk. Para fazer isso, selecionar o menu *Analisar*, depois *Estatísticas Descritivas* e *Explorar*. Na próxima tela, na opção de *Lista Dependente*, colocar o primeiro *benchmark*, na *Lista de Fatores* é preciso colocar os sistemas operacionais cadastrados na lista de variáveis. Ainda nessa mesma tela ir à opção *Gráficos*, desmarcar a opção *Caule e Folha* e marcar a opção *Gráficos de Normalidades com Testes*, depois ir à opção *Continuar* e depois em *OK,* ai irá nos trazer três tabelas e quatro gráficos com todas as informações detalhadas do teste, neste projeto foi aproveitada a média, o desvio padrão e o resultado do teste de Shapiro-Wilk.

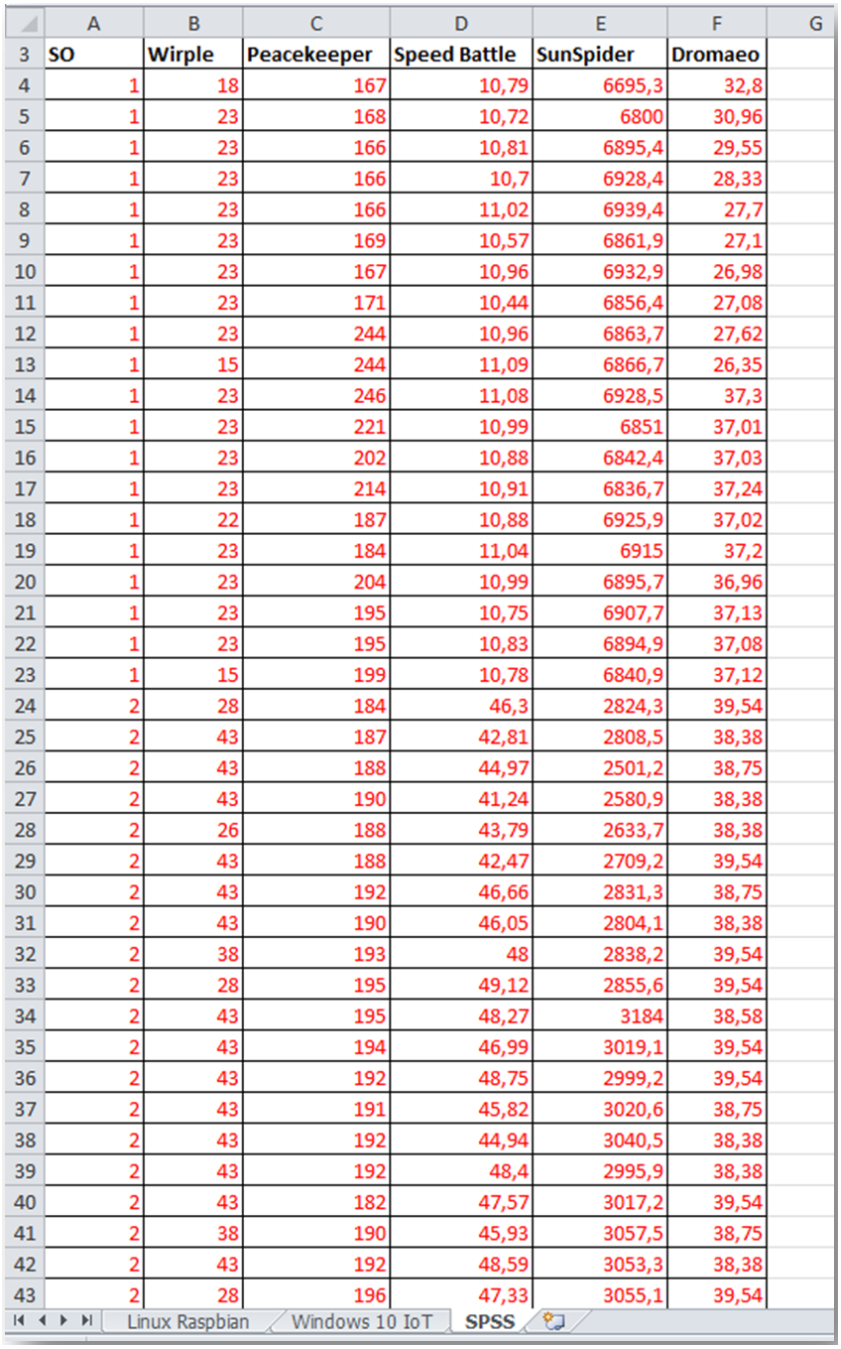
Depois é preciso fazer isso com os outros quatro *benchmarks online*. Os *benchmarks* que apresentaram a distribuição não normal foi preciso fazer o teste de U de Mann-Whitney. Para fazer este teste é preciso ir no menu *Analisar*, depois em *Testes Não-Paramétricos*, depois *Caixa de Diálogos Legadas*, e depois selecionar a opção 2 amostras independentes, e deixar marcada a caixa do teste de Mann-Whitney.

Para fazer os testes de distribuição normal foram realizados o Teste T de Student e o teste de Levene. Para utilizar o Teste T é preciso ir no menu *Analisar*, depois ir em *Comparar Médias*, e ir na opção *Teste T de Amostras Independentes*. Neste projeto foi utilizado o resultado da coluna *Sig. (2 extremidades),* se os valores calculados forem parecidos é preciso ir na coluna do Teste de Levene e verificar os valores se possuem variâncias iguais ou não.

8 RESULTADOS

Todos os testes de *benchmarking* realizados foram passados para uma planilha de Excel, onde mostra a pontuação dos *benchmarkings* e o tempo de realização para cada teste, as pontuações dos *benchmarks online* são mostrados na imagem 9. O resto dos testes detalhados está no apêndice final deste projeto.

Figuras 9 – Pontuação dos benchmarkings online



	A	B	C	D	E	F	G
3	SO	Wirple	Peacekeeper	Speed Battle	SunSpider	Dromaeo	
4	1	18	167	10,79	6695,3	32,8	
5	1	23	168	10,72	6800	30,96	
6	1	23	166	10,81	6895,4	29,55	
7	1	23	166	10,7	6928,4	28,33	
8	1	23	166	11,02	6939,4	27,7	
9	1	23	169	10,57	6861,9	27,1	
10	1	23	167	10,96	6932,9	26,98	
11	1	23	171	10,44	6856,4	27,08	
12	1	23	244	10,96	6863,7	27,62	
13	1	15	244	11,09	6866,7	26,35	
14	1	23	246	11,08	6928,5	37,3	
15	1	23	221	10,99	6851	37,01	
16	1	23	202	10,88	6842,4	37,03	
17	1	23	214	10,91	6836,7	37,24	
18	1	22	187	10,88	6925,9	37,02	
19	1	23	184	11,04	6915	37,2	
20	1	23	204	10,99	6895,7	36,96	
21	1	23	195	10,75	6907,7	37,13	
22	1	23	195	10,83	6894,9	37,08	
23	1	15	199	10,78	6840,9	37,12	
24	2	28	184	46,3	2824,3	39,54	
25	2	43	187	42,81	2808,5	38,38	
26	2	43	188	44,97	2501,2	38,75	
27	2	43	190	41,24	2580,9	38,38	
28	2	26	188	43,79	2633,7	38,38	
29	2	43	188	42,47	2709,2	39,54	
30	2	43	192	46,66	2831,3	38,75	
31	2	43	190	46,05	2804,1	38,38	
32	2	38	193	48	2838,2	39,54	
33	2	28	195	49,12	2855,6	39,54	
34	2	43	195	48,27	3184	38,58	
35	2	43	194	46,99	3019,1	39,54	
36	2	43	192	48,75	2999,2	39,54	
37	2	43	191	45,82	3020,6	38,75	
38	2	43	192	44,94	3040,5	38,38	
39	2	43	192	48,4	2995,9	38,38	
40	2	43	182	47,57	3017,2	39,54	
41	2	38	190	45,93	3057,5	38,75	
42	2	43	192	48,59	3053,3	38,38	
43	2	28	196	47,33	3055,1	39,54	

Linux Raspbian | Windows 10 IoT | SPSS

Fonte: Do próprio Autor.

Para fazer às médias, os desvios padrões, a análise e o processamento dos testes foi utilizado o *software SPSS*, todos os testes detalhados do SPSS estão no apêndice deste projeto. Com as médias, o desvio padrão e os valores dos testes de Shapiro-Wilk, U de Mann-Whitney, Test T de Student e o teste de Levene, foi passado às informações para o Excel e foi montada a seguinte tabela, como mostra a imagem 10.

Figura 10 – Média dos *benchmarkings online*

SPSS

Benchmarks Online	Linux Raspian	Windows 10 IoT	Distribuição	Valor p
Wirple	21,90 +- 2,61	39,40 +- 6,30	Não normal	< 0,001
Peacekeeper	193,75 +- 27,92	190,55 +- 3,60	Normal	0,617
SpeedBattle	10,85 +- 0,16	46,20 +- 2,25	Normal	< 0,001
SunSpider	6873,94 +- 57,39	2891,47 +- 182,78	Normal	< 0,001
Dromaeo	32,74 +- 4,66	38,92 +- 0,53	Não normal	< 0,001
	Valores: Média e Desvio Padrão	Valores: Média e Desvio Padrão		Valor p= probabilidade

Médias Linux Raspbian	
Benchmark	Linux Raspbian
Wirple	21,90
Peacekeeper	193,75
SpeedBattle	10,85
SunSpider	6873,94
Dromaeo	32,77

Médias Windows 10 IoT	
Benchmark	Windows 10 IoT
Wirple	39,40
Peacekeeper	190,55
SpeedBattle	46,20
SunSpider	2891,47
Dromaeo	38,92

Fonte: Do próprio Autor.

O Windows 10 IoT não suportava os *benchmarks desktop*, por causa da incompatibilidade de extensões de arquivos, foi realizado os testes dos programas *desktop* apenas no Linux Raspbian. Se no Windows 10 IoT funcionasse, poderia ser feito testes mais detalhados, e ainda comparados os *benchmarks online* com os *benchmarks desktop*.

Os testes dos *benchmarks desktop* estão no apêndice deste projeto, os testes dos *benchmarks desktop* do Linux Raspbian não foram processados no SPPS, pois não era possível comparar os dados com os *benchmarks* do Windows 10 IoT.

Após a análise das médias dos *benchmarks online*, foi montado o gráfico que trás as seguintes informações, como mostra a imagem 11.

Figura 11 – Gráfico com os resultados de cada *benchmarking online*

Resultado Final:

Benchmark	Sistema Operacional Adequado
Wirple	Windows 10 IoT
Peacekeeper	Equivalentes
SpeedBattle	Windows 10 IoT
SunSpider	Linux Raspbian
Dromaeo	Windows 10 IoT

Fonte: Do próprio Autor.

De acordo com a estatística, as médias dos *benchmarks* Wirple, SpeedBattle e Dromaeo, se adaptam melhor ao sistema operacional Windows 10 IoT utilizando o Raspberry Pi. Enquanto que o *benchmark* SunSpider se adapta melhor ao Linux Raspbian e o *benchmark* Peacekeeper por ter resultados com médias semelhantes nos dois sistemas operacionais, ele é adequado para se usar nos dois sistemas operacionais.

Então dos *benchmarks online* o sistema operacional que se adapta melhor ao uso do Raspberry Pi 2 Modelo B é o Windows 10 IoT.

9 CONCLUSÃO

A comparação de desempenho entre sistemas operacionais através de *benchmarkings* é um processo demorado, mas permite-se analisar como o sistema operacional se comporta ao processar as informações e ao receber testes de stress em seus componentes.

Neste projeto encontrou-se certa dificuldade em encontrar *benchmarks* apropriados e com testes mais detalhados em ambos os sistemas operacionais, isso porque os *benchmarks* utilizados neste projeto eram gratuitos, e os *benchmarks* considerados melhores eram pagos e licenciados.

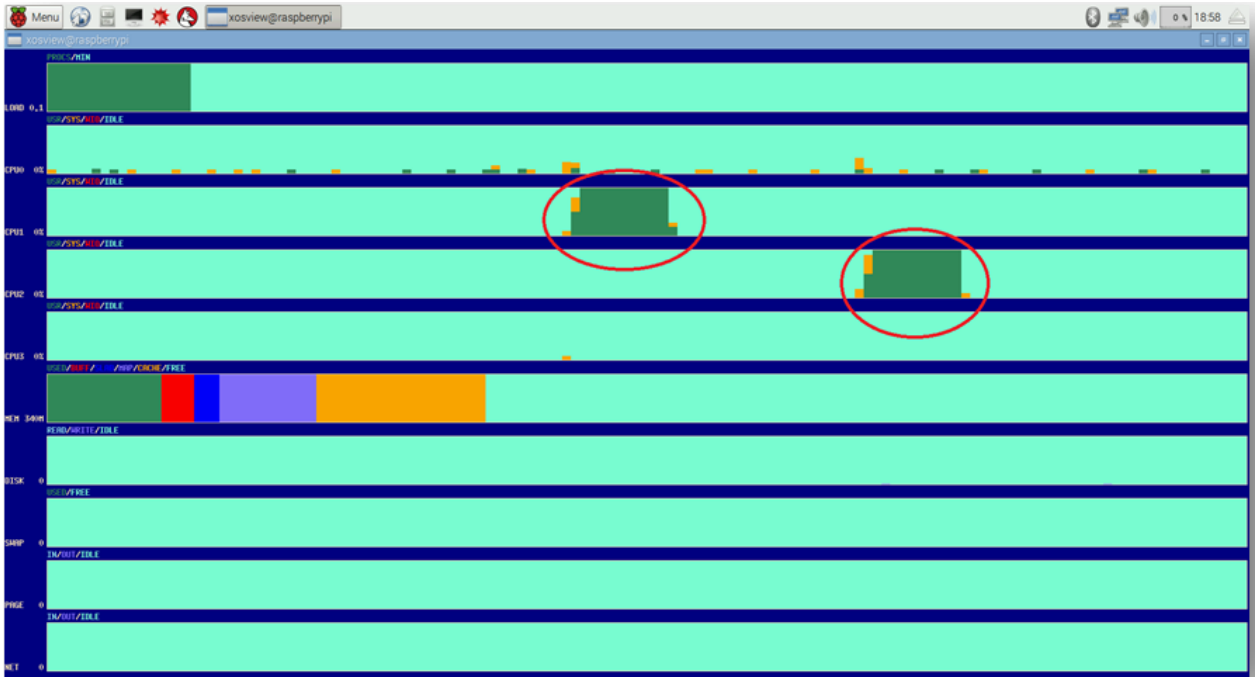
Neste projeto o Windows 10 IoT se adaptou melhor aos *benchmarks online* do que o Linux Raspbian que é o sistema nativo do Raspberry Pi, apesar de que os *benchmarks online* do Windows 10 IoT foram compilados remotamente através de um navegador remoto, e não com o navegador nativo do Windows, pois nem mesmo tinha instado no sistema operacional.

Durante todo o projeto de testes de *benchmarking* foi percebido alguns problemas em ambos os sistemas operacionais, tais como aquecimento do Raspberry Pi, foi notado que a placa esquentava em alguns momentos. Para solucionar isso foi acoplado nele um dissipador de calor, mas mesmo assim a placa ainda chegava a esquentar algumas vezes.

Em alguns momentos ocorriam alguns travamentos no sistema operacional, mas não foi encontrada a causa real desse problema, algumas causas possíveis para este problema seria o aquecimento mencionado anteriormente.

Na imagem 12 mostra um *benchmark* que foi instalado no Linux que acompanha o sistema operacional em tempo real, este *benchmark* chamado de Xosview, não faz análise de nenhum componente do sistema operacional apenas mostra o desempenho do sistema em tempo real, esta imagem mostra o sistema sem rodar nenhum aplicativo apenas com o sistema iniciado e de vez em quando aconteciam alguns picos de processamento, ou seja, seu processador oscilava sem rodar nada no sistema operacional. Isto poderia acontecer em decorrência de alguma incompatibilidade na arquitetura ARM de seu processador.

Figura 12 – *Benchmark* Xosview



Fonte: Do próprio Autor.

Na análise de desempenho de qualquer tipo de sistema operacional ou *hardware* é muito importante para descobrirem-se falhas em suas estruturas e definirem-se os pontos que devem ser melhorados dentro de cada sistema ou partes específicas de cada *hardware*. Neste projeto foi pensado que o sistema operacional que ia se adaptar melhor ao Raspberry seria seu sistema nativo Linux Raspbian, mas de acordo com os resultados obtidos o Windows 10 IoT foi mais adequado. Foi detectado que o Rapsberry sofria alguns travamentos em ambos os sistemas operacionais, mas não foi identificado o real motivo de isso acontecer.

Com base no conhecimento científico deste projeto, os trabalhos futuros seriam para procurar porque acontece essa instabilidade no processamento do Raspberry Pi, ou testar com o Raspberry Pi 3, lançado no ano de 2016, porque essa instabilidade pode ser apenas no modelo 2. Também se podem fazer testes de benchmarking com outros sistemas operacionais tais como o Linux Ubuntu Mate, Ubuntu Core, Pinet, Noobs, ou OSMC. Estes são outros sistemas operacionais que rodam no Raspberry Pi.

REFERÊNCIAS

ALENCAR, Marcelo Sampaio de. **A História do Linux.** 2005. 10 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Campina Grande, Paraíba, 2005.

ALMEIDA, Antônia de; ELIAN, Silvia; NOBRE, Juvêncio. Modificações e alternativas aos testes de Levene e de Brown e Forsythe para igualdade de variâncias e médias. **Revista Colombiana de Estadística,** Colombia, v. 31, n. 2, p.241-260, dez. 2008. Mensal.

