



UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

JOÃO VICTOR MOTA DO NASCIMENTO

ANÁLISE DA IMPLEMENTAÇÃO DE REDES DE SENsoRES VISUAIS SEM FIO
UTILIZANDO O RASPBERRY PI

FEIRA DE SANTANA
2017

JOÃO VICTOR MOTA DO NASCIMENTO

ANÁLISE DA IMPLEMENTAÇÃO DE REDES DE SENsoRES VISUAIS SEM FIO
UTILIZANDO O RASPBERRY PI

Trabalho de Conclusão de Curso apresentado
ao Colegiado do curso de Engenharia de
Computação como requisito parcial para
obtenção do grau de Bacharel em Engenharia
de Computação pela Universidade Estadual
de Feira de Santana.

Orientador: Dr. Daniel Gouveia Costa

FEIRA DE SANTANA
2017

Dedico esta monografia aos meus pais.

AGRADECIMENTOS

Agradeço à minha mãe Tânia por ser sempre forte e me apoiar nos momentos mais difíceis. Ao meu pai João (in memoriam) por ter feito o possível e o impossível para nos proporcionar sempre o melhor.

Às minhas avós Maria(in memoriam) e Loura(in memoriam) por seus cuidados e comprometimentos com a família. Meu avô Daniel(in memoriam), pelo seu carisma infinito. Meu avô João(in memoriam), que apesar de não tê-lo conhecido, o tenho como fonte de inspiração por conhecer sua história.

Minha companheira Natália, por estar ao meu lado em todos os momentos no desenrolar desta graduação, me incentivando e dando conselhos que jamais poderiam partir de mim.

Aos meus amigos da graduação, especificamente à equipe Untitled.c(pp). Muitas oportunidades surgiram e surgirão apenas por conhecer Iago, Joel e Fábio. Vocês são peças muito importantes na minha vida pessoal e profissional.

Ao meu orientador Daniel, por sua paciência e capacidade de achar tempo onde ninguém mais observa. Quando eu crescer quero ser igual a você.

Todos os professores que compõem o curso de Engenharia de Computação da UEFS.

Aos demais amigos que conheci durante o curso.

LISTA DE FIGURAS

Figura 1 Cobertura em RSSFs e RSVSFs	10
Figura 2 Raspberry Pi 2 Modelo B	12
Figura 3 Comparaçao de custo vs. Velocidade do processador	13
Figura 4 Módulo de câmera versão 1.3	14
Figura 5 Criptografia de chave simétrica	17
Figura 6 Criptografia de chave pública	17
Figura 7 Circuito INA219	18
Figura 8 Setup de medição	23
Figura 9 Alimentação do Raspberry Pi no INA219	23
Figura 10 Medição de inicialização	24
Figura 11 Medição ativada pelo Raspberry Pi	24
Figura 12 Fedberry - Diagrama de potência de inicialização	31
Figura 13 Fedberry - Diagrama de energia de inicialização	32
Figura 14 ArchLinux ARM - Diagrama de potência de inicialização	33
Figura 15 ArchLinux ARM - Diagrama de energia de inicialização	33
Figura 16 Raspbian Lite - Diagrama de potência de inicialização	34

Figura 17 Raspbian Lite - Diagrama de energia de inicialização	34
Figura 18 Raspbian - Diagrama de potência de inicialização	35
Figura 19 Raspbian - Diagrama de energia de inicialização	35
Figura 20 Comparativo de tipos de conectividade e consumo	36

LISTA DE TABELAS

Tabela 1	Versões do Raspberry Pi	11
Tabela 2	Características da Câmera	14
Tabela 3	Modos de operação - INA219	19
Tabela 4	Medições de câmera	26
Tabela 5	Comparativo de sistemas operacionais	29

SUMÁRIO

1	INTRODUÇÃO.....	7
2	FUNDAMENTAÇÃO TEÓRICA	9
2.1	REDES DE SENSORES SEM FIO	9
2.2	RASPBERRY PI	11
2.2.1	ESPECIFICAÇÕES DE HARDWARE	11
2.2.2	O MÓDULO DE CÂMERA	13
2.3	IMAGEM EM REDES DE SENSORES	14
2.4	SEGURANÇA EM REDES DE SENSORES	15
2.5	CIRCUITO SENSOR DE CORRENTE	18
2.6	CONSIDERAÇÕES.....	19
3	METODOLOGIA.....	21
3.1	REVISÃO BIBLIOGRÁFICA	21
3.2	AVALIAÇÃO DOS SISTEMAS OPERACIONAIS	21
3.3	MEDIÇÕES DE CONSUMO ENERGÉTICO	22
3.3.1	CONSUMO NA INICIALIZAÇÃO	23
3.3.2	MEDIÇÕES SUBSEQUENTES.....	24
3.3.3	CÂMERA	25
3.3.4	CRİPTOGRAFIA	26
3.3.5	MATERIAIS	27
4	DESENVOLVIMENTO E RESULTADOS.....	28
4.1	AVALIAÇÃO DE SISTEMAS OPERACIONAIS	28
4.2	COMPARATIVO DE SISTEMAS OPERACIONAIS	29
4.3	MEDIÇÕES DE CONSUMO.....	31
4.3.1	INICIALIZAÇÃO	31
4.3.2	CONECTIVIDADE	37
	REFERÊNCIAS	38

1 INTRODUÇÃO

Nos últimos anos, avanços tecnológicos têm possibilitado o desenvolvimento de sensores para diferentes tipos de aplicações. Em suas primeiras gerações, estes sensores possuíam limitado poder de processamento e pequena capacidade de armazenamento, apresentando um conjunto de desafios relacionados à gerência de recursos (BARONTI et al., 2007). Contudo, o desenvolvimento da tecnologia e a redução dos custos vêm permitindo a construção de sensores com maior poder computacional, o que pode revolucionar a forma como dados escalares e contínuos podem ser monitorados (MOOSAVI; SADEGHI-NIARAKI, 2015). De maneira geral, um conjunto de nós organizados com uma finalidade específica de monitoramento e/ou controle, e transmitindo dados sem o uso de cabos ou fios, é denominado de rede de sensores sem fio (RSSF). Tais redes podem ser utilizadas nas mais diversas aplicações, tais como no monitoramento de desastres naturais, em ações militares ou na exploração de áreas de risco (YICK; MUKHERJEE; GHOSAL, 2008).