BATTISTI, Iara Denise Endruweit; BATTISTI, Gerson. **MÉTODOS ESTATÍSTICOS.** Ijuí: Unijuí, 2008. 80 p.

BEAGLEBOARD. **BeagleBone.** 2016. Disponível em: <http://beagleboard.org/bone-original>. Acesso em: 08 abr. 2016.

BRITO, Edivaldo. **Testamos o Windows 10 Technical Preview; veja as primeiras impressões.** 2014. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2014/10/testamos-o-windows-10-technical-preview-veja-primeiras-impressoes.html>. Acesso em: 22 nov. 2015.

CAMPOS, Augusto C. **Introdução ao Linux.** São Paulo: Br-linux, 2003. 42 p.

CARDOSO, Antônio. **O paradigma do software livre – Linux Manual.** 2000. 72 f. Monografia (Especialização) - Curso de Ciencia da Computação, Universidade de São Paulo, São Paulo, 2000. Cap. 2.

CRESPO, Antônio Arnot. **Estatística Fácil.** São Paulo: Saraiva, 1999. 115 p.

DEBIAN (United States). **Sobre o Debian.** 2016. Disponível em: <https://www.debian.org/>. Acesso em: 08 maio 2016.

DREW, S. **From Knowledge to Action: The Impact of Benchmarking an Organization** Performance Long Range Planning, v. 30, n 3, pp 427, 441,1997.

ELIAS, Diego (Ed.). **A diferença entre métrica e indicador.** 2016. Disponível em: <http://www.binapratica.com.br/#!metrica-x-indicador/c240l>. Acesso em: 12 jun. 2016.

GAY, Warren. **Mastering the Raspberry Pi.** Berlin: Technology In Action, 2014. 482 p.

GOMES, Luiz Eduardo de Mello. **BENCHMARKING E APRENDIZAGEM ORGANIZACIONAL Estudo de Caso na Companhia de Processamento de Dados de Minas Gerais - Prodemge.** 2001. 125 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2001.

GOMES, Pedro Henrique; LEITE, Tatiane Silvia; CAETANO, Uirauna Imirim. **A Arquitetura ARM:** Projeto de Sistemas Computacionais. São Paulo: Campus, 2012. 8 p.

GORNI, Antônio Augusto. **INTRODUÇÃO À PROTOTIPAGEM RÁPIDA E SEUS PROCESSOS.** 2001. Disponível em: <http://www.gorni.eng.br/protrap.html>. Acesso em: 11 abr. 2016.

GUARIZZO, Karina. **MÉTRICAS DE SOFTWARE.** 2008. 48 f. Monografia (Especialização) - Curso de Ciência da Computação, Faculdade de Jaguariúna, Jaguariúna, 2008.

GUPTA, M. Surya Deekshith; PATCHAVA, Vamsikrishna; MENEZES, V Irginia. Healthcare based on loT using Raspberry Pi. In: INTERNATIONAL CONFERENCE ON GREEN COMPUTING AND INTERNET OF THINGS (ICGCLOT), 1., 2015, India. **Healthcare based on loT using Raspberry Pi.** India: Isbn, 2015. p. 796 - 799. Disponível em: <5-http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7380571>. Acesso em: 10 abr. 2016.

HORAN, Brendan. **Practical Raspberry Pi.** Berlin: Technology In Action, 2013. 261 p.

IBM (Estados Unidos). Ibm Knowledge Center (Org.). **Métricas de Desempenho.** 2016. Disponível em: <http://www.ibm.com/support/knowledgecenter/pt-br/SSEP7J_10.2.2/com.ibm.swg.ba.cognos.crn_arch.10.2.2.doc/c_performancemetrics.html>. Acesso em: 12 jun. 2016.

LACOMBE, Francisco; HEILBORN, Gilberto. **Administração:** Princípios e Tendências.São Paulo: Saraiva, 2003.

LOPES, Manuela de Mesquita; BRANCO, Verônica T. F. Castelo; SOARES, Jorge Barbosa. Utilização dos testes estatísticos de Kolmogorov-Smirnov e Shapiro-Wilk para verificação da normalidade para materiais de pavimentação. **Transportes,** São Paulo, v. 21, n. 1, p.59-66, 2013. Quadrimestral.

MARTINS, Silvestre Gomes; SANTOS, Alexsandra Santana dos. **O BENCHMARKING E SUA APLICABILIDADE EM UNIDADES DE INFORMAÇÃO: UMA ABORDAGEM REFLEXIVA.** 2009. 12 f. Monografia (Especialização) - Curso de Biblioteconomia, Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, 2010.

MANDRIVA (United States). **MANDRIVA.** 2015. Disponível em: <www.mandriva.com/br>. Acesso em: 08 maio 2016.

MELO, Ednei Pacheco de. **OS SISTEMAS GNU/LINUX.** Goiás: Universidade Federal de Goiás, 2008. 48 p.

MENEGUELLI, Marcelle Fernandes et al. BENCHMARKING: FERRAMENTA A SERVIÇO DA INOVAÇÃO. **Revista Eletrônica da Faculdade Metodista Granbery,** Juiz de Fora, v. 3, p.1-16, 01 jul. 2007. Semestral.

MICROSOFT (United States). **Guia de Licenciamento:** Integrador de Sistemas OEM Microsoft. 2016. Disponível em: <ownload.microsoft.com/download/B/7/1/.../Guia de Licenciamento.pdf>. Acesso em: 22 maio 2016.

MICROSOFT (United States) (Org.). **Learn about Windows 10 IoT Core:** Why Windows 10 IoT Core?. 2016. Disponível em: <https://developer.microsoft.com/en-us/windows/iot/iotcore>. Acesso em: 22 maio 2016.

MICROSOFT (United States) (Org.). **Uma história do Windows.** 2015. Disponível em: <http://windows.microsoft.com/pt-br/windows/history#T1=era0>. Acesso em: 21 maio 2016.

MONK, Simon. **Raspberry Pi Cookbook.** Sebastopol: O'reilly, 2013. 410 p.

PEREIRA, Eduino da Veiga. **Integração dos Sistemas Operativos Windows e Linux:** Análise e Descrição dos Mecanismos de Integração. 2006. 81 f. Monografia (Especialização) - Curso de Engenharia de Sistemas e Informática, Universidade Jean Piaget de Cabo Verde, Cidade da Praia, 2006. Cap. 2.

PEREIRA, Renata I. S.; JUCÁ, Sandro C. S.. **Sistema Embarcado Linux baseado em Raspberry Pi para gravação online de Microcontroladores PIC.** 2015. 6 f. Monografia (Especialização) - Curso de Engenharia Elétrica, Universidade Federal do Ceará, Fortaleza, 2015.

PEREIRA, Vivian Cristhiane Monteiro. **TESTE U DE MANN- WHITNEY (TESTE U).** 2010. Disponível em: <http://www.leg.ufpr.br/lib/exe/fetch.php/disciplinas:ce001:vivian_-_teste_u_de_mann-whitney.pdf>. Acesso em: 12 nov. 2016.

RASPBERRY PI. **30:00 MINUTE PROJECTS.** Cambridge: The Magpi, v. 39, 01 nov. 2015. Mensal. Disponível em: <raspberrypi.org/magpi>. Acesso em: 08 abr. 2016.

RASPBERRY PI, Foundation. **DOWNLOADS.** 2016. Disponível em: <https://www.raspberrypi.org/downloads/>. Acesso em: 11 abr. 2016.

RASPBERRY PI, Foundation. **USB.** 2016. Disponível em: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/usb/README.md>. Acesso em: 10 abr. 2016.

RASPBERRY PI, Foundation. **POWER SUPPLY.** 2016. Disponível em: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md>. Acesso em: 10 abr. 2016.

RASPBERRY PI. **INTRODUCING... THE RASPBERRY PI: A new breed of computer.** Cambridge: The Magpi, v. 1, 01 maio 2012. Mensal. Disponível em: <raspberrypi.org/magpi>. Acesso em: 10 abr. 2016.

RASPBERRY (United States). **Welcome to Raspbian.** 2016. Disponível em: <https://www.raspbian.org>. Acesso em: 03 maio 2016.

RICHARDSON, Matt; WALLACE, Shawn. **Getting Started with Raspberry Pi.** Sebastopol: O'reilly, 2012. 177 p.

SABER ELETRÔNICA: Industrial. São Paulo: Saber, mar. abr. 2013. Mensal. 2013. Disponível em: <http://www.revistapcecia.com.br/download/SE468_webS2.pdf>. Acesso em: 27 mar. 2016.

SILVA, Tais Medeiros. **TESTE t-STUDENT:** TESTE IGUALDADE DE VARIÂNCIAS. 2014. 14 f. Monografia (Especialização) - Curso de Estatística, Universidade Federal do Pará, Belém, 2014.

SOTO, Fernando Miguel de Alava; SILVA, Rodrigo de Losina; MENEZES, Alexandre Folle de. **Linux Introdução.** 1.3 Porto Alegre - Rs: Acreinfo - Consultoria e Serviços de Informática, 2002. 93 p.

SPENDOLINI, Michael J. **Benchmarking.** São Paulo: Makroon Books, 1993.

SUSE (Brasil). **Suse.** 2016. Disponível em: <https://www.suse.com/pt-br/>. Acesso em: 08 maio 2016.

UPTON, Eben; HALFACREE, Gareth. **Raspberry Pi User Guide.** São Paulo: Novatec, 2012. 152 p.

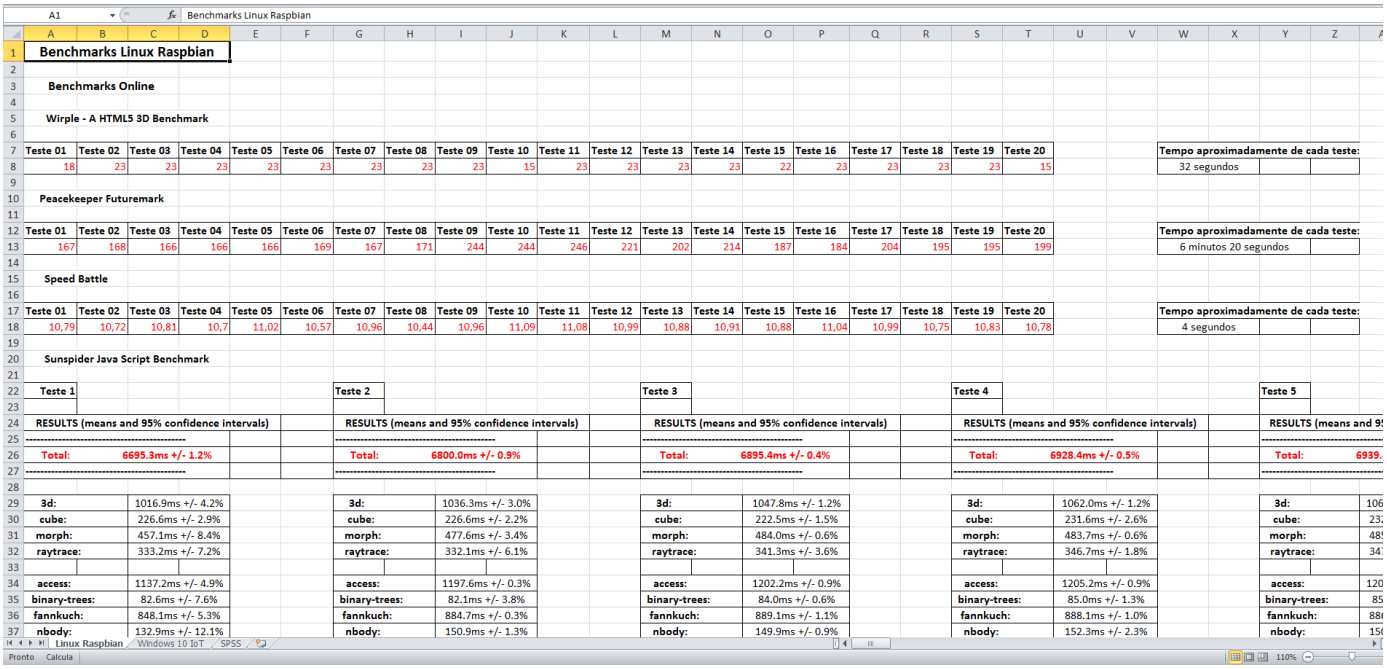
WAQUED, CÁrbio Almeida. **BENCHMARKING COMO BASE PARA MELHORIA CONTÍNUA DE PROCESSOS E SUA APLICABILIDADE EM REPRESENTANTES REGIONAIS.** 2002. 122 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2002. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/83005/192598.pdf?sequence=1>. Acesso em: 06 dez. 2015.

WARNER, Timothy L.. **Hacking Raspberry Pi.** Usa: Que, 2014. 523 p.

ZAIRI, Mohamed; LEONARD, Paul. **Benchmarking Prático –** O Guia Completo. São Paulo: Atlas, 1995

APÊNDICE(S)

APÊNDICE A – Planilha de Benchmarkings



A1 | Benchmarks Linux Raspbian

Benchmarks Linux Raspbian

Benchmarks Online

Wirple - A HTML5 3D Benchmark

Teste 01	Teste 02	Teste 03	Teste 04	Teste 05	Teste 06	Teste 07	Teste 08	Teste 09	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20	Tempo aproximadamente de cada teste:
18	23	23	23	23	23	23	23	23	15	23	23	23	23	22	23	23	23	23	15	32 segundos

Peacekeeper Futuremark

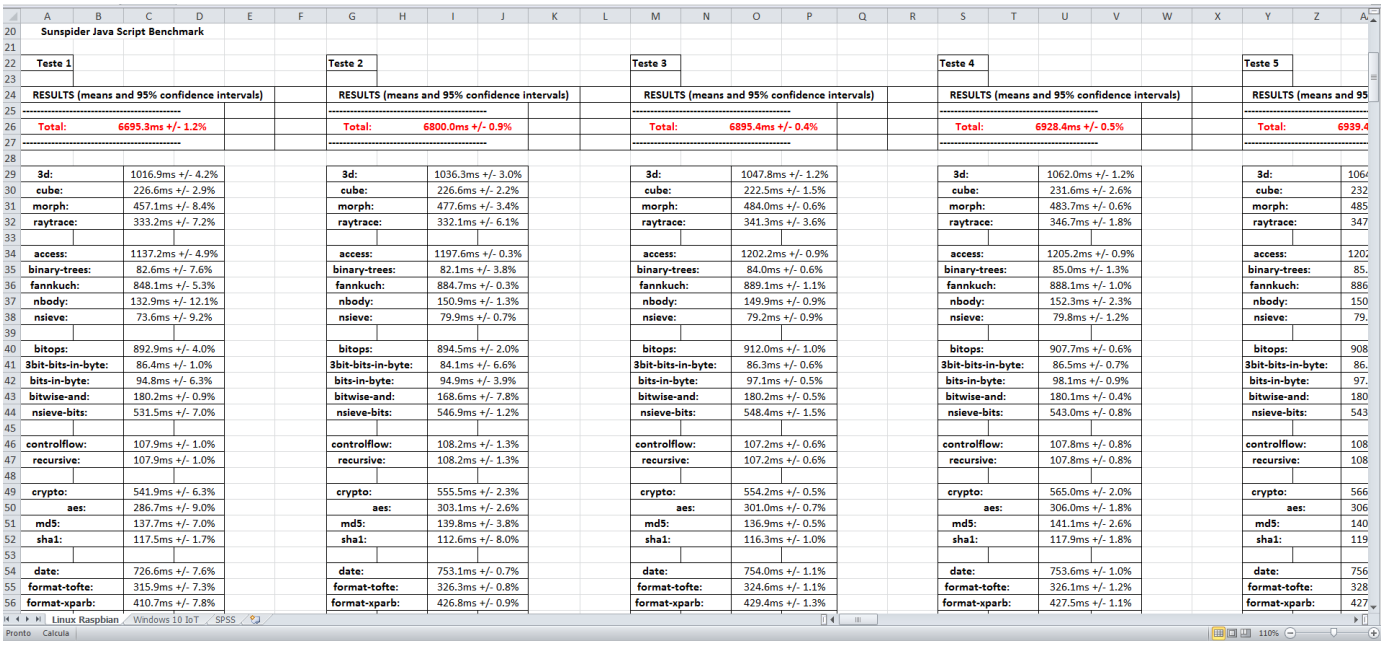
Teste 01	Teste 02	Teste 03	Teste 04	Teste 05	Teste 06	Teste 07	Teste 08	Teste 09	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20	Tempo aproximadamente de cada teste:
167	168	166	166	166	169	167	171	244	244	246	221	202	214	187	184	204	195	195	199	6 minutos 20 segundos

Speed Battle

Teste 01	Teste 02	Teste 03	Teste 04	Teste 05	Teste 06	Teste 07	Teste 08	Teste 09	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20	Tempo aproximadamente de cada teste:
10,79	10,72	10,81	10,7	11,02	10,57	10,96	10,44	10,96	11,09	11,08	10,99	10,88	10,91	10,88	11,04	10,99	10,75	10,83	10,78	4 segundos