A associação de câmeras ao conjunto de nós sensores em uma RSSF abre margem para um novo conjunto de possibilidades de monitoramento. Com isso, novas camadas de complexidade são acrescentadas aos mecanismos já existentes. Esta nova classe de sensores é denominada Redes de Sensores Visuais Sem Fio (RSVSF)(ALMALKAWI et al., 2010). De fato, os desafios inerentes à captura, processamento e transmissão de dados visuais abre uma nova área de pesquisa, com particularidades próprias (AKYILDIZ; MELODIA; CHOWDHURY, 2007). Nesse contexto, o desenvolvimento e utilização de sensores visuais adequados, para um eficiente monitoramento com câmeras, é bastante desejável.

Um dos desafios que surgem do uso de câmeras em nós sensores é a grande quantidade de dados de imagem que podem ser produzidos. Um solução para esse potencial problema é a adoção de mecanismos de compressão (ZAINELDIN; ELHOSSEINI; ALI, 2015), embora tais mecanismos devam ser adequados à capacidade dos nós sensores. Tal cenário pode ser ainda mais complexo, pois pode-se requerer que a transmissão de dados visuais seja resistente à perda de pacotes, dificultando a adoção de algumas das possibilidades de algoritmos de codificação/compressão (PHAM, 2015). Em geral, se uma expressiva quantidade de dados é gerada por um nó sensor a partir da captura de imagens, por exemplo, pode-se optar entre utilizar codificação com alto custo computacional, para reduzir o montante de dados a transmitir, ou aumentar a largura de banda, que pode ser insuficiente em alguns casos (MAMMERI; HADJOU; KHOUMSI, 2012). Conhecer os recursos de processamento disponíveis em nós sensores é, portanto, desejável num contexto de RSVSF.

Além das questões inerentes à codificação e compressão de dados visuais, outro aspecto importante a ser considerado é a segurança dos dados que trafegam por uma RSVSF. De acordo com Winkler e Rinner (2014), devido à natureza dos dados visuais, algumas aplicações podem requisitar diferentes níveis de proteção aos dados trafegados. Contudo, o processo de cifra e recuperação demandam uma capacidade de processamento que pode não estar presentes em nós sensores, o que pode tornar inviável a aplicação de medidas de segurança (GONÇALVES; COSTA, 2015).

Assim, uma solução para garantir compressão e segurança em redes de sensores visuais sem fio é a utilização de sensores da nova geração, que possuem maior capacidade de processamento e armazenamento de dados, e cujos custos de aquisição vem caindo ao longo do tempo. Entre os potenciais dispositivos eletrônicos de baixo custo que podem atuar como nós sensores, o Raspberry Pi é um computador de baixo custo que foi apresentado em 2012 e atualmente tem se tornado referência por ser um dispositivo barato, de baixo consumo e com inúmeras interfaces de entrada e saída (VUJOVIĆ; MAKSIMOVIĆ, 2015). Segundo Vujovic e Maksimovic (2014), o Raspberry Pi traz as vantagens de um computador pessoal, como flexibilidade, ao domínio de redes de sensores. Além disso, esses dispositivos são portáteis, têm maior performance que sensores ditos tradicionais e fornecem possibilidades de uso no projeto de sistemas embarcados.

Este trabalho tem como objetivo realizar análises qualitativas e quantitativas em relação a adoção do Raspberry Pi como sensor visual em RSVSF, com foco na codificação, compressão e criptografia de imagens capturadas pela câmera embarcada no sensor. Mais especificamente:

- Determinar métricas para a avaliação de desempenho de sensores visuais;
- Avaliar os sistemas operacionais para o Raspberry Pi, seus recursos e limitações;
- Propor o uso de um sistema operacional baseado na avaliação anterior;
- Verificar a eficiência de algoritmos de codificação e criptografia de imagem nesse contexto; e
- Gerar comparativos de custo computacional e consumo energético.

Espera-se, assim, contribuir com a área de pesquisa em questão, oferecendo análises que servirão para corroborar (ou não) com a adoção desse tipo de sensor em RSVSF modernas.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda os temas que servem como aporte teórico para a argumentação utilizada na pesquisa. Está dividida como se segue: Redes de Sensores Sem Fio, incluindo informações sobre Redes de Sensores Visuais Sem Fio; O Raspberry Pi, computador de baixo custo utilizado na pesquisa; Imagem em Redes de Sensores Sem Fio, abordando os processos de aquisição e codificação; e Segurança em Redes de Sensores Visuais Sem Fio, mostrando os conceitos de Confidencialidade e o papel da Criptografia neste contexto.

2.1 REDES DE SENSORES SEM FIO

O conceito de Redes de Sensores Sem Fio (RSSF) vem mudando ao longo dos anos. Pottie (1998) em seu trabalho determina que os nós sensores têm baixíssima capacidade de processamento e largura de banda, transmitindo dados com pouca velocidade. Papageorgiou (2003) afirma que, dentre outras, as limitações de hardware devem sempre ser levadas em consideração. Já Buratti et al. (2009), numa pesquisa mais recente, tende a reduzir o impacto dessas limitações, quando afirma que uma RSSF geralmente pode ser descrita como uma rede de dispositivos que podem extrair informações do ambiente e se comunicar através de meios sem fio. O mesmo autor ainda considera que, devido à grande variedade de aplicações das RSSFs, os requisitos do sistema podem variar de forma considerável.

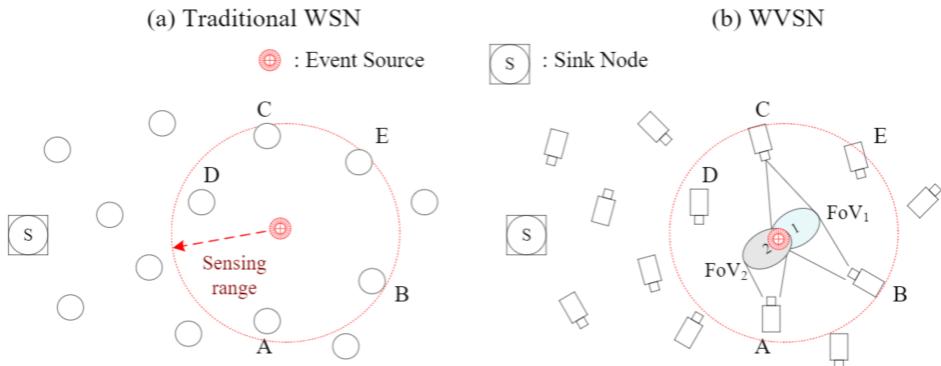
RSSFs podem estar presentes nas mais diversas aplicações. Desde o monitoramento do ambiente, saúde, logística e aplicações militares, as possibilidades geralmente podem ser separadas em duas categorias: detecção de eventos e estimativa de processos espaciais. Para a primeira classificação, os sensores são organizados de forma a detectar um evento, por exemplo um incêndio numa floresta ou terremoto etc e, neste caso, a decisão é tomada de forma distribuída, a fim de diminuir a probabilidade de alarme falso e manter a eficiência da rede. A segunda categoria objetiva estimar algum fenômeno físico que pode ser modelado como um processo aleatório bi-dimensional. Tais fenômenos podem incluir a pressão atmosférica ou temperatura de determinadas áreas. Os dados capturados são processados de forma distribuída, pelos próprios nós sensores, ou centralizada (BURATTI et al., 2009).

Os recentes avanços da microeletrônica possibilitaram o desenvolvimento de câmeras de baixo custo que, em conjunto com nós sensores tradicionais de RSSFs abrem uma enorme área de pesquisa, atribuída à uma nova classe de redes de sensores, as Redes de Sensores Visuais Sem Fio (RSVSF) (ALMALKAWI et al., 2010). Quando comparadas às RSSFs que só transmitem informações escalares, as RSVSFs permitem uma gama

de aplicações, já que os dados visuais fornecem informações mais ricas que podem ser aproveitadas, por exemplo, no monitoramento da vida selvagem e vigilância (YAP; YEN, 2014).

Segundo Yap e Yen (2014) as RSVSFs diferem das RSSFs em cinco aspectos principais: Cobertura do Campo de Visão, Estresse da Rede, Colisões de Pacotes, Processamento de Dados e Cobertura com Oclusão. (1) A Cobertura do CV trata-se de quais nós serão capazes de informar se um evento ocorreu na rede. Devido ao posicionamento das câmeras, além de um raio de comunicação, cada nó sensor possui um Campo de Visão que depende de uma certa direção ou ângulo. A figura 1b mostra como, mesmo que os nós sensores estejam dentro do raio de comunicação, nem sempre é possível obter informações do alvo. (2) O Estresse da Rede se dá por conta da massiva quantidade de dados gerados por cada nó sensor, demandando mais recursos de rede para sua transmissão. (3) A colisão de pacotes ocorre pois, quando um evento ocorre, os nós vizinhos tendem a transmitir dados de forma simultânea. Além disso, como a quantidade de dados é grande, os sensores devem transmitir uma sequência de pacotes para que a informação seja reconstruída, aumentando assim a probabilidade de colisões. (4) O Processamento em RSVSFs é mais complexo devido à natureza dos dados coletados, exigindo maior capacidade de processamento e armazenamento. Para evitar o consumo de tantos recursos de hardware, algumas técnicas são empregadas, como a compressão que será tratada em tópicos posteriores deste trabalho. (5) Cobertura com Oclusão surge do fato de que, mesmo o alvo estando dentro do Campo de Visão do nó sensor, obstáculos podem interferir no monitoramento. Isso resulta em cenários que não são desejáveis para uma RSVSF (YAP; YEN, 2014).

Figura 1: Cobertura em RSSFs e RSVSFs



Fonte: Yap e Yen (2014)

2.2 RASPBERRY PI

O Raspberry Pi é um computador portátil que teve sua primeira versão lançada em 2012. Criado pela Fundação Raspberry Pi, uma organização sem fins lucrativos sediada no Reino Unido, surgiu como forma de atrair os jovens para a computação sendo um dispositivo eletrônico fácil de configurar e modificar, despertando a curiosidade e interesse pela área como visto nos anos 1990 (SJOGELID, 2015; RASPBERRY PI FOUNDATION, 2016a).

2.2.1 Especificações de Hardware

O Raspberry Pi passou por várias revisões até o estado atual. A tabela 1 fornece os dados de lançamento para cada versão. Destes lançamentos, os modelos A e B das primeiras versões não estão mais sendo comercializados (RASPBERRY PI FOUNDATION, 2016a). Este trabalho visa estudar, inicialmente, a Versão 2 modelo B. Uma representação desta versão pode ser vista na figura 2.

Tabela 1: Versões do Raspberry Pi

Versão	Lançamento
Modelo B	2012
Modelo A	2013
Compute Module	2014
Modelo A+	2014
Modelo B+	2015
Versão 2 Modelo B	2015
Zero	2015
Versão 3 Modelo B	2016

Fonte: elinux.org

Lançado em 2015, o Raspberry Pi 2 modelo B possui recursos robustos de hardware. Custando apenas 35 dólares americanos, o computador dispõe de um processador ARM Cortex-A7 de quatro núcleos, operando a 900MHz e sendo compatível com uma gama de distribuições GNU/Linux. Além disso, possui 1 gigabyte de memória RAM, suficiente para a maioria das aplicações. Quanto às entradas e saídas, apresenta quatro portas USB (Barramento Serial Universal) que permitem conectar periféricos como um teclado, mouse ou um dispositivo de armazenamento flash. Possui ainda portas para saída de vídeo em alta definição (HDMI) e vídeo composto, saída de áudio e uma entrada para o