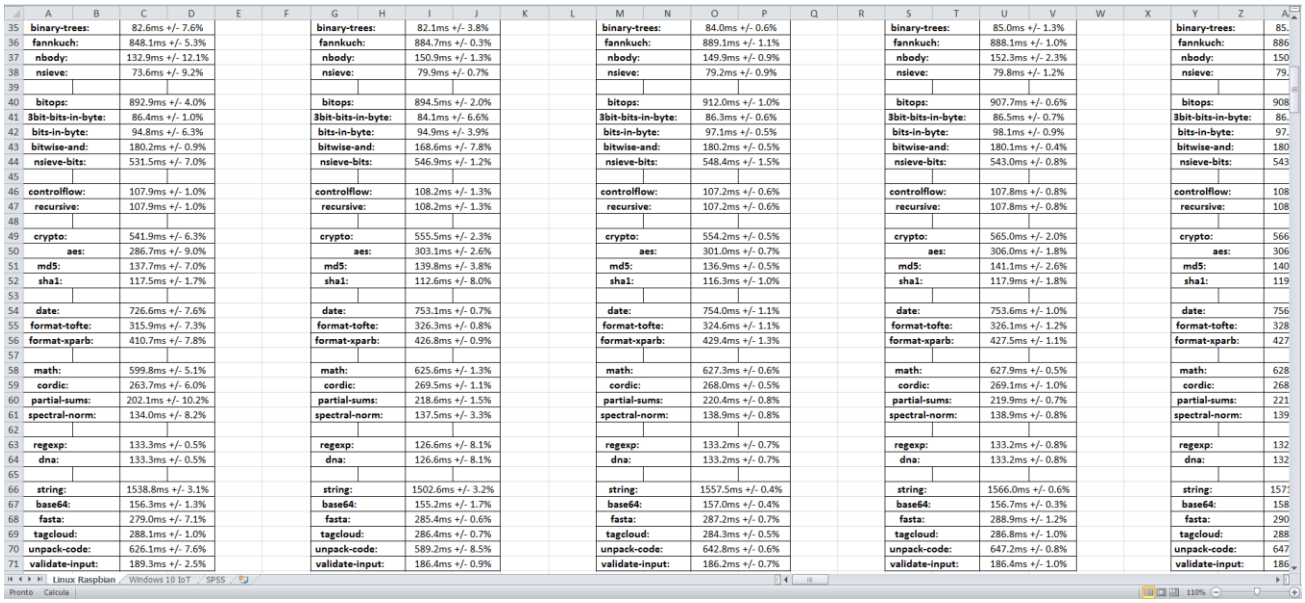
Sunspider Java Script Benchmark

Teste 1		Teste 2		Teste 3		Teste 4		Teste 5	
RESULTS (means and 95% confidence intervals)		RESULTS (means and 95% confidence intervals)		RESULTS (means and 95% confidence intervals)		RESULTS (means and 95% confidence intervals)		RESULTS (means and 95	
Total:	6695.3ms +/- 1.2%	Total:	6800.0ms +/- 0.9%	Total:	6895.4ms +/- 0.4%	Total:	6928.4ms +/- 0.5%	Total:	6939.
3d:	1016.9ms +/- 4.2%	3d:	1036.3ms +/- 3.0%	3d:	1047.8ms +/- 1.2%	3d:	1062.0ms +/- 1.2%	3d:	106
cube:	226.6ms +/- 2.9%	cube:	226.6ms +/- 2.2%	cube:	222.5ms +/- 1.5%	cube:	231.6ms +/- 2.6%	cube:	23
morph:	457.1ms +/- 8.4%	morph:	477.6ms +/- 3.4%	morph:	484.0ms +/- 0.6%	morph:	483.7ms +/- 0.6%	morph:	48
raytrace:	333.2ms +/- 7.2%	raytrace:	332.1ms +/- 6.1%	raytrace:	341.3ms +/- 3.6%	raytrace:	346.7ms +/- 1.8%	raytrace:	34
access:	1137.2ms +/- 4.9%	access:	1197.6ms +/- 0.3%	access:	1202.2ms +/- 0.9%	access:	1205.2ms +/- 0.9%	access:	120
binary-trees:	82.6ms +/- 7.6%	binary-trees:	82.1ms +/- 3.8%	binary-trees:	84.0ms +/- 0.6%	binary-trees:	85.0ms +/- 1.3%	binary-trees:	85
fannkuch:	848.1ms +/- 5.3%	fannkuch:	884.7ms +/- 0.3%	fannkuch:	889.1ms +/- 1.1%	fannkuch:	888.1ms +/- 1.0%	fannkuch:	88
nbody:	132.9ms +/- 12.1%	nbody:	150.9ms +/- 1.3%	nbody:	149.9ms +/- 0.9%	nbody:	152.3ms +/- 2.3%	nbody:	15

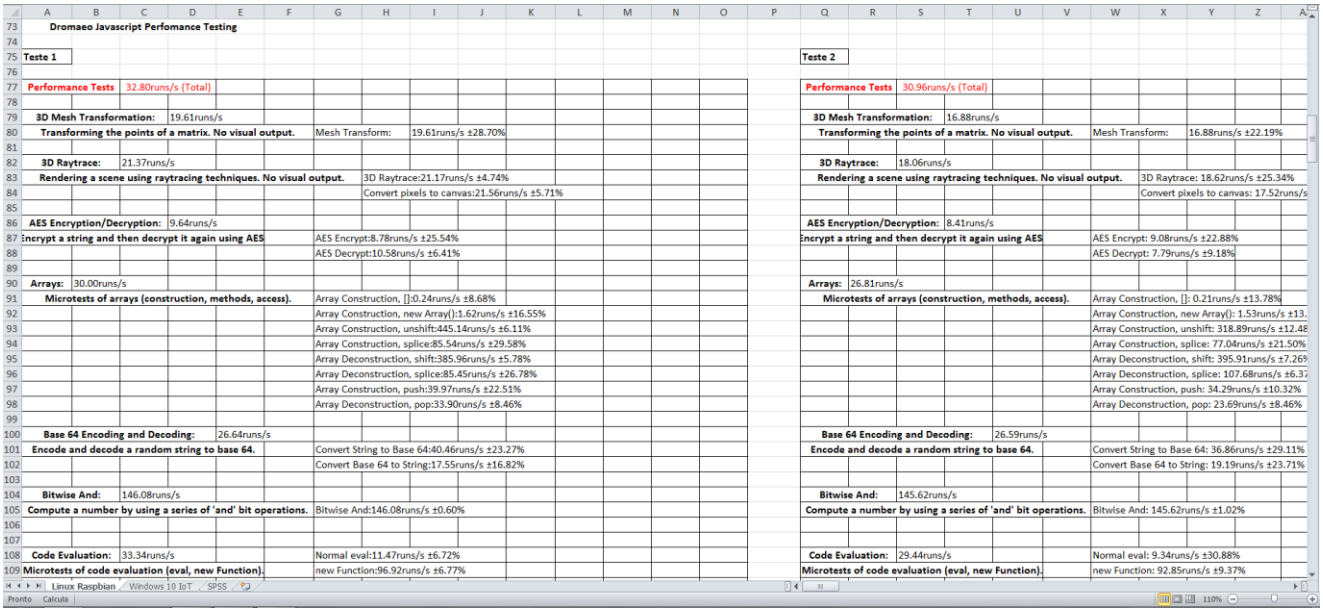


Sunspider Java Script Benchmark

Teste 1		Teste 2		Teste 3		Teste 4		Teste 5	
RESULTS (means and 95% confidence intervals)		RESULTS (means and 95% confidence intervals)		RESULTS (means and 95% confidence intervals)		RESULTS (means and 95% confidence intervals)		RESULTS (means and 95	
Total:	6695.3ms +/- 1.2%	Total:	6800.0ms +/- 0.9%	Total:	6895.4ms +/- 0.4%	Total:	6928.4ms +/- 0.5%	Total:	6939.4
3d:	1016.9ms +/- 4.2%	3d:	1036.3ms +/- 3.0%	3d:	1047.8ms +/- 1.2%	3d:	1062.0ms +/- 1.2%	3d:	106
cube:	226.6ms +/- 2.9%	cube:	226.6ms +/- 2.2%	cube:	222.5ms +/- 1.5%	cube:	231.6ms +/- 2.6%	cube:	232
morph:	457.1ms +/- 8.4%	morph:	477.6ms +/- 3.4%	morph:	484.0ms +/- 0.6%	morph:	483.7ms +/- 0.6%	morph:	485
raytrace:	333.2ms +/- 7.2%	raytrace:	332.1ms +/- 6.1%	raytrace:	341.3ms +/- 3.6%	raytrace:	346.7ms +/- 1.8%	raytrace:	347
access:	1137.2ms +/- 4.9%	access:	1197.6ms +/- 0.3%	access:	1202.2ms +/- 0.9%	access:	1205.2ms +/- 0.9%	access:	120
binary-trees:	82.6ms +/- 7.6%	binary-trees:	82.1ms +/- 3.8%	binary-trees:	84.0ms +/- 0.6%	binary-trees:	85.0ms +/- 1.3%	binary-trees:	85.
fannkuch:	848.1ms +/- 5.3%	fannkuch:	884.7ms +/- 0.3%	fannkuch:	889.1ms +/- 1.1%	fannkuch:	888.1ms +/- 1.0%	fannkuch:	886
nbody:	132.9ms +/- 12.1%	nbody:	150.9ms +/- 1.3%	nbody:	149.9ms +/- 0.9%	nbody:	152.3ms +/- 2.3%	nbody:	150
nsieve:	73.6ms +/- 9.2%	nsieve:	79.9ms +/- 0.7%	nsieve:	79.2ms +/- 0.9%	nsieve:	79.8ms +/- 1.2%	nsieve:	79.
bitops:	892.9ms +/- 4.0%	bitops:	894.5ms +/- 2.0%	bitops:	912.0ms +/- 1.0%	bitops:	907.7ms +/- 0.6%	bitops:	908
3bit-bits-in-byte:	86.4ms +/- 1.0%	3bit-bits-in-byte:	84.1ms +/- 6.6%	3bit-bits-in-byte:	86.3ms +/- 0.6%	3bit-bits-in-byte:	86.5ms +/- 0.7%	3bit-bits-in-byte:	86.
bits-in-byte:	94.8ms +/- 6.3%	bits-in-byte:	94.9ms +/- 3.9%	bits-in-byte:	97.1ms +/- 0.5%	bits-in-byte:	98.1ms +/- 0.9%	bits-in-byte:	97.
bitwise-and:	180.2ms +/- 0.9%	bitwise-and:	168.6ms +/- 7.8%	bitwise-and:	180.2ms +/- 0.5%	bitwise-and:	180.1ms +/- 0.4%	bitwise-and:	180
nsieve-bits:	531.5ms +/- 7.0%	nsieve-bits:	546.9ms +/- 1.2%	nsieve-bits:	548.4ms +/- 1.5%	nsieve-bits:	543.0ms +/- 0.8%	nsieve-bits:	543
controlflow:	107.9ms +/- 1.0%	controlflow:	108.2ms +/- 1.3%	controlflow:	107.2ms +/- 0.6%	controlflow:	107.8ms +/- 0.8%	controlflow:	108
recursive:	107.9ms +/- 1.0%	recursive:	108.2ms +/- 1.3%	recursive:	107.2ms +/- 0.6%	recursive:	107.8ms +/- 0.8%	recursive:	108
crypto:	541.9ms +/- 6.3%	crypto:	555.5ms +/- 2.3%	crypto:	554.2ms +/- 0.5%	crypto:	565.0ms +/- 2.0%	crypto:	566
aes:	286.7ms +/- 9.0%	aes:	303.1ms +/- 2.6%	aes:	301.0ms +/- 0.7%	aes:	306.0ms +/- 1.8%	aes:	306
md5:	137.7ms +/- 7.0%	md5:	139.8ms +/- 3.8%	md5:	136.9ms +/- 0.5%	md5:	141.1ms +/- 2.6%	md5:	140
sha1:	117.5ms +/- 1.7%	sha1:	112.6ms +/- 8.0%	sha1:	116.3ms +/- 1.0%	sha1:	117.9ms +/- 1.8%	sha1:	119
date:	726.6ms +/- 7.6%	date:	753.1ms +/- 0.7%	date:	754.0ms +/- 1.1%	date:	753.6ms +/- 1.0%	date:	756
format-tofte:	315.9ms +/- 7.3%	format-tofte:	326.3ms +/- 0.8%	format-tofte:	324.6ms +/- 1.1%	format-tofte:	326.1ms +/- 1.2%	format-tofte:	328
format-xparb:	410.7ms +/- 7.8%	format-xparb:	426.8ms +/- 0.9%	format-xparb:	429.4ms +/- 1.3%	format-xparb:	427.5ms +/- 1.1%	format-xparb:	427



35	binary-trees:	82.6ms +/- 7.6%			binary-trees:	82.1ms +/- 3.8%			binary-trees:	84.0ms +/- 0.6%			binary-trees:	85.0ms +/- 1.3%			binary-trees:	85.
36	fannkuch:	848.1ms +/- 5.3%			fannkuch:	884.7ms +/- 0.3%			fannkuch:	889.1ms +/- 1.1%			fannkuch:	888.1ms +/- 1.0%			fannkuch:	886
37	nbody:	132.9ms +/- 12.1%			nbody:	150.9ms +/- 1.3%			nbody:	149.9ms +/- 0.9%			nbody:	152.3ms +/- 2.3%			nbody:	150
38	nsieve:	73.6ms +/- 9.2%			nsieve:	79.9ms +/- 0.7%			nsieve:	79.2ms +/- 0.9%			nsieve:	79.8ms +/- 1.2%			nsieve:	79.
39																		
40	bitops:	892.9ms +/- 4.0%			bitops:	894.5ms +/- 2.0%			bitops:	912.0ms +/- 1.0%			bitops:	907.7ms +/- 0.6%			bitops:	908
41	3bit-bits-in-byte:	86.4ms +/- 1.0%			3bit-bits-in-byte:	84.1ms +/- 6.6%			3bit-bits-in-byte:	86.3ms +/- 0.6%			3bit-bits-in-byte:	86.5ms +/- 0.7%			3bit-bits-in-byte:	86.
42	bits-in-byte:	94.8ms +/- 6.3%			bits-in-byte:	94.9ms +/- 3.9%			bits-in-byte:	97.1ms +/- 0.5%			bits-in-byte:	98.1ms +/- 0.9%			bits-in-byte:	97.
43	bitwise-and:	180.2ms +/- 0.9%			bitwise-and:	168.6ms +/- 7.8%			bitwise-and:	180.2ms +/- 0.5%			bitwise-and:	180.1ms +/- 0.4%			bitwise-and:	180
44	nsieve-bits:	531.5ms +/- 7.0%			nsieve-bits:	546.9ms +/- 1.2%			nsieve-bits:	548.4ms +/- 1.5%			nsieve-bits:	543.0ms +/- 0.8%			nsieve-bits:	543
45																		
46	controlflow:	107.9ms +/- 1.0%			controlflow:	108.2ms +/- 1.3%			controlflow:	107.2ms +/- 0.6%			controlflow:	107.8ms +/- 0.8%			controlflow:	108
47	recursive:	107.9ms +/- 1.0%			recursive:	108.2ms +/- 1.3%			recursive:	107.2ms +/- 0.6%			recursive:	107.8ms +/- 0.8%			recursive:	108
48																		
49	crypto:	541.9ms +/- 6.3%			crypto:	555.5ms +/- 2.3%			crypto:	554.2ms +/- 0.5%			crypto:	565.0ms +/- 2.0%			crypto:	566
50	aes:	286.7ms +/- 9.0%			aes:	303.1ms +/- 2.6%			aes:	301.0ms +/- 0.7%			aes:	306.0ms +/- 1.8%			aes:	306
51	md5:	137.7ms +/- 7.0%			md5:	139.8ms +/- 3.8%			md5:	136.9ms +/- 0.5%			md5:	141.1ms +/- 2.6%			md5:	140
52	sha1:	117.5ms +/- 1.7%			sha1:	112.6ms +/- 8.0%			sha1:	116.3ms +/- 1.0%			sha1:	117.9ms +/- 1.8%			sha1:	119
53																		
54	date:	726.6ms +/- 7.6%			date:	753.1ms +/- 0.7%			date:	754.0ms +/- 1.1%			date:	753.6ms +/- 1.0%			date:	756
55	format-tofte:	315.9ms +/- 7.3%			format-tofte:	326.3ms +/- 0.8%			format-tofte:	324.6ms +/- 1.1%			format-tofte:	326.1ms +/- 1.2%			format-tofte:	328
56	format-xparb:	410.7ms +/- 7.8%			format-xparb:	426.8ms +/- 0.9%			format-xparb:	429.4ms +/- 1.3%			format-xparb:	427.5ms +/- 1.1%			format-xparb:	427
57																		
58	math:	599.8ms +/- 5.1%			math:	625.6ms +/- 1.3%			math:	627.3ms +/- 0.6%			math:	627.9ms +/- 0.5%			math:	628
59	cordic:	263.7ms +/- 6.0%			cordic:	269.5ms +/- 1.1%			cordic:	268.0ms +/- 0.5%			cordic:	269.1ms +/- 1.0%			cordic:	268
60	partial-sums:	202.1ms +/- 10.2%			partial-sums:	218.6ms +/- 1.5%			partial-sums:	220.4ms +/- 0.8%			partial-sums:	219.9ms +/- 0.7%			partial-sums:	221
61	spectral-norm:	134.0ms +/- 8.2%			spectral-norm:	137.5ms +/- 3.3%			spectral-norm:	138.9ms +/- 0.8%			spectral-norm:	138.9ms +/- 0.8%			spectral-norm:	139
62																		
63	regexp:	133.3ms +/- 0.5%			regexp:	126.6ms +/- 8.1%			regexp:	133.2ms +/- 0.7%			regexp:	133.2ms +/- 0.8%			regexp:	132
64	dna:	133.3ms +/- 0.5%			dna:	126.6ms +/- 8.1%			dna:	133.2ms +/- 0.7%			dna:	133.2ms +/- 0.8%			dna:	132
65																		
66	string:	1538.8ms +/- 3.1%			string:	1502.6ms +/- 3.2%			string:	1557.5ms +/- 0.4%			string:	1566.0ms +/- 0.6%			string:	157
67	base64:	156.3ms +/- 1.3%			base64:	155.2ms +/- 1.7%			base64:	157.0ms +/- 0.4%			base64:	156.7ms +/- 0.3%			base64:	158
68	fasta:	279.0ms +/- 7.1%			fasta:	285.4ms +/- 0.6%			fasta:	287.2ms +/- 0.7%			fasta:	288.9ms +/- 1.2%			fasta:	290
69	tagcloud:	288.1ms +/- 1.0%			tagcloud:	286.4ms +/- 0.7%			tagcloud:	284.3ms +/- 0.5%			tagcloud:	286.8ms +/- 1.0%			tagcloud:	288
70	unpack-code:	626.1ms +/- 7.6%			unpack-code:	589.2ms +/- 8.5%			unpack-code:	642.8ms +/- 0.6%			unpack-code:	647.2ms +/- 0.8%			unpack-code:	647
71	validate-input:	189.3ms +/- 2.5%			validate-input:	186.4ms +/- 0.9%			validate-input:	186.2ms +/- 0.7%			validate-input:	186.4ms +/- 1.0%			validate-input:	186