Figura 2: Raspberry Pi Modelo B



Fonte: Autor

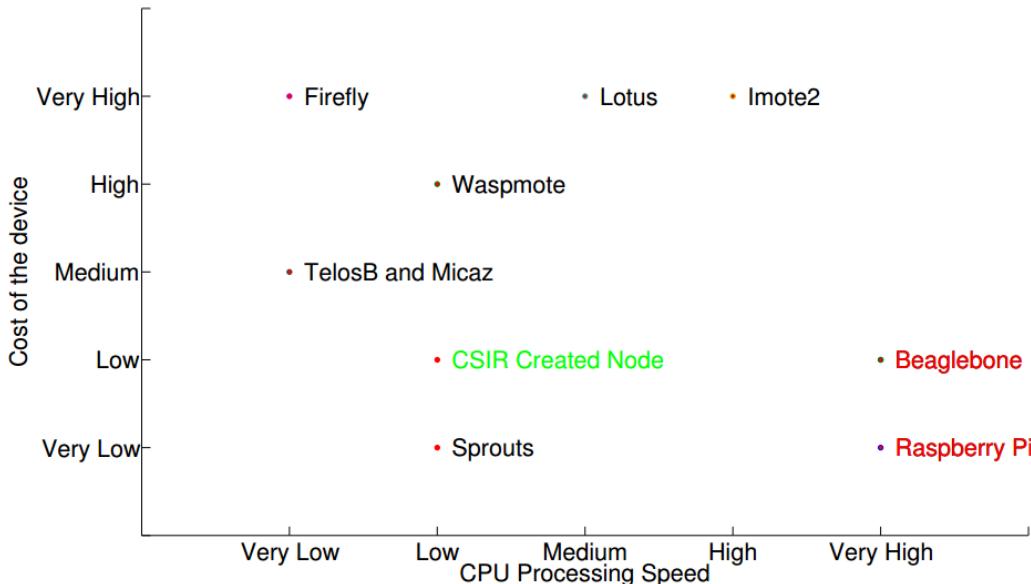
módulo de câmera que é produzido pela própria fabricante. Para conectividade há uma porta Ethernet, além da possibilidade de usar um adaptador Wifi USB.(RASPBERRY PI FOUNDATION, 2016b).

O Raspberry Pi é interessante no projeto de sistemas embarcados, pois fornece uma interface de prototipação, expondo 40 pinos digitais de propósito geral (GPIO). Estes pinos podem ser controlados via software nas duas vias: A aplicação pode alterar o valor da saída presente no pino ou responder a alterações na entrada por meio de interrupções (RASPBERRY PI FOUNDATION, 2016b; BROADCOM, 2012).

Quando comparado com outros modelos de hardware que também podem ser utilizados como nós sensores, como feito em Fisher et al. (2015), o Raspberry Pi tem a melhor relação custo x benefício. Os resultados da comparação podem ser verificados na figura 3. Os limites para os custos indicados vão de 30 dólares (Sprouts e Raspberry Pi) a 300 dólares (Lotus e Imote2).

Sendo um computador poderoso, de baixo custo e com alta possibilidade de customização, o Raspberry Pi torna-se atraente para projetos de redes de sensores. O RPi pode atuar tanto como um centralizador de informações, recebendo dados de outros nós da rede quanto como um nó responsável por sensoriamento que requer: (1) grande volume de dados, pois possui boa capacidade de armazenamento; (2) segurança, por seu alto poder de processamento, permitindo aplicações diversas de criptografia; e (3) conectividade, já que dispõe de soluções nativas de conexão (ethernet) e possibilidade de expansão com um conector USB (Wifi e Bluetooth) além de protocolos de RSSF como o X-Bee através das portas de entrada e saída (GPIOs).

Figura 3: Comparação de custo vs. Velocidade do processador



Fonte: Fisher et al. (2015)

2.2.2 O Módulo de Câmera

Tratando-se de RSVSFs, é essencial que haja uma forma de capturar imagens. Para isso, o Raspberry Pi pode fazer uso de câmeras USB, porém apresentando algumas desvantagens:

- Custo
- Tamanho
- Velocidade
- Consumo energético

Por outro lado, a Raspberry Pi Foundation tem como um de seus produtos um módulo de câmera compatível com a Camera Serial Interface (CSI), porta de dados presente no Raspberry Pi (figura 4).

Esta câmera consegue ser melhor que câmeras USB nos aspectos de velocidade, já que a CSI é diretamente ligada à unidade de processamento gráfico (GPU) deixando o processador livre para realizar outras tarefas; tamanho, já que normalmente as câmeras USB vêm com a estrutura plástica que é parte do design; consumo energético, já que a CSI é uma interface de baixa tensão (1.2V) e, mesmo trafegando dados em alta velocidade (cerca de 1 GB/s) não há uso da CPU no processo. No aspecto custo, o módulo da câmera sai pelo valor de 25 dólares americanos, o que é pouco para uma câmera que consegue capturar vídeos em alta definição (CALIN, 2015). Segundo Yap e Yen (2014) o que pode

Figura 4: Módulo de câmera do Raspberry Pi, versão 1.3



Fonte: Autor

ditar a escolha entre um módulo e uma câmera USB é a resolução da imagem - que tende a ser menor nos módulos - porém a qualidade da câmera fornecida pela Raspberry Pi Foundation prova que um módulo pode ser a escolha mais correta neste contexto. As características da do módulo de câmera estão sumarizadas na tabela 2.

Tabela 2: Características da Câmera

Característica	Valor
Resolução máxima de imagem	2592x1944 (5MP)
Resolução máxima de vídeo	1920x1080 (full HD)
Codificadores de imagem	JPEG, BMP, PNG, GIF, RGBA, YUV
Codificadores de vídeo	H264, MJPEG

Fonte: picamera.readthedocs.org

2.3 IMAGEM EM REDES DE SENSORES

Sabe-se que a adição de informações visuais em uma RSSF aumenta tanto a possibilidade de aplicações quanto a complexidade da rede. De fato, uma das diferenças principais entre RSSFs e RSVSFs está na forma a qual os sensores de imagem percebem a informação do ambiente. Além disso, o modelo de sensoriamento utilizado por câmeras é substancialmente diferente de outras redes sem fio: ao invés de obter dados de um raio denominado Raio de Sensoriamento (Sensing Radius, comum em redes de sensores escalares), os dados de imagem seguem um modelo direcional chamado de Campo de Visão (SORO; HEINZELMAN, 2009).

De acordo com ZainEldin, Elhosseini e Ali (2015), dados de imagem possuem uma grande quantidade de redundâncias por conterem pixels altamente relacionados entre si. Esta correlação abre possibilidade para a utilização de algoritmos que focam em reduzir a quantidade de dados de entrada: algoritmos de compressão. Os conceitos de codificação e compressão estão intimamente ligados e, enquanto a codificação consiste em organizar os dados de forma a serem recuperados e/ou transmitidos em um momento posterior, a compressão almeja organizá-los de forma eficiente, reduzindo o espaço necessário para armazená-los e, no caso de redes de sensores, reduzindo também o esforço da rede para os transmitir.

O processo de compressão varia de acordo com a natureza dos dados e com o resultado esperado. De forma geral, algoritmos de compressão podem ser classificados de duas formas: com perdas e sem perdas. Os benefícios de se utilizar uma compressão com perdas são: (1) a redução do tempo de codificação / decodificação; (2) maior taxa de compressão (definida como a razão entre o tamanho de saída e o tamanho de entrada) e (3) menor consumo energético, no caso aplicações que se preocupam com a energia (ZAINELDIN; ELHOSSEINI; ALI, 2015).

Uma das características desejáveis de algoritmos de compressão em RSVSFs é a baixa correlação de dados de saída. Numa Rede de Sensores os dados são enviados divididos em pacotes que carregam a informação a ser reconstruída e recuperada pelo operador da rede. Isso significa que, se há baixa correlação entre cada pacote, os efeitos gerados pela perda de pedaços da informação são minimizados. Tomando como exemplo o caso da codificação JPEG, Pham (2015) informa que o formato dificilmente suporta 10% de perda de pacotes. Sendo assim, a correta escolha de algoritmos de compressão impacta diretamente na qualidade da rede.

2.4 SEGURANÇA EM REDES DE SENSORES

Os dados que trafegam em uma RSSF podem representar informações sem compromisso com a segurança, a exemplo do monitoramento de fenômenos físicos como a temperatura ou pressão. Por outro lado, o contexto muda quando os nós sensores são equipados com câmeras, formando as RSVSFs. Como dados visuais fornecem muita informação do ambiente, certas aplicações - principalmente militares - demandam princípios de segurança que as redes podem não estar preparadas para lidar, a exemplo da confidencialidade, autenticidade e integridade. Dados confidenciais devem ter a característica de serem secretos para observadores externos, só podendo ser lidos pelo destinatário que possui a chave. A autenticidade permite afirmar que os pacotes que

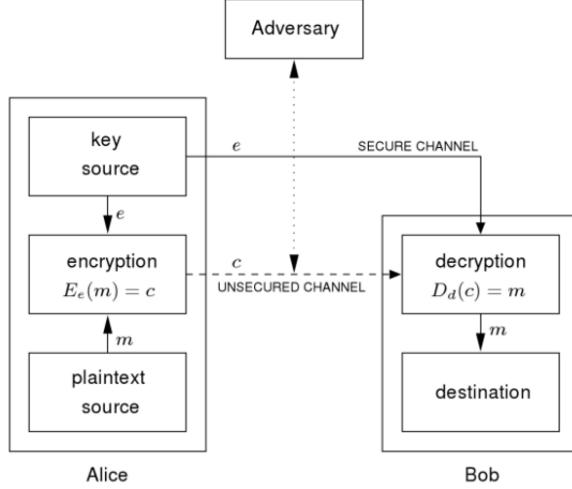
trafegam na rede são autênticos, isto é, provém de fontes conhecidas e seguras. A integridade diz respeito à não-manipulação dos pacotes, garantindo que as informações enviadas serão as mesmas recebidas (GONÇALVES; COSTA, 2015).

Para garantir tais princípios da segurança da informação, emprega-se a criptografia. De maneira simplificada, a criptografia é um conjunto de técnicas para transformar as informações que se desejam proteger em um conteúdo ilegível, passível de ser recuperado apenas pelo recipiente. Para isso, usa-se a chave de criptografia que deve ser mantida em segredo (GONÇALVES; COSTA, 2015 apud NAVEENKUMAR; PANDURANGA et al., 2013). Demonstrando um importante papel para prover segurança em RSVSFs modernas, a criptografia pode ser empregada para garantir: (1) confidencialidade, já que os dados só podem ser conhecidos pelos detentores da chave secreta; (2) autenticidade, já que os nós que contém as chaves podem ser identificados (assinatura) e (3) integridade, pois se os dados não são conhecidos, não podem ser modificados sem inviabilizar o processo inverso (GONÇALVES; COSTA, 2015).

Algoritmos de criptografia podem ser classificados quanto ao método de distribuição de chaves: chave simétrica e assimétrica. O método escolhido impacta diretamente no processo de cifra e recuperação. Considere um esquema de criptografia no qual as transformações $\{E_e : e \in \kappa\}$ e $\{D_d : d \in \kappa\}$ de cifra e recuperação, respectivamente, onde κ é o espaço de chaves válidas. O esquema é dito simétrico se para cada par (e, d) de chaves de cifra e recuperação é possível determinar e de forma eficiente (computacionalmente viável) conhecendo somente d e vice-versa. Como geralmente e e d são iguais, o nome simétrico faz mais sentido. Como exemplo, suponha que Alice deseja enviar uma mensagem para Bob. Alice criptografa a mensagem utilizando para isso uma chave que é conhecida tanto por ela quanto por Bob. Ao receber a mensagem, Bob utiliza a mesma chave e consegue recuperar a mensagem. Um grande problema surge pois o canal de comunicação deve ser seguro para que a chave só seja conhecida pelas partes interessadas e não por um observador externo (MENEZES; OORSCHOT; VANSTONE, 1996). O exemplo pode ser observado na figura 5.