73	Dromaeo Javascript Perfomance Testing																												
74																													
75	Teste 1																Teste 2												
76																													
77	Performance Tests	32.80runs/s (Total)															Performance Tests	30.96runs/s (Total)											
78																													
79	3D Mesh Transformation:	19.61runs/s															3D Mesh Transformation:	16.88runs/s											
80	Transforming the points of a matrix. No visual output.					Mesh Transform:	19.61runs/s ±28.70%										Transforming the points of a matrix. No visual output.							Mesh Transform:	16.88runs/s ±22.19%				
81																													
82	3D Raytrace:	21.37runs/s															3D Raytrace:	18.06runs/s											
83	Rendering a scene using raytracing techniques. No visual output.					3D Raytrace:21.17runs/s ±4.74%											Rendering a scene using raytracing techniques. No visual output.							3D Raytrace: 18.62runs/s ±25.34%				
84						Convert pixels to canvas:21.56runs/s ±5.71%																			Convert pixels to canvas: 17.52runs/s				
85																													
86	AES Encryption/Decryption:	9.64runs/s															AES Encryption/Decryption:	8.41runs/s											
87	Encrypt a string and then decrypt it again using AES					AES Encrypt:8.78runs/s ±25.54%											Encrypt a string and then decrypt it again using AES							AES Encrypt: 9.08runs/s ±22.88%				
88						AES Decrypt:10.58runs/s ±6.41%																			AES Decrypt: 7.79runs/s ±9.18%				
89																													
90	Arrays:	30.00runs/s															Arrays:	26.81runs/s											
91	Microtests of arrays (construction, methods, access).					Array Construction, []:0.24runs/s ±8.68%											Microtests of arrays (construction, methods, access).							Array Construction, []: 0.21runs/s ±13.78%				
92						Array Construction, new Array():1.62runs/s ±16.55%																			Array Construction, new Array(): 1.53runs/s ±13.				
93						Array Construction, unshift:445.14runs/s ±6.11%																			Array Construction, unshift: 318.89runs/s ±12.48				
94						Array Construction, splice:85.54runs/s ±29.58%																			Array Construction, splice: 77.04runs/s ±21.50%				
95						Array Deconstruction, shift:385.96runs/s ±5.78%																			Array Deconstruction, shift: 395.91runs/s ±7.26%				
96						Array Deconstruction, splice:85.45runs/s ±26.78%																			Array Deconstruction, splice: 107.68runs/s ±6.37				
97						Array Construction, push:39.97runs/s ±22.51%																			Array Construction, push: 34.29runs/s ±10.32%				
98						Array Deconstruction, pop:33.90runs/s ±8.46%																			Array Deconstruction, pop: 23.69runs/s ±8.46%				
99																													
100	Base 64 Encoding and Decoding:	26.64runs/s															Base 64 Encoding and Decoding:	26.59runs/s											
101	Encode and decode a random string to base 64.					Convert String to Base 64:40.46runs/s ±23.27%											Encode and decode a random string to base 64.							Convert String to Base 64: 36.86runs/s ±29.11%				
102						Convert Base 64 to String:17.55runs/s ±16.82%																			Convert Base 64 to String: 19.19runs/s ±23.71%				
103																													
104	Bitwise And:	146.08runs/s															Bitwise And:	145.62runs/s											
105	Compute a number by using a series of 'and' bit operations.					Bitwise And:146.08runs/s ±0.60%											Compute a number by using a series of 'and' bit operations.							Bitwise And: 145.62runs/s ±1.02%				
106																													
107																													
108	Code Evaluation:	33.34runs/s															Code Evaluation:	29.44runs/s											
109	Microtests of code evaluation (eval, new Function).					Normal eval:11.47runs/s ±6.72%											Microtests of code evaluation (eval, new Function).							Normal eval: 9.34runs/s ±30.88%				
						new Function:96.92runs/s ±6.77%																			new Function: 92.85runs/s ±9.37%				

108	**Code Evaluation:**	33.34runs/s	Normal eval:11.47runs/s ±6.72%	**Code Evaluation:**	29.44runs/s	Normal eval: 9.34runs/s ±30.88%
109	**Microtests of code evaluation (eval, new Function).**		new Function:96.92runs/s ±6.77%	**Microtests of code evaluation (eval, new Function).**		new Function: 92.85runs/s ±9.37%
110						
111	**Compute Bits in Byte:**	37.83runs/s		**Compute Bits in Byte:**	29.46runs/s	
112	**Compute the number of bits in a number using bitops.**		3bit bits in byte:37.83runs/s ±1.01%	**Compute the number of bits in a number using bitops.**		3bit bits in byte: 29.46runs/s ±23.52%
113						
114	**Compute Bits in Byte (2):**	48.34runs/s		**Compute Bits in Byte (2):**	44.09runs/s	
115	**Compute the number of bits in a number using bitops.**		Bit in byte (2):48.34runs/s ±25.21%	**Compute the number of bits in a number using bitops.**		Bit in byte (2): 44.09runs/s ±23.67%
116						
117	**DNA Sequence Alignment:**	48.95runs/s		**DNA Sequence Alignment:**	54.64runs/s	
118	**Find DNA matches within a larger sequence.**		Homo sapiens alu:33.26runs/s ±37.37%	**Find DNA matches within a larger sequence.**		Homo sapiens alu: 30.39runs/s ±28.20%
119			IUB ambiguity codes:66.42runs/s ±18.27%			IUB ambiguity codes: 94.64runs/s ±0.32%
120			Homo sapiens frequency:53.09runs/s ±18.62%			Homo sapiens frequency: 56.71runs/s ±0.75%
121						
122	**DNA Sequence Counting:**	39.12runs/s		**DNA Sequence Counting:**	40.30runs/s	
123	**Counts occurences in a DNA sequence.**		DNA Match (1):41.36runs/s ±0.32%	**Counts occurences in a DNA sequence.**		DNA Match (1): 41.23runs/s ±0.53%
124			DNA Match (2):41.09runs/s ±5.61%			DNA Match (2): 41.61runs/s ±0.31%
125			DNA Match (3):33.30runs/s ±26.07%			DNA Match (3): 41.84runs/s ±0.55%
126			DNA Replace:41.42runs/s ±17.80%			DNA Replace: 36.75runs/s ±7.06%
127						
128	**DOM Attributes:**	31.94runs/s		**DOM Attributes:**	28.37runs/s	
129	**Setting and getting DOM node attributes.**		getAttribute:65.45runs/s ±0.17%	**Setting and getting DOM node attributes**		getAttribute: 56.74runs/s ±23.64%
130			element.property:87.78runs/s ±0.83%			element.property: 86.57runs/s ±0.60%
131			setAttribute:26.82runs/s ±0.39%			setAttribute: 25.69runs/s ±0.97%
132			element.property = value:26.41runs/s ±3.08%			element.property = value: 20.48runs/s ±28.19%
133			element.expando = value:14.08runs/s ±23.05%			element.expando = value: 13.31runs/s ±23.81%
134			element.expando:18.53runs/s ±27.09%			element.expando: 15.14runs/s ±10.53%
135						
136	**DOM Attributes (Prototype):**	46.86runs/s		**DOM Attributes (Prototype):**	46.04runs/s	
137	**Setting and getting DOM node attributes using the Prototype JavaScript Library.**		Prototype - addClassName:39.34runs/s ±29.55%	**Setting and getting DOM node attributes using the Prototype JavaScript Library.**		Prototype - add
138			Prototype - removeClassName:41.71runs/s ±6.17%			Prototype - rem
139			Prototype - hasClassName:54.46runs/s ±7.08%			Prototype - has
140			Prototype - readAttribute:70.08runs/s ±5.51%			Prototype - read
141			Prototype - writeAttribute:36.07runs/s ±29.49%			Prototype - writ
142						
143	**DOM Attributes (jQuery):**	57.85runs/s		**DOM Attributes (jQuery):**	47.68runs/s	
144	**Setting and getting DOM node attributes using the jQuery JavaScript Library.**		jQuery - addClass:81.20runs/s ±19.23%	**Setting and getting DOM node attributes using the jQuery JavaScript Library.**		jQuery - addClass: 71.57ru

Linux Raspbian / Windows 10 IoT / SPSS

Pronto Calcula 110%

143	**DOM Attributes (jQuery):**	57.85runs/s		**DOM Attributes (jQuery):**	47.68runs/s	
144	**Setting and getting DOM node attributes using the jQuery JavaScript Library.**		jQuery - addClass:81.20runs/s ±19.23%	**Setting and getting DOM node attributes using the jQuery JavaScript Library.**		jQuery - addClass: 71.57ru
145			jQuery - removeClass:84.80runs/s ±6.60%			jQuery - removeClass: 69.8
146			jQuery - hasClass x10:39.74runs/s ±2.82%			jQuery - hasClass x10: 28.
147			jQuery - attr(class) x100:41.91runs/s ±0.10%			jQuery - attr(class) x100: 3
148			jQuery - attr(class,test):77.79runs/s ±24.80%			jQuery - attr(class,test): 7
149			jQuery - removeAttribute:42.00runs/s ±6.16%			jQuery - removeAttribute:
150						
151	**DOM Events (Prototype):**	29.73runs/s		**DOM Events (Prototype):**	27.99runs/s	
152	**Binding, removing, and triggering DOM events using the Prototype JavaScript Library.**		Prototype - observe:20.07runs/s ±23.25%	**Binding, removing, and triggering DOM events using the Prototype JavaScript Library.**		Prototype - obse
153			Prototype - fire:33.27runs/s ±11.35%			Prototype - fire:
154			Prototype - stopObserving:39.35runs/s ±1.25%			Prototype - stop
155						
156	**DOM Events (jQuery):**	4.15runs/s		**DOM Events (jQuery):**	3.72runs/s	
157	**Binding, removing, and triggering DOM events using the jQuery JavaScript Library.**		jQuery - bind:18.70runs/s ±32.32%	**Binding, removing, and triggering DOM events using the jQuery JavaScript Library.**		jQuery - bind: 1
158			jQuery - trigger:0.26runs/s ±3.50%			jQuery - trigger:
159			jQuery - unbind x10:14.87runs/s ±25.19%			jQuery - unbind
160						
161	**DOM Modification:**	40.58runs/s		**DOM Modification:**	37.45runs/s	
162	**Creating and injecting DOM nodes into a document.**		createElement:65.75runs/s ±38.10%	**Creating and injecting DOM nodes into a document.**		createElement: 61.83runs/s ±61.72%
163			createTextNode:14.04runs/s ±75.02%			createTextNode: 15.06runs/s ±52.94%
164			innerHTML:9.48runs/s ±26.06%			innerHTML: 9.15runs/s ±27.60%
165			cloneNode:25.74runs/s ±0.46%			cloneNode: 19.67runs/s ±19.71%
166			appendChild:144.37runs/s ±23.18%			appendChild: 136.48runs/s ±18.43%
167			insertBefore:137.24runs/s ±2.31%			insertBefore: 120.57runs/s ±23.27%
168						
169	**DOM Modification (Prototype):**	40.19runs/s		**DOM Modification (Prototype):**	36.50runs/s	
170	**Creating and injecting DOM nodes into a document using the Prototype JavaScript Library.**		Prototype - update():55.25runs/s ±40.60%	**Creating and injecting DOM nodes into a document using the Prototype JavaScript Library.**		Prototype - upd
171			Prototype - before:40.51runs/s ±23.80%			Prototype - befo
172			Prototype - after:32.60runs/s ±39.51%			Prototype - afte
173			Prototype - prepend:33.85runs/s ±19.50%			Prototype - prep
174			Prototype - append:42.46runs/s ±23.39%			Prototype - app
175						
176	**DOM Modification (jQuery):**	150.69runs/s		**DOM Modification (jQuery):**	128.86runs/s	
177	**Creating and injecting DOM nodes into a document using the jQuery JavaScript Library.**		jQuery - html():126.40runs/s ±22.18%	**Creating and injecting DOM nodes into a document using the jQuery JavaScript Library.**		jQuery - html():
178			jQuery - before():149.05runs/s ±32.27%			jQuery - before(
179			jQuery - after():184.71runs/s ±7.08%			jQuery - after():

Linux Raspbian / Windows 10 IoT / SPSS

Pronto Calcula 110%

Row	A–E	C/D/S/T	F–O	Q–U	S/T	V–AA
176	**DOM Modification (jQuery):**	150.69runs/s		**DOM Modification (jQuery):**	128.86runs/s	
177	**Creating and injecting DOM nodes into a document using the jQuery JavaScript Library.**		jQuery - html():126.40runs/s ±22.18%	**Creating and injecting DOM nodes into a document using the jQuery JavaScript Library.**		jQuery - html():
178			jQuery - before():149.05runs/s ±32.27%			jQuery - before(
179			jQuery - after():184.71runs/s ±7.08%			jQuery - after():
180			jQuery - prepend():158.66runs/s ±6.48%			jQuery - prepen
181			jQuery - append():140.73runs/s ±59.56%			jQuery - append
182						
183	**DOM Query:**	968.17runs/s		**DOM Query:**	944.17runs/s	
184	**Querying DOM elements in a document.**		getElementById:29.35runs/s ±31.03%	**Querying DOM elements in a document.**		getElementById: 26.67runs/s ±26.29%
185			getElementById (not in document):200.28runs/s ±0.69%			getElementById (not in document): 159.85runs/s ±28.33%
186			getElementsByTagName(div):7454.20runs/s ±0.18%			getElementsByTagName(div): 7125.60runs/s ±0.14%
187			getElementsByTagName(p):5905.00runs/s ±21.75%			getElementsByTagName(p): 6979.20runs/s ±0.18%
188			getElementsByTagName(a):7259.60runs/s ±0.14%			getElementsByTagName(a): 7091.60runs/s ±0.18%
189			getElementsByTagName(*):7430.20runs/s ±0.43%			getElementsByTagName(*): 7193.40runs/s ±0.16%
190			getElementsByTagName (not in document):12242.20runs/s ±0.14%			getElementsByTagName (not in document): 12037.80runs/
191			getElementsByName:43.17runs/s ±27.54%			getElementsByName: 37.57runs/s ±13.97%
192			getElementsByName (not in document):101.34runs/s ±24.79%			getElementsByName (not in document): 121.90runs/s ±21.
193						
194	**DOM Style (Prototype):**	30.98runs/s		**DOM Style (Prototype):**	31.84runs/s	
195	**Getting and setting CSS information on DOM elements using the Prototype JavaScript Library**		Prototype - getStyle():20.48runs/s ±3.63%	**Getting and setting CSS information on DOM elements using the Prototype JavaScript Library**		Proto
196			Prototype - setStyle():19.89runs/s ±41.57%			Proto
197			Prototype - getHeight():17.73runs/s ±20.72%			Proto
198			Prototype - getWidth():15.08runs/s ±22.92%			Proto
199			Prototype - visible():63.82runs/s ±25.55%			Proto
200			Prototype - .show():63.30runs/s ±35.66%			Proto
201			Prototype - .hide():54.33runs/s ±18.90%			Proto
202			Prototype - .toggle() x10:35.53runs/s ±26.86%			Proto
203						
204	**DOM Style (jQuery):**	39.32runs/s		**DOM Style (jQuery):**	37.84runs/s	
205	**Getting and setting CSS information on DOM elements using the jQuery JavaScript Library.**		jQuery - css(color) x100:21.17runs/s ±25.08%	**Getting and setting CSS information on DOM elements using the jQuery JavaScript Library.**		jQuer
206			jQuery - css(color,red):56.44runs/s ±22.76%			jQuer
207			jQuery - height() x10:24.81runs/s ±26.13%			jQuer
208			jQuery - width() x10:49.49runs/s ±22.45%			jQuer
209			jQuery - .is(:visible):109.39runs/s ±28.28%			jQuer
210			jQuery - .show():18.50runs/s ±26.83%			jQuer
211			jQuery - .hide():77.69runs/s ±0.56%			jQuer
212			jQuery - .toggle():24.80runs/s ±0.71%			jQuer