Considere um esquema de cifra $\{E_e : e \in \kappa\}$ e $\{D_d : d \in \kappa\}$ um esquema recuperação, onde κ é o espaço de chaves válidas. Considere qualquer par associado de cifra/recuperação (E_e, D_d) e que cada par tem a propriedade de que, conhecendo E_e é ineficiente, dada uma mensagem cifrada aleatória c encontrar a mensagem $\{m | E_e(m) = c\}$. O método é denominado criptografia de chave pública se a chave e é mantida pública e a chave d é mantida em segredo. Para o esquema ser seguro, deve ser inviável computacionalmente encontrar a chave de recuperação d a partir da chave de cifra e .

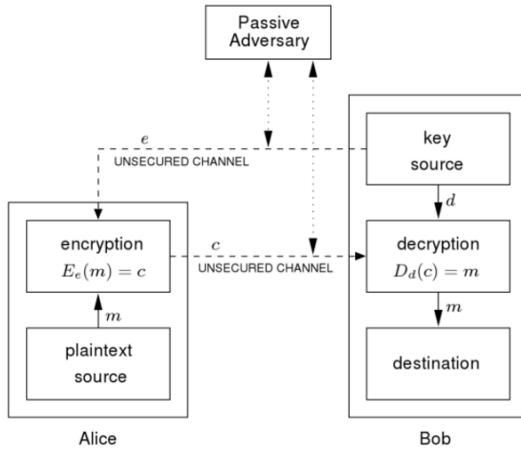
Figura 5: Criptografia de chave simétrica



Fonte: Menezes, Oorschot e Vanstone (1996)

Como exemplo, suponha que Alice deseja enviar uma mensagem para Bob. Suponha também que Bob detenha de um cadeado que só ele consegue abrir com sua chave secreta. Alice, em posse do cadeado de Bob escreve a mensagem e tranca o cadeado, podendo enviar o pacote por um meio inseguro sem que observadores externos possam recuperar a mensagem. (MENEZES; OORSCHOT; VANSTONE, 1996). O exemplo pode ser visto na figura 6.

Figura 6: Criptografia de chave pública

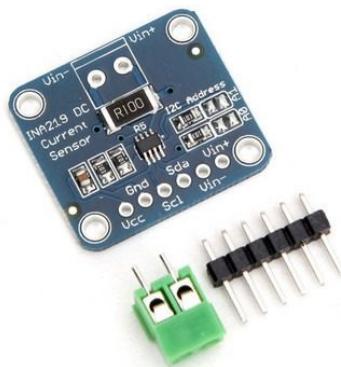


Fonte: Menezes, Oorschot e Vanstone (1996)

2.5 CIRCUITO SENSOR DE CORRENTE

Para a aquisição precisa de dados de corrente de um determinado dispositivo faz-se necessário o uso de um circuito sensor de corrente. Para este trabalho escolheu-se o circuito INA219 (figura 7), chip fabricado pela Texas Instruments em sua versão breakout-board desenvolvida pela Adafruit. Este circuito é capaz de medir tensões de 0 a 26V e variações mínimas de corrente (0.4 mA).

Figura 7: Circuito INA219



Fonte: Adafruit

Existem várias formas de se medir a corrente que passa por um condutor (utilizando leis da física de proporcionalidade da corrente à efeitos conhecidos). O circuito estudado e empregado neste trabalho faz o uso da Lei de Ohm, que diz que a tensão U (Volts) é diretamente proporcional à corrente i (Ampères), dada uma resistência fixa R (Ohms):

$$U = Ri$$

O circuito de medição escolhido utiliza um resistor muito pequeno denominado shunt. Este resistor é responsável por causar uma queda de tensão em série com o dispositivo sob teste que é proporcional à corrente consumida pelo dispositivo. No caso do INA219 o shunt possui uma resistência de 0.1 Ohms. A escolha deste resistor impacta diretamente nas medições, já que um valor muito grande causa uma queda igualmente grande de tensão (comprometendo o funcionamento do aparelho sendo testado); e um valor muito pequeno sofre com variações ínfimas na tensão de saída (KAUP; GOTTSCHLING; HAUSHEER, 2014).

O circuito INA219 é responsável por medir a queda de tensão causada pela corrente do dispositivo no shunt e o faz por meio de um conversor analógico / digital de 12 bits.

Vários modos de operação são possíveis e podem ser adaptados a cada aplicação. A tabela 3 mostra os modos possíveis e o tempo necessário para realizar uma leitura (TEXAS INSTRUMENTS, 2015).

Tabela 3: Modos de operação - INA219

Resolução / modo	Tempo
9 bit	84 μ s
10 bit	148 μ s
11 bit	276 μ s
12 bit	532 μ s
12 bit / 2 amostras	1.06 ms
12 bit / 4 amostras	2.13 ms
12 bit / 8 amostras	4.26 ms
12 bit / 16 amostras	8.51 ms
12 bit / 32 amostras	17.02 ms
12 bit / 64 amostras	34.05 ms
12 bit / 128 amostras	68.10 ms

Fonte: Texas Instruments

O modo de resolução de 12 bits possui ainda a operação de média dos últimos N valores como mostrado na tabela 3. A amostragem com média funciona como um filtro passa-baixas de primeira ordem, aliviando os efeitos da natureza intrinsecamente ruidosa do sinal. (TEXAS INSTRUMENTS, 2015)

2.6 CONSIDERAÇÕES

Este capítulo apresentou os conceitos que são de suma importância para este trabalho. Foram introduzidos os conceitos de Redes de Sensores Sem Fio e sua variante que utiliza câmeras para o sensoriamento mais rico do ambiente (Redes de Sensores Visuais Sem Fio); o Raspberry Pi; métodos de codificação e compressão de imagens; e aspectos relacionados à segurança de redes de sensores.

Nota-se que os conceitos estão relacionados através dos desafios inerentes ao uso de algoritmos clássicos de compressão e criptografia em um contexto que, predominantemente, faz-se o uso de dispositivos com baixa capacidade de processamento. Percebe-se também que o Raspberry Pi pode fornecer ensejos para a realização de redes de sensores modernas, robustas e de baixo custo, já que é um dispositivo eletrônico portátil que fornece muitas aplicações interessantes no projeto de sistemas embarcados.

O circuito de medição estudado mostra-se capaz de atuar com a precisão necessária para fornecer bases nas tomadas de decisão que este trabalho deseja proporcionar. Utilizar um circuito integrado permite que o custo seja reduzido e aumenta a confiabilidade dos resultados.

3 METODOLOGIA

Para o desenvolvimento deste trabalho, a metodologia utilizada foi fundamentada em redes de sensores sem fio, redes de sensores sem fio visuais, segurança em redes de sensores e nas especificações do Raspberry Pi. Pesquisas foram realizadas acerca de possíveis sistemas operacionais capazes de suprir as necessidades desta pesquisa: fácil acesso e instalação e inclusão das ferramentas de controle da câmera do Raspberry Pi.

3.1 REVISÃO BIBLIOGRÁFICA

O início deste trabalho exigiu a revisão de literatura descrita na seção 2. A revisão foi importante para aprofundar o entendimento dos conceitos relacionados à este trabalho e evidenciar os possíveis pontos de exploração para dar base aos objetivos gerais e específicos. A literatura deste trabalho possui produções recentes (o próprio Raspberry Pi foi lançado em 2015) e percebeu-se uma mudança na concepção e definição de uma rede de sensores sem fio. Os conceitos atuais demonstram uma maior preocupação com a segurança dos dados que trafegam numa rede sem fio, requisito que não era tão reforçado anos atrás.

3.2 AVALIAÇÃO DOS SISTEMAS OPERACIONAIS

Os sistemas operacionais escolhidos para avaliação implementam o `systemd`, que é responsável por orquestrar os processos e serviços durante a inicialização do sistema. O tempo que os SOs levam para inicializar foram medidos utilizando o comando `#:systemd-analyze`. A saída deste programa mostra quanto tempo decorreu-se desde o início até a finalização do boot.

Para aferir o tempo de resposta de rede, um software personalizado foi desenvolvido. Este programa é responsável por detectar que o Raspberry Pi foi ligado e, utilizando o comando `ping`, verificar quanto tempo leva até que o sistema operacional inicialize as rotinas de rede, momento que o RPi responde às requisições. Esta métrica é importante pois se tratando de RSSF, ter conectividade logo após a inicialização permite a interação com outros elementos da rede.

O tamanho de download foi medido como consequência da própria transferência das imagens dos sistemas operacionais. Cada SO teria que ser baixado para ser instalado nos cartões de memória, posteriormente inseridos no Raspberry. Há duas métricas decorrentes deste processo: Tamanho do download e tamanho da ISO. O download, apesar de ser uma métrica importante, não impacta de forma real o RPi pois os autores

dos SOs compactam a imagem ISO e a transferência destas imagens é feita através de um computador pessoal, anteriormente à instalação. Já o tamanho da ISO, formato de arquivo após a descompactação, tem um impacto direto, determinando a capacidade necessária do cartão de memória utilizado e, consequentemente, o custo.

As métricas de uso de recursos como memória RAM e uso do processador (quantidade de processos) foram capturadas após a inicialização do sistema operacional. O GNU/Linux provisiona um sistema de arquivos virtual denominado `/proc`. Nele estão contidas informações sobre processos, dispositivos, hardware. Pode-se acessar cada arquivo separadamente para buscar os dados de cada parte do sistema, por exemplo `cat /proc/cpuinfo` para a obtenção das configurações da CPU. Ocorre que há uma ferramenta que sumariza estas informações chamada `procinfo`. Ela está disponível em todos os repositórios utilizados nesta pesquisa (AUR, APT e DNF). Este software fornece o uso de memória RAM e a quantidade de processos executados até o momento.

Após o levantamento dos dados de cada sistema operacional, foi realizada a escolha do sistema operacional que seria utilizado para a realização das medições de consumo energético. Para isso, utilizou-se as métricas propostas e uma ordenação por ordem de importância para a aplicação em redes de sensores sem fio visuais. Para o contexto de RSSF, a ordem escolhida foi:

1. Tempo de boot / rede
2. Consumo de recursos (RAM e processos)
3. Tamanho das imagens (descompactadas)
4. Documentação disponível

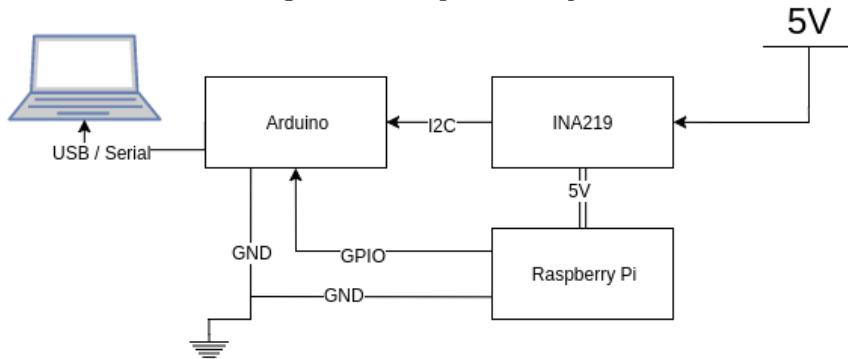
3.3 MEDIÇÕES DE CONSUMO ENERGÉTICO

Para aquisição dos dados de consumo energético, desenvolveu-se um sistema de captura que utiliza um computador pessoal, um Arduino e um circuito integrado sensor de corrente. Este setup pode ser observado na figura 8.

O processo de captura inicia-se com a alimentação do Raspberry Pi. Um adaptador converte a tensão de rede (110V) em uma saída regulada de 5V que servirá de fonte para o Raspberry. O fio positivo foi então cortado (fisicamente) e conectado aos pinos V_{in+} e V_{in-} do INA219 (figura 9).