Linux Raspbian / Windows 10 IoT / SPSS

Pronto Calcula 110%

Row	A–E	C/D/S/T	E–O	Q–U	S/T	U–AA
204	**DOM Style (jQuery):**	39.32runs/s		**DOM Style (jQuery):**	37.84runs/s	
205	**Getting and setting CSS information on DOM elements using the jQuery JavaScript Library.**		jQuery - css(color) x100:21.17runs/s ±25.08%	**Getting and setting CSS information on DOM elements using the jQuery JavaScript Library.**		jQuer
206			jQuery - css(color,red):56.44runs/s ±22.76%			jQuer
207			jQuery - height() x10:24.81runs/s ±26.13%			jQuer
208			jQuery - width() x10:49.49runs/s ±22.45%			jQuer
209			jQuery - .is(:visible):109.39runs/s ±28.28%			jQuer
210			jQuery - .show():18.50runs/s ±26.83%			jQuer
211			jQuery - .hide():77.69runs/s ±0.56%			jQuer
212			jQuery - .toggle():24.80runs/s ±0.71%			jQuer
213						
214	**DOM Traversal:**	27.94runs/s		**DOM Traversal:**	25.52runs/s	
215	**Traversing a DOM structure.**		firstChild:18.72runs/s ±25.86%	**Traversing a DOM structure.**		firstChild: 18.99runs/s ±25.38%
216			lastChild:21.77runs/s ±0.11%			lastChild: 17.48runs/s ±23.20%
217			nextSibling:37.04runs/s ±0.10%			nextSibling: 37.18runs/s ±0.09%
218			previousSibling:36.44runs/s ±0.07%			previousSibling: 35.74runs/s ±0.49%
219			childNodes:30.96runs/s ±0.06%			childNodes: 24.53runs/s ±23.81%
220						
221	**DOM Traversal (Prototype):**	26.21runs/s		**DOM Traversal (Prototype):**	23.45runs/s	
222	**Traversing a DOM structure using the Prototype JavaScript Library.**		Prototype - up:60.59runs/s ±43.82%	**Traversing a DOM structure using the Prototype JavaScript Library.**		Prototype - up: 60.02runs/s ±26.05%
223			Prototype - ancestors:39.93runs/s ±38.40%			Prototype - ancestors: 35.21runs/s ±2
224			Prototype - previous:26.79runs/s ±35.88%			Prototype - previous: 28.84runs/s ±32
225			Prototype - previousSiblings:20.81runs/s ±26.49%			Prototype - previousSiblings: 16.15ru
226			Prototype - next:24.35runs/s ±6.12%			Prototype - next: 24.11runs/s ±23.05
227			Prototype - nextSiblings:21.79runs/s ±27.64%			Prototype - nextSiblings: 17.64runs/s
228			Prototype - siblings:9.21runs/s ±22.36%			Prototype - siblings: 7.75runs/s ±3.2
229			Prototype - childElements:33.79runs/s ±31.61%			Prototype - childElements: 28.22runs,
230						
231	**DOM Traversal (jQuery):**	9.42runs/s		**DOM Traversal (jQuery):**	9.79runs/s	
232	**Traversing a DOM structure using the jQuery JavaScript Library**		jQuery - parent x10:10.35runs/s ±38.00%	**Traversing a DOM structure using the jQuery JavaScript Library**		jQuery - parent x10: 10.03runs/s ±30.
233			jQuery - parents x10:3.57runs/s ±2.35%			jQuery - parents x10: 3.46runs/s ±17.
234			jQuery - prev x10:21.69runs/s ±24.06%			jQuery - prev x10: 23.49runs/s ±2.60
235			jQuery - prevAll:4.80runs/s ±16.24%			jQuery - prevAll: 5.13runs/s ±26.52%
236			jQuery - next x10:20.50runs/s ±9.60%			jQuery - next x10: 21.68runs/s ±10.3
237			jQuery - nextAll:5.42runs/s ±17.77%			jQuery - nextAll: 6.14runs/s ±25.52%
238			jQuery - siblings:2.55runs/s ±24.76%			jQuery - siblings: 2.69runs/s ±21.91%
239			jQuery - children:56.90runs/s ±23.65%			jQuery - children: 56.42runs/s ±23.23
240						

Linux Raspbian / Windows 10 IoT / SPSS

Pronto Calcula 110%

Row	A–O			Q–AA		
241	Date Formatting:	21.24runs/s		Date Formatting:	21.10runs/s	
242	Converting a date into a string representation.		Date Format (2):21.24runs/s ±12.69%	Converting a date into a string representation.		Date Format (2): 21.10runs/s ±15.39%
243						
244	Date Formatting (2):	28.95runs/s		Date Formatting (2):	30.26runs/s	
245	Converting a date into a string representation.		Format Date:28.95runs/s ±27.05%	Converting a date into a string representation.		Format Date: 30.26runs/s ±28.97%
246						
247	DeltaBlue Constraint Solving:	12.66runs/s		DeltaBlue Constraint Solving:	12.77runs/s	
248	Computing a number of limitations on a set of values.		Constraint Solving:12.66runs/s ±29.49%	Computing a number of limitations on a set of values.		Constraint Solving: 12.77runs/s ±26.71%
249						
250	Fannkuch:	17.13runs/s		Fannkuch:	14.77runs/s	
251	Figure out the number of ways in which a set of numbers can be manipulated.		Pfannkuchen:17.13runs/s ±0.76%	Figure out the number of ways in which a set of numbers can be manipulated.		Pfannkuchen: 14.77runs/s
252						
253	MD5 Hashing:	51.32runs/s		MD5 Hashing:	40.12runs/s	
254	Hash a long string using MD5.		MD5:51.32runs/s ±1.14%	Hash a long string using MD5.		MD5: 40.12runs/s ±27.43%
255						
256	N-Body Rotation and Gravity:	54.13runs/s		N-Body Rotation and Gravity:	43.72runs/s	
257	Compute the location of multiple planets based upon rotation and gravity.		N-Body:54.13runs/s ±0.92%	Compute the location of multiple planets based upon rotation and gravity.		N-Body: 43.72runs/s ±26.7
258						
259	Partial Sum Calculation:	88.52runs/s		Partial Sum Calculation:	104.09runs/s	
260	Calculate the partial sum of a few different number series.		Partial Sums:88.52runs/s ±27.47%	Calculate the partial sum of a few different number series.		Partial Sums: 104.09runs/s ±20.24%
261						
262	Prime Number Computation:	34.42runs/s		Prime Number Computation:	31.00runs/s	
263	Compute the number of prime numbers in a specific range of numbers.		N-Sieve:34.42runs/s ±17.98%	Compute the number of prime numbers in a specific range of numbers.		N-Sieve: 31.00runs/s ±5.55
264						
265	Prime Number Computation (2):	8.64runs/s		Prime Number Computation (2):	11.50runs/s	
266	Compute the number of prime numbers in a specific range of numbers using bit operations.		N-Sieve Bits:8.64runs/s ±15.27%	Compute the number of prime numbers in a specific range of numbers using bit operations.		N-Sie
267						
268	RSA Encryption/Decryption:	5.30runs/s		RSA Encryption/Decryption:	6.62runs/s	
269	Encrypt a string and then decrypt it again using RSA		RSA Encrypt:21.17runs/s ±19.97%	Encrypt a string and then decrypt it again using RSA		RSA Encrypt: 28.44runs/s ±2.67%
270			RSA Decrypt:1.33runs/s ±26.61%			RSA Decrypt: 1.54runs/s ±0.63%
271						
272	RayTracer:	2.90runs/s		RayTracer:	2.70runs/s	
273	Renders a scene using raytracing (no rendering done).		RayTrace:2.90runs/s ±21.35%	Renders a scene using raytracing (no rendering done).		RayTrace: 2.70runs/s ±20.97%
274						
275	Recursive Number Calculation:	44.60runs/s		Recursive Number Calculation:	42.77runs/s	
276	Compute various numbers in a recursive manner.		Ack:60.25runs/s ±0.11%	Compute various numbers in a recursive manner.		Ack: 53.49runs/s ±22.02%
277			Fib:32.49runs/s ±0.16%			Fib: 32.31runs/s ±0.18%

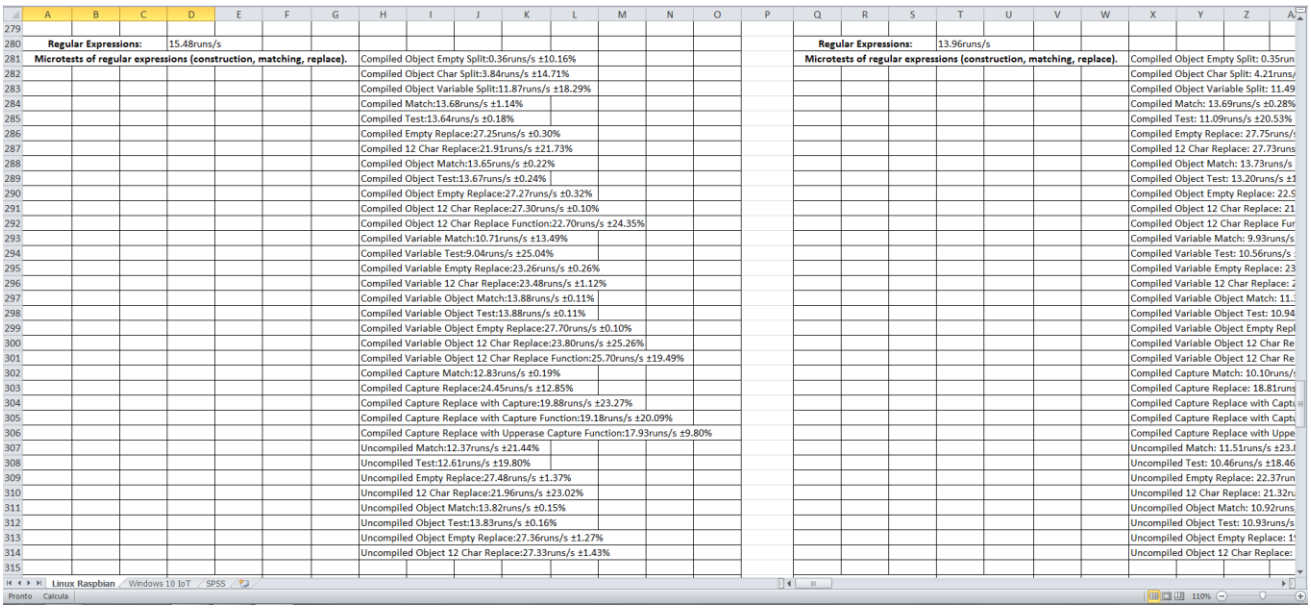
Linux Raspbian / Windows 10 IoT / SPSS

Pronto Calcula 110%

Row	A–O			Q–AA		
316	Richards Benchmarks:	29.23runs/s		Richards Benchmarks:	34.38runs/s	
317	A series of benchmarks to test the quality of system implementation languages.		Richards:29.23runs/s ±23.33%	A series of benchmarks to test the quality of system implementation languages.		Richards: 34.38
318						
319	Rotating 3D Cube:	31.36runs/s		Rotating 3D Cube:	30.87runs/s	
320	Rotating the individual pixels of a cube. No rendering done.		Rotate 3D Cube:31.36runs/s ±30.06%	Rotating the individual pixels of a cube. No rendering done.		Rotate 3D Cube: 30.87runs/s ±19.27%
321						
322	SHA1 Hashing:	46.46runs/s		SHA1 Hashing:	43.47runs/s	
323	Hash a long string using SHA1.		SHA1 Hashing:46.46runs/s ±26.82%	Hash a long string using SHA1.		SHA1 Hashing: 43.47runs/s ±24.05%
324						
325	Script Unpacking:	3.38runs/s		Script Unpacking:	3.45runs/s	
326	Decompressing scripts run through Dean Edwards' Packer.		Unpack Code:3.38runs/s ±30.52%	Decompressing scripts run through Dean Edwards' Packer.		Unpack Code: 3.45runs/s ±22.35%
327						
328	Spectral Norm of a Matrix:	46.59runs/s		Spectral Norm of a Matrix:	44.38runs/s	
329	Calculate the spectral norm of a matrix of numbers.		Spectral Norm:46.59runs/s ±0.95%	Calculate the spectral norm of a matrix of numbers.		Spectral Norm: 44.38runs/s ±16.34%
330						
331	String Parsing and Searching:	4.03runs/s		String Parsing and Searching:	3.94runs/s	
332	Tests for parsing languages and searching strings.		Earley:12.68runs/s ±35.98%	Tests for parsing languages and searching strings.		Earley: 12.43runs/s ±22.29%
333			Boyer:1.28runs/s ±36.19%			Boyer: 1.25runs/s ±35.90%
334						
335	Strings:	67.63runs/s		Strings:	67.35runs/s	
336	Microtests of strings (concatenation, methods).		Concat String:182.62runs/s ±35.15%	Microtests of strings (concatenation, methods).		Concat String: 177.25runs/s ±34.32%
337			Concat String Object:204.44runs/s ±29.54%			Concat String Object: 194.52runs/s ±30.11%
338			Concat String from charCode:257.24runs/s ±3.41%			Concat String from charCode: 277.63runs/s ±10.
339			Array String Join:207.72runs/s ±35.45%			Array String Join: 220.58runs/s ±43.44%
340			String Split:18.55runs/s ±35.28%			String Split: 18.91runs/s ±5.22%
341			String Split on Char:8.81runs/s ±26.83%			String Split on Char: 8.23runs/s ±30.84%
342			charAt:113.47runs/s ±26.28%			charAt: 137.77runs/s ±1.09%
343			[Number]:204.74runs/s ±0.42%			[Number]: 188.56runs/s ±19.67%
344			charCodeAt:123.48runs/s ±0.27%			charCodeAt: 106.15runs/s ±26.52%
345			indexOf:84.34runs/s ±0.16%			indexOf: 89.45runs/s ±0.06%
346			lastIndexOf:68.67runs/s ±0.18%			lastIndexOf: 67.04runs/s ±0.11%
347			slice:16.17runs/s ±15.37%			slice: 18.53runs/s ±24.27%
348			substr:17.74runs/s ±23.94%			substr: 19.39runs/s ±23.57%
349			substring:17.11runs/s ±25.73%			substring: 14.65runs/s ±26.31%
350			toLowerCase:186.25runs/s ±10.26%			toLowerCase: 147.26runs/s ±22.63%
351			toUpperCase:41.88runs/s ±14.86%			toUpperCase: 44.36runs/s ±17.66%
352			comparing:62.39runs/s ±21.58%			comparing: 64.52runs/s ±34.67%

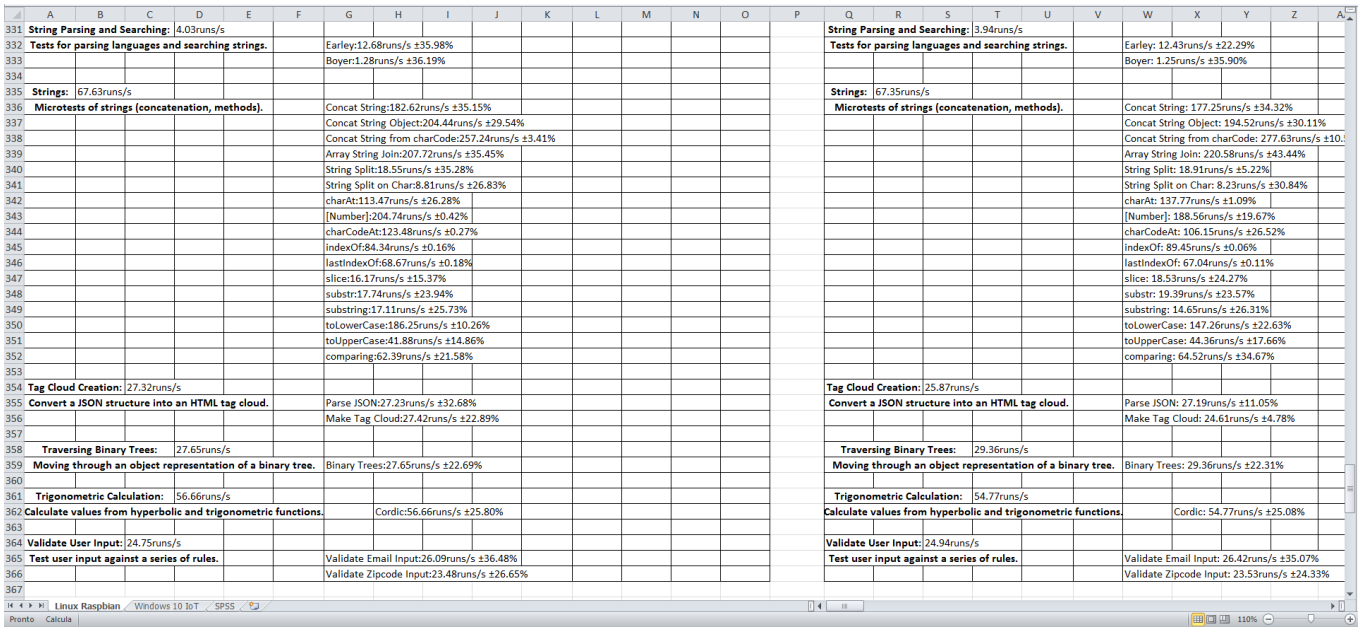
Linux Raspbian / Windows 10 IoT / SPSS

Pronto Calcula 110%



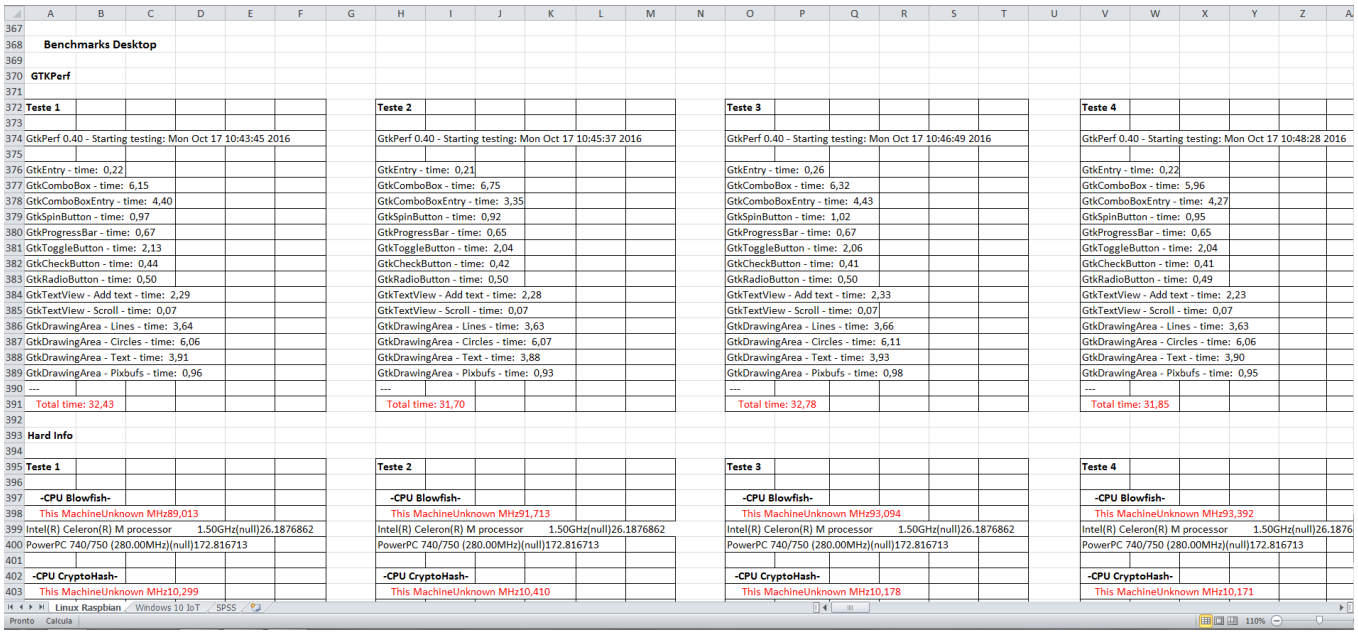
Regular Expressions:	15.48runs/s	
Microtests of regular expressions (construction, matching, replace).	Compiled Object Empty Split:0.36runs/s ±10.16%	
	Compiled Object Char Split:3.84runs/s ±14.71%	
	Compiled Object Variable Split:11.87runs/s ±18.29%	
	Compiled Match:13.68runs/s ±1.14%	
	Compiled Test:13.64runs/s ±0.18%	
	Compiled Empty Replace:27.25runs/s ±0.30%	
	Compiled 12 Char Replace:21.91runs/s ±21.73%	
	Compiled Object Match:13.65runs/s ±0.22%	
	Compiled Object Test:13.67runs/s ±0.24%	
	Compiled Object Empty Replace:27.27runs/s ±0.32%	
	Compiled Object 12 Char Replace:27.30runs/s ±0.10%	
	Compiled Object 12 Char Replace Function:22.70runs/s ±24.35%	
	Compiled Variable Match:10.71runs/s ±13.49%	
	Compiled Variable Test:9.04runs/s ±25.04%	
	Compiled Variable Empty Replace:23.26runs/s ±0.26%	
	Compiled Variable 12 Char Replace:23.48runs/s ±1.12%	
	Compiled Variable Object Match:13.88runs/s ±0.11%	
	Compiled Variable Object Test:13.88runs/s ±0.11%	
	Compiled Variable Object Empty Replace:27.70runs/s ±0.10%	
	Compiled Variable Object 12 Char Replace:23.80runs/s ±25.26%	
	Compiled Variable Object 12 Char Replace Function:25.70runs/s ±19.49%	
	Compiled Capture Match:12.83runs/s ±0.19%	
	Compiled Capture Replace:24.45runs/s ±12.85%	
	Compiled Capture Replace with Capture:19.88runs/s ±23.27%	
	Compiled Capture Replace with Capture Function:19.18runs/s ±20.09%	
	Compiled Capture Replace with Upperase Capture Function:17.93runs/s ±9.80%	
	Uncompiled Match:12.37runs/s ±21.44%	
	Uncompiled Test:12.61runs/s ±19.80%	
	Uncompiled Empty Replace:27.48runs/s ±1.37%	
	Uncompiled 12 Char Replace:21.96runs/s ±23.02%	
	Uncompiled Object Match:13.82runs/s ±0.15%	
	Uncompiled Object Test:13.83runs/s ±0.16%	
	Uncompiled Object Empty Replace:27.36runs/s ±1.27%	
	Uncompiled Object 12 Char Replace:27.33runs/s ±1.43%	

Regular Expressions:	13.96runs/s	
Microtests of regular expressions (construction, matching, replace).	Compiled Object Empty Split: 0.35run	
	Compiled Object Char Split: 4.21runs/	
	Compiled Object Variable Split: 11.49	
	Compiled Match: 13.69runs/s ±0.28%	
	Compiled Test: 11.09runs/s ±20.53%	
	Compiled Empty Replace: 27.75runs/	
	Compiled 12 Char Replace: 27.73runs	
	Compiled Object Match: 13.73runs/s	
	Compiled Object Test: 13.20runs/s ±1	
	Compiled Object Empty Replace: 22.5	
	Compiled Object 12 Char Replace: 21	
	Compiled Object 12 Char Replace Fur	
	Compiled Variable Match: 9.93runs/s	
	Compiled Variable Test: 10.56runs/s	
	Compiled Variable Empty Replace: 23	
	Compiled Variable 12 Char Replace: 2	
	Compiled Variable Object Match: 11.	
	Compiled Variable Object Test: 10.94	
	Compiled Variable Object Empty Repl	
	Compiled Variable Object 12 Char Re	
	Compiled Variable Object 12 Char Re	
	Compiled Capture Match: 10.10runs/	
	Compiled Capture Replace: 18.81runs	
	Compiled Capture Replace with Capti	
	Compiled Capture Replace with Capti	
	Compiled Capture Replace with Uppe	
	Uncompiled Match: 11.51runs/s ±23.1	
	Uncompiled Test: 10.46runs/s ±18.46	
	Uncompiled Empty Replace: 22.37run	
	Uncompiled 12 Char Replace: 21.32ru	
	Uncompiled Object Match: 10.92runs	
	Uncompiled Object Test: 10.93runs/s	
	Uncompiled Object Empty Replace: 1	
	Uncompiled Object 12 Char Replace:	



String Parsing and Searching:	4.03runs/s	
Tests for parsing languages and searching strings.	Earley:12.68runs/s ±35.98%	
	Boyer:1.28runs/s ±36.19%	
Strings:	67.63runs/s	
Microtests of strings (concatenation, methods).	Concat String:182.62runs/s ±35.15%	
	Concat String Object:204.44runs/s ±29.54%	
	Concat String from charCode:257.24runs/s ±3.41%	
	Array String Join:207.72runs/s ±35.45%	
	String Split:18.55runs/s ±35.28%	
	String Split on Char:8.81runs/s ±26.83%	
	charAt:113.47runs/s ±26.28%	
	[Number]:204.74runs/s ±0.42%	
	charCodeAt:123.48runs/s ±0.27%	
	indexOf:84.34runs/s ±0.16%	
	lastIndexOf:68.67runs/s ±0.18%	
	slice:16.17runs/s ±15.37%	
	substr:17.74runs/s ±23.94%	
	substring:17.11runs/s ±25.73%	
	toLowerCase:186.25runs/s ±10.26%	
	toUpperCase:41.88runs/s ±14.86%	
	comparing:62.39runs/s ±21.58%	
Tag Cloud Creation:	27.32runs/s	
Convert a JSON structure into an HTML tag cloud.	Parse JSON:27.23runs/s ±32.68%	
	Make Tag Cloud:27.42runs/s ±22.89%	
Traversing Binary Trees:	27.65runs/s	
Moving through an object representation of a binary tree.	Binary Trees:27.65runs/s ±22.69%	
Trigonometric Calculation:	56.66runs/s	
Calculate values from hyperbolic and trigonometric functions.	Cordic:56.66runs/s ±25.80%	
Validate User Input:	24.75runs/s	
Test user input against a series of rules.	Validate Email Input:26.09runs/s ±36.48%	
	Validate Zipcode Input:23.48runs/s ±26.65%	

String Parsing and Searching:	3.94runs/s	
Tests for parsing languages and searching strings.	Earley: 12.43runs/s ±22.29%	
	Boyer: 1.25runs/s ±35.90%	
Strings:	67.35runs/s	
Microtests of strings (concatenation, methods).	Concat String: 177.25runs/s ±34.32%	
	Concat String Object: 194.52runs/s ±30.11%	
	Concat String from charCode: 277.63runs/s ±10.	
	Array String Join: 220.58runs/s ±43.44%	
	String Split: 18.91runs/s ±5.22%	
	String Split on Char: 8.23runs/s ±30.84%	
	charAt: 137.77runs/s ±1.09%	
	[Number]: 188.56runs/s ±19.67%	
	charCodeAt: 106.15runs/s ±26.52%	
	indexOf: 89.45runs/s ±0.06%	
	lastIndexOf: 67.04runs/s ±0.11%	
	slice: 18.53runs/s ±24.27%	
	substr: 19.39runs/s ±23.57%	
	substring: 14.65runs/s ±26.31%	
	toLowerCase: 147.26runs/s ±22.63%	
	toUpperCase: 44.36runs/s ±17.66%	
	comparing: 64.52runs/s ±34.67%	
Tag Cloud Creation:	25.87runs/s	
Convert a JSON structure into an HTML tag cloud.	Parse JSON: 27.19runs/s ±11.05%	
	Make Tag Cloud: 24.61runs/s ±4.78%	
Traversing Binary Trees:	29.36runs/s	
Moving through an object representation of a binary tree.	Binary Trees: 29.36runs/s ±22.31%	
Trigonometric Calculation:	54.77runs/s	
Calculate values from hyperbolic and trigonometric functions.	Cordic: 54.77runs/s ±25.08%	
Validate User Input:	24.94runs/s	
Test user input against a series of rules.	Validate Email Input: 26.42runs/s ±35.07%	
	Validate Zipcode Input: 23.53runs/s ±24.33%	



Benchmarks Desktop

GTKPerf

Teste 1	**Teste 2**	**Teste 3**	**Teste 4**
GtkPerf 0.40 - Starting testing: Mon Oct 17 10:43:45 2016	GtkPerf 0.40 - Starting testing: Mon Oct 17 10:45:37 2016	GtkPerf 0.40 - Starting testing: Mon Oct 17 10:46:49 2016	GtkPerf 0.40 - Starting testing: Mon Oct 17 10:48:28 2016
GtkEntry - time: 0,22	GtkEntry - time: 0,21	GtkEntry - time: 0,26	GtkEntry - time: 0,22
GtkComboBox - time: 6,15	GtkComboBox - time: 6,75	GtkComboBox - time: 6,32	GtkComboBox - time: 5,96
GtkComboBoxEntry - time: 4,40	GtkComboBoxEntry - time: 3,35	GtkComboBoxEntry - time: 4,43	GtkComboBoxEntry - time: 4,27
GtkSpinButton - time: 0,97	GtkSpinButton - time: 0,92	GtkSpinButton - time: 1,02	GtkSpinButton - time: 0,95
GtkProgressBar - time: 0,67	GtkProgressBar - time: 0,65	GtkProgressBar - time: 0,67	GtkProgressBar - time: 0,65
GtkToggleButton - time: 2,13	GtkToggleButton - time: 2,04	GtkToggleButton - time: 2,06	GtkToggleButton - time: 2,04
GtkCheckButton - time: 0,44	GtkCheckButton - time: 0,42	GtkCheckButton - time: 0,41	GtkCheckButton - time: 0,41
GtkRadioButton - time: 0,50	GtkRadioButton - time: 0,50	GtkRadioButton - time: 0,50	GtkRadioButton - time: 0,49
GtkTextView - Add text - time: 2,29	GtkTextView - Add text - time: 2,28	GtkTextView - Add text - time: 2,33	GtkTextView - Add text - time: 2,23
GtkTextView - Scroll - time: 0,07	GtkTextView - Scroll - time: 0,07	GtkTextView - Scroll - time: 0,07	GtkTextView - Scroll - time: 0,07
GtkDrawingArea - Lines - time: 3,64	GtkDrawingArea - Lines - time: 3,63	GtkDrawingArea - Lines - time: 3,66	GtkDrawingArea - Lines - time: 3,63
GtkDrawingArea - Circles - time: 6,06	GtkDrawingArea - Circles - time: 6,07	GtkDrawingArea - Circles - time: 6,11	GtkDrawingArea - Circles - time: 6,06
GtkDrawingArea - Text - time: 3,91	GtkDrawingArea - Text - time: 3,88	GtkDrawingArea - Text - time: 3,93	GtkDrawingArea - Text - time: 3,90
GtkDrawingArea - Pixbufs - time: 0,96	GtkDrawingArea - Pixbufs - time: 0,93	GtkDrawingArea - Pixbufs - time: 0,98	GtkDrawingArea - Pixbufs - time: 0,95
---	---	---	---
Total time: 32,43	Total time: 31,70	Total time: 32,78	Total time: 31,85

Hard Info

Teste 1	**Teste 2**	**Teste 3**	**Teste 4**
-CPU Blowfish-	**-CPU Blowfish-**	**-CPU Blowfish-**	**-CPU Blowfish-**
This MachineUnknown MHz89,013	This MachineUnknown MHz91,713	This MachineUnknown MHz93,094	This MachineUnknown MHz93,392
Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876862	Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876862	Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876862	Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876
PowerPC 740/750 (280.00MHz)(null)172.816713	PowerPC 740/750 (280.00MHz)(null)172.816713	PowerPC 740/750 (280.00MHz)(null)172.816713	PowerPC 740/750 (280.00MHz)(null)172.816713
-CPU CryptoHash-	**-CPU CryptoHash-**	**-CPU CryptoHash-**	**-CPU CryptoHash-**
This MachineUnknown MHz10,299	This MachineUnknown MHz10,410	This MachineUnknown MHz10,178	This MachineUnknown MHz10,171

Linux Raspbian | Windows 10 IoT | SPSS

Pronto | Calcula

110%

Totais

Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
6695,3	6800	6895,4	6928,4	6939,4	6861,9	6932,9	6856,4	6863,7	6866,7	6928,5	6851	6842,4	6836,7	6925,9	6915	6895,7	6907,7	6894,9	6840,9

Tempo aproximadamente de cada teste:	
1 minuto e 40 segundos	

Totais

Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
32,8	30,96	29,55	28,33	27,7	27,1	26,98	27,08	27,62	26,35	37,3	37,01	37,03	37,24	37,02	37,2	36,96	37,13	37,08	37,12

Tempo aproximadamente de cada teste:		
16 minutos		

Hard Info

Teste 1	Teste 2	Teste 3	Teste 4
-CPU Blowfish-	-CPU Blowfish-	-CPU Blowfish-	-CPU Blowfish-
This MachineUnknown MHz89,013	This MachineUnknown MHz91,713	This MachineUnknown MHz93,094	This MachineUnknown MHz93,392
Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876862	Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876862	Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876862	Intel(R) Celeron(R) M processor 1.50GHz(null)26.1876
PowerPC 740/750 (280.00MHz)(null)172.816713	PowerPC 740/750 (280.00MHz)(null)172.816713	PowerPC 740/750 (280.00MHz)(null)172.816713	PowerPC 740/750 (280.00MHz)(null)172.816713
-CPU CryptoHash-	-CPU CryptoHash-	-CPU CryptoHash-	-CPU CryptoHash-
This MachineUnknown MHz10,299	This MachineUnknown MHz10,410	This MachineUnknown MHz10,178	This MachineUnknown MHz10,171
-CPU Fibonacci-	-CPU Fibonacci-	-CPU Fibonacci-	-CPU Fibonacci-
This MachineUnknown MHz24,195	This MachineUnknown MHz24,117	This MachineUnknown MHz24,391	This MachineUnknown MHz24,212
Intel(R) Celeron(R) M processor 1.50GHz(null)8.1375674	Intel(R) Celeron(R) M processor 1.50GHz(null)8.1375674	Intel(R) Celeron(R) M processor 1.50GHz(null)8.1375674	Intel(R) Celeron(R) M processor 1.50GHz(null)8.13756
PowerPC 740/750 (280.00MHz)(null)58.07682	PowerPC 740/750 (280.00MHz)(null)58.07682	PowerPC 740/750 (280.00MHz)(null)58.07682	PowerPC 740/750 (280.00MHz)(null)58.07682
-CPU N-Queens-	-CPU N-Queens-	-CPU N-Queens-	-CPU N-Queens-
This MachineUnknown MHz76,597	This MachineUnknown MHz76,773	This MachineUnknown MHz77,483	This MachineUnknown MHz76,935
-FPU FFT-	-FPU FFT-	-FPU FFT-	-FPU FFT-
This MachineUnknown MHz111,469	This MachineUnknown MHz111,369	This MachineUnknown MHz110,900	This MachineUnknown MHz109,970
-FPU Raytracing-	-FPU Raytracing-	-FPU Raytracing-	-FPU Raytracing-
This MachineUnknown MHz129,677	This MachineUnknown MHz130,564	This MachineUnknown MHz131,541	This MachineUnknown MHz132,283
Intel(R) Celeron(R) M processor 1.50GHz(null)40.8816714	Intel(R) Celeron(R) M processor 1.50GHz(null)40.8816714	Intel(R) Celeron(R) M processor 1.50GHz(null)40.8816714	Intel(R) Celeron(R) M processor 1.50GHz(null)40.8816
PowerPC 740/750 (280.00MHz)(null)161.312647	PowerPC 740/750 (280.00MHz)(null)161.312647	PowerPC 740/750 (280.00MHz)(null)161.312647	PowerPC 740/750 (280.00MHz)(null)161.312647

Totais	-CPU Blowfish-																		
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
89,013	91,713	93,094	93,392	92,731	93,654	92,683	93,391	92,633	92,463	93,081	93,646	62,29	61,845	63,416	61,819	62,546	62,272	62,664	62,254

Totais	-CPU CryptoHash-																		
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
10,299	10,41	10,178	10,171	10,127	10,236	10,15	10,201	10,152	10,271	10,178	10,212	15,343	15,241	14,501	15,363	14,711	15,377	15,367	15,304

Totais	-CPU Fibonacci-																		
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
24,195	24,117	24,391	24,212	24,394	24,202	24,465	24,411	24,486	24,189	24,116	24,453	16,11	16,394	16,91	16,349	16,911	16,783	16,251	16,112

Totais	-CPU N-Queens-																		
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
76,597	76,773	77,483	76,935	77,951	77,707	76,995	77,606	77,475	77,04	77,71	77,743	51,731	53,165	52,234	52,529	51,437	51,78	52,524	51,523

Totais	-FPU FFT-																		
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
111,469	111,369	110,9	109,97	110,314	110,835	112,015	110,961	111,49	111,481	111,427	111,364	75,705	76,473	77,922	75,351	75,215	75,633	76,542	76,042

Totais	-FPU Raytracing-																		
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
129,677	130,564	131,541	132,283	130,611	132,393	130,971	132,576	132,673	132,308	132,165	132,597	87,613	87,541	88,988	88,591	88,803	89,413	89,795	87,559

Tempo aproximadamente de cada teste:
8 minutos
Observação: 8 minutos é o teste completo envolvendo todos os 6 benchmarks do Hard Info

Totais																		
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7	Teste 8	Teste 9	Teste 10	Teste 11	Teste 12	Teste 13	Teste 14	Teste 15	Teste 16	Teste 17	Teste 18	Teste 19	Teste 20
32,43	31,7	32,78	31,85	32,35	32,31	31,87	31,99	32,38	32,43	32,14	32,29	32,34	33,15	32,03	32,44	32,59	32,1	31,65	31,78

Tempo aproximadamente de cada teste:
32 segundos

APÊNDICE B – TESTES SPSS

Resumo de processamento do caso

		Casos					
		Válido		Ausente		Total	
	Sistema Operacional	N	Porcentagem	N	Porcentagem	N	Porcentagem
Wirple	Linux Raspbian	20	100,0%	0	0,0%	20	100,0%
	Windows 10 IoT	20	100,0%	0	0,0%	20	100,0%

Descritivos

	Sistema Operacional			Estatística	Erro Padrão
Wirple	Linux Raspbian	Média		21,90	,584
		95% Intervalo de Confiança para Média	Limite inferior	20,68	
			Limite superior	23,12	
		5% da média aparada		22,22	
		Mediana		23,00	
		Variância		6,832	
		Desvio Padrão		2,614	
		Mínimo		15	
		Máximo		23	
		Intervalo		8	
		Intervalo interquartil		0	
		Assimetria		-2,276	,512
		Curtose		3,872	,992
	Windows 10 IoT	Média		39,40	1,409
		95% Intervalo de Confiança para Média	Limite inferior	36,45	
			Limite superior	42,35	
		5% da média aparada		39,94	
		Mediana		43,00	
		Variância		39,726	
		Desvio Padrão		6,303	
		Mínimo		26	
		Máximo		43	
		Intervalo		17	
		Intervalo interquartil		5	
		Assimetria		-1,449	,512
		Curtose		,364	,992

Testes de Normalidade

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Sistema Operacional	Estatística	df	Sig.	Estatística	df	Sig.
Wirple	Linux Raspbian	,463	20	,000	,474	20	,000
	Windows 10 IoT	,416	20	,000	,604	20	,000

a. Correlação de Significância de Lilliefors

Resumo de processamento do caso

		Casos					
		Válido		Ausente		Total	
	Sistema Operacional	N	Porcentagem	N	Porcentagem	N	Porcentagem
Peacekeeper	Linux Raspbian	20	100,0%	0	0,0%	20	100,0%
	Windows 10 IoT	20	100,0%	0	0,0%	20	100,0%

Descritivos

	Sistema Operacional			Estatística	Erro Padrão
Peacekeeper	Linux Raspbian	Média		193,75	6,243
		95% Intervalo de Confiança para Média	Limite inferior	180,68	
			Limite superior	206,82	
		5% da média aparada		192,39	
		Mediana		191,00	
		Variância		779,566	
		Desvio Padrão		27,921	
		Mínimo		166	
		Máximo		246	
		Intervalo		80	
		Intervalo interquartil		44	
		Assimetria		,750	,512
		Curtose		-,587	,992
	Windows 10 IoT	Média		190,55	,806
		95% Intervalo de Confiança para Média	Limite inferior	188,86	
			Limite superior	192,24	
		5% da média aparada		190,72	
		Mediana		191,50	
		Variância		12,997	
		Desvio Padrão		3,605	
		Mínimo		182	
		Máximo		196	
		Intervalo		14	
		Intervalo interquartil		5	
		Assimetria		-,739	,512
		Curtose		,453	,992

Testes de Normalidade

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Sistema Operacional	Estatística	df	Sig.	Estatística	df	Sig.
Peacekeeper	Linux Raspbian	,192	20	,051	,859	20	,008
	Windows 10 IoT	,156	20	,200*	,946	20	,308

*. Este é um limite inferior da significância verdadeira.

a. Correlação de Significância de Lilliefors

Resumo de processamento do caso

		Casos					
		Válido		Ausente		Total	
	Sistema Operacional	N	Porcentagem	N	Porcentagem	N	Porcentagem
SpeedBattle	Linux Raspbian	20	100,0%	0	0,0%	20	100,0%
	Windows 10 IoT	20	100,0%	0	0,0%	20	100,0%

Descritivos

	Sistema Operacional			Estatística	Erro Padrão
SpeedBattle	Linux Raspbian	Média		10,8595	,03801
		95% Intervalo de Confiança para Média	Limite inferior	10,7800	
			Limite superior	10,9390	
		5% da média aparada		10,8700	
		Mediana		10,8800	
		Variância		,029	
		Desvio Padrão		,16997	
		Mínimo		10,44	
		Máximo		11,09	
		Intervalo		,65	
		Intervalo interquartil		,23	
		Assimetria		-,805	,512
		Curtose		,523	,992
	Windows 10 IoT	Média		46,2000	,50331
		95% Intervalo de Confiança para Média	Limite inferior	45,1466	
			Limite superior	47,2534	
		5% da média aparada		46,3133	
		Mediana		46,4800	
		Variância		5,066	
		Desvio Padrão		2,25085	
		Mínimo		41,24	
		Máximo		49,12	
		Intervalo		7,88	
		Intervalo interquartil		3,26	
		Assimetria		-,747	,512
		Curtose		-,232	,992

Testes de Normalidade

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Sistema Operacional	Estatística	df	Sig.	Estatística	df	Sig.
SpeedBattle	Linux Raspbian	,123	20	,200*	,947	20	,327
	Windows 10 IoT	,133	20	,200*	,933	20	,177

*. Este é um limite inferior da significância verdadeira.

a. Correlação de Significância de Lilliefors

Resumo de processamento do caso

		Casos					
		Válido		Ausente		Total	
	Sistema Operacional	N	Porcentagem	N	Porcentagem	N	Porcentagem
SunSpider	Linux Raspbian	20	100,0%	0	0,0%	20	100,0%
	Windows 10 IoT	20	100,0%	0	0,0%	20	100,0%

Descritivos

	Sistema Operacional			Estatística	Erro Padrão
SunSpider	Linux Raspbian	Média		6873,9400	12,83412
		95% Intervalo de Confiança para Média	Limite inferior	6847,0779	
			Limite superior	6900,8021	
		5% da média aparada		6880,2278	
		Mediana		6880,8000	
		Variância		3294,293	
		Desvio Padrão		57,39593	
		Mínimo		6695,30	
		Máximo		6939,40	
		Intervalo		244,10	
		Intervalo interquartil		78,63	
		Assimetria		-1,595	,512
		Curtose		3,869	,992
	Windows 10 IoT	Média		2891,4700	40,87217
		95% Intervalo de Confiança para Média	Limite inferior	2805,9236	
			Limite superior	2977,0164	
		5% da média aparada		2896,9000	
		Mediana		2925,7500	
		Variância		33410,687	
		Desvio Padrão		182,78591	
		Mínimo		2501,20	
		Máximo		3184,00	
		Intervalo		682,80	
		Intervalo interquartil		230,33	
		Assimetria		-,617	,512
		Curtose		-,356	,992

Testes de Normalidade

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Sistema Operacional	Estatística	df	Sig.	Estatística	df	Sig.
SunSpider	Linux Raspbian	,158	20	,200*	,860	20	,008
	Windows 10 IoT	,216	20	,015	,924	20	,120

*. Este é um limite inferior da significância verdadeira.

a. Correlação de Significância de Lilliefors

Resumo de processamento do caso

		Casos					
		Válido		Ausente		Total	
	Sistema Operacional	N	Porcentagem	N	Porcentagem	N	Porcentagem
Dromaeo	Linux Raspbian	20	100,0%	0	0,0%	20	100,0%
	Windows 10 IoT	20	100,0%	0	0,0%	20	100,0%

Descritivos

	Sistema Operacional			Estatística	Erro Padrão
Dromaeo	Linux Raspbian	Média		32,7780	1,04281
		95% Intervalo de Confiança para Média	Limite inferior	30,5954	
			Limite superior	34,9606	
		5% da média aparada		32,8839	
		Mediana		34,8800	
		Variância		21,749	
		Desvio Padrão		4,66357	
		Mínimo		26,35	
		Máximo		37,30	
		Intervalo		10,95	
		Intervalo interquartil		9,47	
		Assimetria		-,237	,512
		Curtose		-1,959	,992
	Windows 10 IoT	Média		38,9280	,11861
		95% Intervalo de Confiança para Média	Limite inferior	38,6797	
			Limite superior	39,1763	
		5% da média aparada		38,9244	
		Mediana		38,7500	
		Variância		,281	
		Desvio Padrão		,53046	
		Mínimo		38,38	
		Máximo		39,54	
		Intervalo		1,16	
		Intervalo interquartil		1,16	
		Assimetria		,242	,512
		Curtose		-1,922	,992

Testes de Normalidade

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Sistema Operacional	Estatística	df	Sig.	Estatística	df	Sig.
Dromaeo	Linux Raspbian	,315	20	,000	,762	20	,000
	Windows 10 IoT	,276	20	,000	,746	20	,000

a. Correlação de Significância de Lilliefors

Teste Mann-Whitney

Classificações

	Sistema Operacional	N	Postos de média	Soma de Classificações
Wirple	Linux Raspbian	20	10,50	210,00
	Windows 10 IoT	20	30,50	610,00
	Total	40		

Estatísticas de teste[a]

	Wirple
U de Mann-Whitney	,000
Wilcoxon W	210,000
Z	-5,725
Significância Sig. (2 extremidades)	,000
Sig exata [2*(Sig. de 1 extremidade)]	,000[b]

a. Variável de Agrupamento: Sistema Operacional

b. Não corrigido para vínculos.

Teste Mann-Whitney

Classificações

	Sistema Operacional	N	Postos de média	Soma de Classificações
Dromaeo	Linux Raspbian	20	10,50	210,00
	Windows 10 IoT	20	30,50	610,00
	Total	40		

Estatísticas de teste[a]

	Dromaeo
U de Mann-Whitney	,000
Wilcoxon W	210,000
Z	-5,448
Significância Sig. (2 extremidades)	,000
Sig exata [2*(Sig. de 1 extremidade)]	,000[b]

a. Variável de Agrupamento: Sistema Operacional

b. Não corrigido para vínculos.

Teste-T

Estatísticas de grupo

	Sistema Operacional	N	Média	Desvio Padrão	Erro padrão da média
Peacekeeper	Linux Raspbian	20	193,75	27,921	6,243
	Windows 10 IoT	20	190,55	3,605	,806

Teste de amostras independentes

		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
Peacekeeper	Variâncias iguais assumidas	32,271	,000	,508	38	,614	3,200	6,295	-9,544	15,944
	Variâncias iguais não assumidas			,508	19,633	,617	3,200	6,295	-9,947	16,347

Teste-T

Estatísticas de grupo

	Sistema Operacional	N	Média	Desvio Padrão	Erro padrão da média
SpeedBattle	Linux Raspbian	20	10,8595	,16997	,03801
	Windows 10 IoT	20	46,2000	2,25085	,50331

Teste de amostras independentes

		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
SpeedBattle	Variâncias iguais assumidas	30,894	,000	-70,017	38	,000	-35,34050	,50474	-36,36229	-34,31871
	Variâncias iguais não assumidas			-70,017	19,217	,000	-35,34050	,50474	-36,39612	-34,28488

Teste-T

Estatísticas de grupo

	Sistema Operacional	N	Média	Desvio Padrão	Erro padrão da média
SunSpider	Linux Raspbian	20	6873,9400	57,39593	12,83412
	Windows 10 IoT	20	2891,4700	182,78591	40,87217

Teste de amostras independentes

		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
SunSpider	Variâncias iguais assumidas	23,770	,000	92,962	38	,000	3982,47000	42,83981	3895,74535	4069,19465
	Variâncias iguais não assumidas			92,962	22,711	,000	3982,47000	42,83981	3893,78661	4071,15339

APENDICE C – ARTIGO CIENTÍFICO

ANÁLISE DE DESEMPENHO UTILIZANDO BENCHMARKS ENTRE LINUX RASPBIAN E WINDOWS 10 IOT COM USO DO RASPBERRY PI

Henrique De Noni1, Sérgio Coral1

1Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias (UNACET) - Universidade do Extremo Sul Catarinense (UNESC) – Av. Universitária, 1105 – Bairro Universitário – Criciúma – SC - Brasil

henriquenoni@hotmail.com, sergiocoral@unesc.net

Abstract. *In this work a performance test was performed between the Linux operating system Raspbian and Windows 10 IoT to evaluate which operating system is best adapted to the use of the Raspberry Pi platform. To perform the tests we used some online benchmarks and some desktop benchmarks. In order to process and analyze the data of the tests and to make the comparatives was used tables and graphs of Excel and the software SPSS of IBM. With the tests carried out, Windows 10 IoT was able to adapt better to Raspberry due to the better means compared to Linux Raspbian. Although Windows 10 IoT has fared better, both operating systems show some instability when using it.*

Keywords: *Raspberry Pi. Linux Raspbian. Windows 10 IoT. Benchmarking.*

Resumo. *Neste trabalho realizou-se um teste de desempenho entre o sistema operacional Linux Raspbian e o Windows 10 IoT para avaliar qual sistema operacional se adapta melhor ao uso da plataforma Raspberry Pi. Para realizar os testes foram utilizados alguns benchmarks online e alguns benchmarks desktop. Para processar e analisar os dados dos testes e fazer os comparativos foi utilizado tabelas e gráficos do Excel e o software SPSS da IBM. Com os testes realizados o Windows 10 IoT conseguiu se adaptar melhor ao Raspberry devido as médias melhores em relação ao Linux Raspbian. Apesar de o Windows 10 IoT ter se saído melhor, em ambos os sistemas operacionais apresentaram um certa instabilidade ao usa-lo.*

Palavras-chave: *Raspberry Pi. Linux Raspbian. Windows 10 IoT. Benchmarking.*

1. Introdução

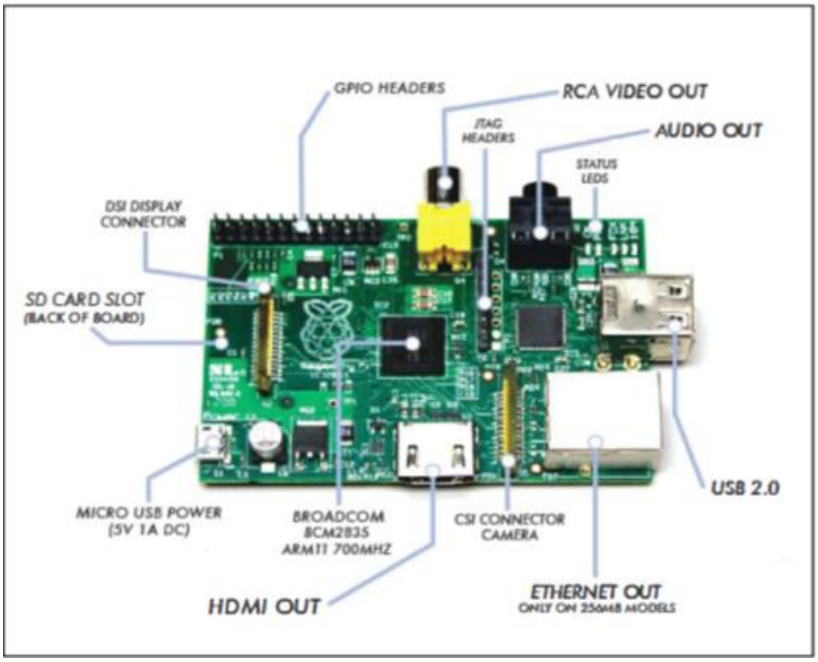
Os setores de TI e da eletrônica, são dois setores que estão aumentando muito espaço nestes últimos anos, essas duas áreas se completam e se estendem em muitos tipos de atividades, na área de desenvolvimento, por exemplo, muitas vezes é preciso soluções rápidas e de baixo custo, o ideal seria plataformas eletrônicas de prototipação pequenas e com pouca programação embutidas, uma dessas plataformas de prototipação é o Raspberry Pi.

O *kernel* Linux é um sistema operacional desenvolvido por milhares de programadores e colaboradores de todo o mundo, com pacotes estáveis para todos os tipos de usuários e até para empresas, ele é o sistema operacional que está sendo cada vez mais encontrado em desktops e que possui o núcleo mais usado em servidores de grande porte no mundo por ser muito estável e livre de ameaças como vírus.
O Windows é um sistema operacional que opera através de interfaces gráficas em formas de janelas, daí o nome Windows, fundado por Bill Gates e Paul Allen, sua última versão o Windows 10 é multiplataforma, ou seja, funciona em computadores, celulares e tablets, o diferencial é que ele funciona em plataformas menores como o Raspberry Pi.
Para avaliar o desempenho de qualquer sistema é necessário um software de *benchmarking*, ou seja, são *softwares* específicos para avaliar o desempenho de determinados sistemas, para realizar a metodologia de benchmarking é necessário definir as métricas de desempenho para então poder ser avaliado.

2. Raspberry Pi

O Raspberry Pi ficou popularmente conhecido no mundo no ano de 2011, ele é uma placa de prototipação de mais ou menos do tamanho de um cartão de crédito que custa cerca de 35 dólares baseada na arquitetura de processadores *Advanced RISC Machines* (ARM), esta plataforma foi bem aceita por desenvolvedores e por profissionais da área de eletrônica.
A peça chave do Raspberry Pi é um processador da Broadcom: BCM 2836, sua arquitetura de 32 bits é baseada num núcleo ARM 11, modelo Cortex-A7 com 4 núcleos, é o mesmo processador que encontraríamos num Iphone 3G, este processador possui 700Mhz de processamento e 512MB de memória, entretanto os modelos mais recentes já vem com 1GB.

Figura 1 – Raspberry Pi



3. Linux Raspbian

O termo Linux é designado para sistemas operacionais que utilizam o kernel Linux, que foi desenvolvido por Linus Torvalds com base no sistema operacional *Minix,* sua licença é livre, ou seja, é possível usa-la, estuda-la, modifica-la livremente de acordo com os termos regidos por ela.
O Raspbian é um sistema operacional gratuito baseado no Linux *Debian* e é um sistema operacional adaptado para o Raspberry Pi, ele vem com mais de 35.000 pacotes e softwares pré-compilados. O sistema ainda está em desenvolvimento para aperfeiçoar ainda mais o Raspberry Pi e ele foi desenvolvido por um grupo de fãs e admiradores do Raspberry. A versão atual do Raspbian é a versão Jessie.
O sistema operacional é a distribuição ideal para quem possui menos conhecimentos dos sistemas Linux, pois já é um sistema que quando instalado vem praticamente pronto. O Raspbian é um sistema quase completo, já que vem com diversas aplicações pré-instaladas, os *drivers* mais importantes instalados e ferramentas para facilitar algumas configurações necessárias. Muitas aplicações e módulos dedicados à programação já vêm incluídos na imagem do SO, bastando apenas iniciar o sistema para acessá-los.
O Raspbian foi o sistema operacional escolhido pela *Raspberry Pi Foundation* como sistema padrão do Raspberry, pois inclui um dos gerenciadores de pacotes mais flexíveis. O *Debian* e o *Ubuntu* não suportam a arquitetura ARM encontrados no processador do Raspberry Pi, mas a *Raspberry Pi Foundation* modificou o *kernel* do *Debian* para acomodar o processador ARM do Raspberry Pi.

4. Windows 10 IoT

O Windows é um sistema operacional desenvolvido pela Microsoft, ele está presente em mais de 90% dos computadores no mundo, desenvolvido por Bill Gates e Paul Allen, sua interface gráfica é toda em janelas, dai o nome Windows.
A última versão do Windows, chamada de Windows 10 foi lançada em 2015, o sistema operacional vem com uma interface totalmente modificada, dando mais enfoque a usuários mais antigos, o menu iniciar volta diferente e a interface do sistema inteiro volta mais intuitiva ao usuário, também tem a chegada de um novo navegador para a Internet o *Microsoft Edge.*
O Windows 10 também funciona em plataformas menores e de pouco processamento, a Microsoft chama de Windows 10 *Internet Of Things* (IoT – Internet das coisas), esse SO funciona em dispositivos que possuem monitor ou não, ele roda no Raspberry Pi e também aceita os recursos do arduino.
Para instalar o Windows 10 IoT nestes dispositivos precisam ter os seguintes requisitos de *hardware*: No mínimo 256 MB de memória RAM e 2 GB de armazenamento, o recomendado é 512 MB de memória, e o processador precisa ter no mínimo 400 MHz de processamento.

5. Benchmarks

A metodologia de *benchmarks* é uma das técnicas mais utilizadas para avaliar métricas de qualidade nas empresas, essa metodologia procura melhorar os processos

empresariais e aumentar a competição para com outras empresas, onde essa metodologia traz vários avanços para dentro das empresas.
Há várias definições para a metodologia de *benchmarking*. Alguns tipos de autores definem que o *benchmarking* vem da palavra japonesa *dantotsu* que significa os processos de pesquisa e exploração dos pontos fortes das empresas concorrentes a fim de melhora-los.
O propósito da metodologia é confrontar dados, processos, procedimentos e situações dentro de uma organização, avaliar estes dados e compara-los com experiências de outras empresas promovendo seu crescimento. Esta metodologia serve para ver o que outras empresas fazem corretamente e o que traz resultados positivos e ao mesmo tempo identifica problemas ou defeitos em seus processos de trabalho e em seus concorrentes.
Serão avaliados o desempenho de dois sistemas operacionais com uso do Raspberry Pi, o Linux Raspbian e o Windows 10 IoT. Para avaliar qual sistema que se adapta melhor ao uso do Raspberry será avaliada uma série de métricas de desempenho dentro de cada sistema operacional.
Foi avaliado o gerenciamento de processamento que avaliou como o Raspberry processa as informações dentro de cada sistema operacional, como o processador trabalha com cálculos matemáticos e renderização.
No desempenho de vídeo foi demonstrado como ele se comporta ao rodar jogos e analisar gráficos, no desempenho dos navegadores foi demonstrado como o Raspberry faz criptografia e descriptografia dos dados, processamento de arrays e de código fonte.
Para fazer estas análises fora, instalados vários softwares de *benchmarks* nos dois sistemas operacionais, e através das análises destes softwares foram comparadas as métricas através de um método estatístico para então dizer qual sistema que se adapta melhor ao uso do Raspberry Pi.

6. Resultados Obtidos

Para a execução de todos os testes foi utilizado um Raspberry Pi 2 Modelo B com 1GB de memória RAM, com um processador Broadcom BCM2836 de 900Mhz e possui uma GPU Broadcom VideoCore IV.
Os *benchmarks online* utilizados foram o Wirple, Peacekeeper, SpeedBattle, SunSpider e Dromaeo. Os *benchmarks desktop* foram utilizados o Gtkperf e o HardInfo. No Linux Raspbian foram testados todos os *benchmarks*, e no Windows 10 IoT apenas os *benchmarks online*. Cada *benchmark* foi executado 20 vezes a fim de conseguir uma base sólida de testes.
Todos os resultados dos testes de *benchmarking* foram armazenados em uma planilha do Microsoft Office Excel. Para realizar a análise, processamento e comparação dos dados foi utilizado o *software Statistical Package for Social Sciences* (SPSS) versão 22 da empresa IBM.

Figura 2 – Média dos *benchmarkings online*

SPSS

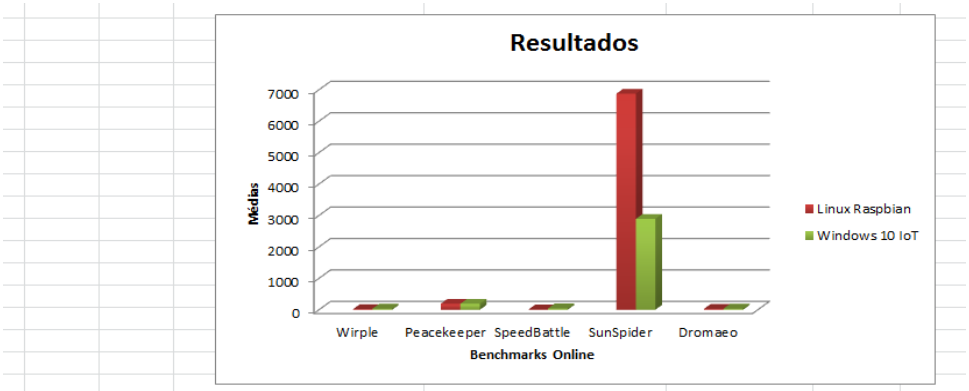
Benchmarks Online	Linux Raspian	Windows 10 IoT	Distribuição	Valor p
Wirple	21,90 +- 2,61	39,40 +- 6,30	Não normal	< 0,001
Peacekeeper	193,75 +- 27,92	190,55 +- 3,60	Normal	0,617
SpeedBattle	10,85 +- 0,16	46,20 +- 2,25	Normal	< 0,001
SunSpider	6873,94 +- 57,39	2891,47 +- 182,78	Normal	< 0,001
Dromaeo	32,74 +- 4,66	38,92 +- 0,53	Não normal	< 0,001
	Valores: Média e Desvio Padrão	Valores: Média e Desvio Padrão		Valor p= probabilidade

Médias Linux Raspbian	
Benchmark	Linux Raspbian
Wirple	21,90
Peacekeeper	193,75
SpeedBattle	10,85
SunSpider	6873,94
Dromaeo	32,77

Médias Windows 10 IoT	
Benchmark	Windows 10 IoT
Wirple	39,40
Peacekeeper	190,55
SpeedBattle	46,20
SunSpider	2891,47
Dromaeo	38,92

O Windows 10 IoT não suportava os *benchmarks desktop*, por causa da incompatibilidade de extensões de arquivos, foi realizado os testes dos programas *desktop* apenas no Linux Raspbian. Se no Windows 10 IoT funcionasse, poderia ser feito testes mais detalhados, e ainda comparados os *benchmarks online* com os *benchmarks desktop*.
De acordo com a estatística, as médias dos *benchmarks* Wirple, SpeedBattle e Dromaeo, se adaptam melhor ao sistema operacional Windows 10 IoT utilizando o Raspberry Pi. Enquanto que o *benchmark* SunSpider se adapta melhor ao Linux Raspbian e o *benchmark* Peacekeeper por ter resultados com médias semelhantes nos dois sistemas operacionais, ele é adequado para se usar nos dois sistemas operacionais.

Figura 3 – Gráfico com os resultados de cada *benchmarking online*



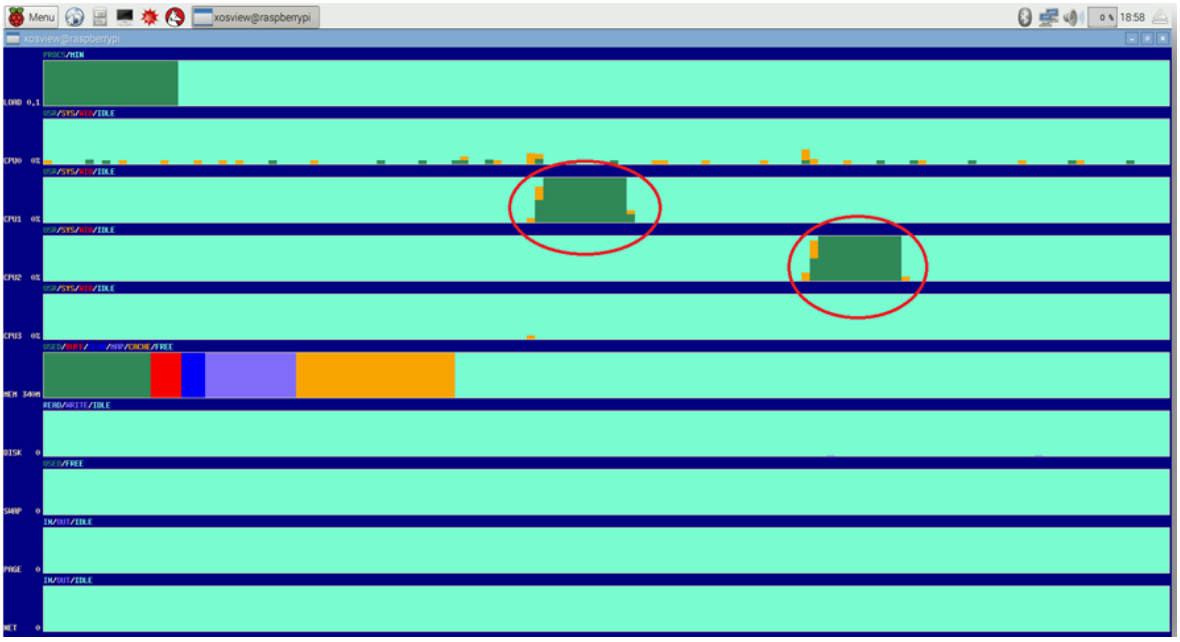
Resultado Final:

Benchmark	Sistema Operacional Adequado
Wirple	Windows 10 IoT
Peacekeeper	Equivalentes
SpeedBattle	Windows 10 IoT
SunSpider	Linux Raspbian
Dromaeo	Windows 10 IoT

Então dos *benchmarks online* o sistema operacional que se adapta melhor ao uso do Raspberry Pi 2 Modelo B é o Windows 10 IoT.

Encontrou-se certa dificuldade em encontrar *benchmarks* apropriados e com testes mais detalhados em ambos os sistemas operacionais, isso porque os *benchmarks* utilizados neste projeto eram gratuitos, e os *benchmarks* considerados melhores eram pagos e licenciados.
O Windows 10 IoT se adaptou melhor aos *benchmarks online* do que o Linux Raspbian que é o sistema nativo do Raspberry Pi, apesar de que os *benchmarks online* do Windows 10 IoT foram compilados remotamente através de um navegador remoto, e não com o navegador nativo do Windows, pois nem mesmo tinha instado no sistema operacional.
Durante todo o projeto de testes de *benchmarking* foi percebido alguns problemas em ambos os sistemas operacionais, tais como aquecimento do Raspberry Pi, foi notado que a placa esquentava em alguns momentos. Para solucionar isso foi acoplado nele um dissipador de calor, mas mesmo assim a placa ainda chegava a esquentar algumas vezes.
Em alguns momentos ocorriam alguns travamentos no sistema operacional, mas não foi encontrada a causa real desse problema, algumas causas possíveis para este problema seria o aquecimento mencionado anteriormente.
Na figura 4 mostra um *benchmark* que foi instalado no Linux que acompanha o sistema operacional em tempo real, este *benchmark* chamado de Xosview, não faz análise de nenhum componente do sistema operacional apenas mostra o desempenho do sistema em tempo real, esta imagem mostra o sistema sem rodar nenhum aplicativo apenas com o sistema iniciado e de vez em quando aconteciam alguns picos de processamento, ou seja, seu processador oscilava sem rodar nada no sistema operacional. Isto poderia acontecer em decorrência de alguma incompatibilidade na arquitetura ARM de seu processador.

Figura 4 – *Benchmark* Xosview



Na análise de desempenho de qualquer tipo de sistema operacional ou *hardware* é muito importante para descobrirem-se falhas em suas estruturas e definirem-se os pontos que

devem ser melhorados dentro de cada sistema ou partes específicas de cada *hardware*. Neste projeto foi pensado que o sistema operacional que ia se adaptar melhor ao Raspberry seria seu sistema nativo Linux Raspbian, mas de acordo com os resultados obtidos o Windows 10 IoT foi mais adequado. Foi detectado que o Rapsberry sofria alguns travamentos em ambos os sistemas operacionais, mas não foi identificado o real motivo de isso acontecer.

Referências

ALENCAR, Marcelo Sampaio de. A História do Linux. 2005. 10 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Campina Grande, Paraíba, 2005.

GOMES, Luiz Eduardo de Mello. BENCHMARKING E APRENDIZAGEM ORGANIZACIONAL Estudo de Caso na Companhia de Processamento de Dados de Minas Gerais - Prodemge. 2001. 125 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2001.

GUPTA, M. Surya Deekshith; PATCHAVA, Vamsikrishna; MENEZES, V Irginia. Healthcare based on loT using Raspberry Pi. In: INTERNATIONAL CONFERENCE ON GREEN COMPUTING AND INTERNET OF THINGS (ICGCLOT), 1., 2015, India. Healthcare based on loT using Raspberry Pi. India: Isbn, 2015. p. 796 - 799. Disponível em: <5 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7380571>. Acesso em: 10 abr. 2016.

MENEGUELLI, Marcelle Fernandes et al. BENCHMARKING: FERRAMENTA A SERVIÇO DA INOVAÇÃO. Revista Eletrônica da Faculdade Metodista Granbery, Juiz de Fora, v. 3, p.1-16, 01 jul. 2007. Semestral.

MICROSOFT (United States) (Org.). Learn about Windows 10 IoT Core: Why Windows 10 IoT Core?. 2016. Disponível em: <https://developer.microsoft.com/en-us/windows/iot/iotcore>. Acesso em: 22 maio 2016

MICROSOFT (United States) (Org.). Uma história do Windows. 2015. Disponível em: <http://windows.microsoft.com/pt-br/windows/history#T1=era0>. Acesso em: 21 maio 2016.

PEREIRA, Renata I. S.; JUCÁ, Sandro C. S.. Sistema Embarcado Linux baseado em Raspberry Pi para gravação online de Microcontroladores PIC. 2015. 6 f. Monografia (Especialização) - Curso de Engenharia Elétrica, Universidade Federal do Ceará, Fortaleza, 2015

RASPBERRY (United States). Welcome to Raspbian. 2016. Disponível em: <https://www.raspbian.org>. Acesso em: 03 maio 2016.

RICHARDSON, Matt; WALLACE, Shawn. Getting Started with Raspberry Pi. Sebastopol: O'reilly, 2012. 177 p.

SABER ELETRÔNICA: Industrial. São Paulo: Saber, mar. abr. 2013. Mensal. 2013. Disponível em: <http://www.revistapcecia.com.br/download/SE468_webS2.pdf>. Acesso em: 27 mar. 2016.

WARNER, Timothy L.. Hacking Raspberry Pi. Usa: Que, 2014. 523 p.