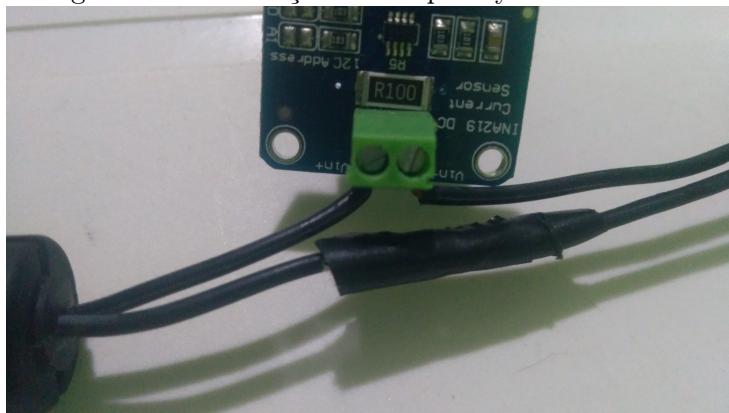
O INA219 se conecta ao Arduino por meio do protocolo I2C. Toda a configuração do sensor é feita utilizando a biblioteca Arduino-INA219 desenvolvida por Korneliusz Jarzębski (código no Github). Esta biblioteca, apesar de não oficial (a Adafruit também a disponibiliza), possui todos os parâmetros necessários para configurar o INA219, ao

Figura 8: Setup de medição



Fonte: Autor

Figura 9: Alimentação do Raspberry Pi no INA219



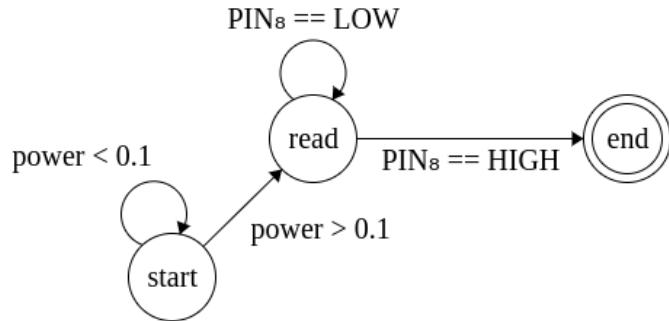
Fonte: Autor

contrário da implementação feita pela Adafruit. Com ela é possível alterar o modo de amostragem e resolução, limites de tensão e corrente suportados, dentre outros ajustes finos.

3.3.1 Consumo na Inicialização

Para medir o consumo na inicialização, a instrumentação está configurada como visto na figura 10. No estado inicial, com a fonte desconectada, a potência de saída é 0 e, portanto, o Arduino não envia nenhum dado via serial. Assim que a fonte é conectada, o sistema passa a operar em modo de leitura até que o pino 8 (GPIO digital escolhido arbitrariamente) possua o nível lógico 1. A transição de estado lógico no pino de entrada do Arduino ocorre devido ao Raspberry Pi executar uma sequência de comandos (script bash) assim que a inicialização é finalizada. Este script utiliza o pacote WiringPi e os comandos #: `gpio mode 1 out`; `gpio write 1 1` para, respectivamente, alterar o pino 1 do Raspberry para o modo de saída e alterar o estado para o nível lógico alto (1).

Figura 10: Medição de inicialização

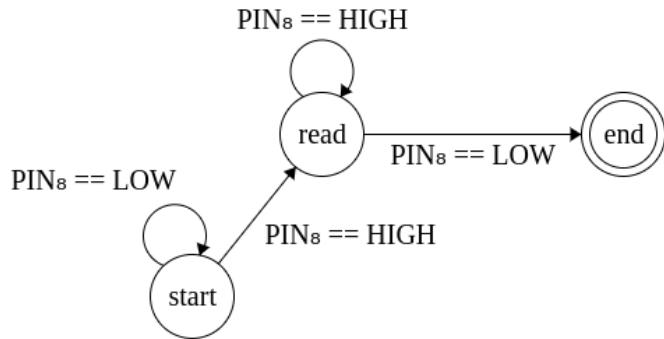


Fonte: Autor

3.3.2 Medições Subsequentes

Para os outros testes, a medição no Arduino é iniciada pelo Raspberry Pi. De acordo com a figura 11, o estado se mantém até que o nível lógico alto seja detectado pelo pino 8, momento que se inicia a leitura dos dados de consumo energético. Todos os testes foram construídos de modo a realizar sempre as ações de alterar o estado para o nível logico alto no início do teste e baixo no final do teste. Nas seções posteriores, estes procedimentos (saída alta e baixa) serão omitidos das descrições.

Figura 11: Medição ativada pelo Raspberry Pi



Fonte: Autor

No Arduino, ambos os comportamentos implementam as máquinas de estados finitas apresentadas nas figuras 10 e 11. O Arduino requer a implementação de duas funções: `void setup()` e `void loop()`. A função `setup` é responsável por configurar o software e é executada somente uma vez. Na função `loop` ocorre o processamento da aplicação e é nela que as leituras são feitas e enviadas via serial. Uma propriedade interessante, e que facilitou o projeto das mudanças iniciais de estado no Arduino, é que

a função `loop` só é executada caso a `setup` termine. Um laço de repetição (`while`), escrito no `setup`, trava o curso normal do programa enquanto a primeira condição não é satisfeita (a leitura da potência > 0.1 para o caso da inicialização ou `digitalRead(8) == 1` para todos os outros casos).

O Arduino possui um temporizador interno que informa quanto tempo se passou desde o início do programa. Ao transicionar de estado para o modo leitura, o software grava o tempo no qual isso aconteceu. Posteriormente, as leituras são realizadas e o tempo é calculado relativo ao tempo inicial. Sendo assim, a saída do programa sempre vai proporcionar medições marcadas como iniciadas no tempo 0 (exceto pela latência de leitura de aproximadamente 2 a 5ms).

Os dados de leitura são enviados ao computador através da porta serial que é acessada com o cabo USB do arduino. Nos sistemas GNU/Linux ela pode ser encontrada em `/dev/ttyACMx`, sendo `x` um número sequencial atribuído pelo sistema. Para todas as medições, utilizou-se a velocidade de `115200 baud (bits/seg)` sem nenhum controle de fluxo. Um software personalizado, desenvolvido utilizando-se a linguagem Python é responsável por ler a saída serial do Arduino e salvar os dados em um arquivo CSV (separado por vírgulas).

3.3.3 Câmera

Para as medições de consumo da câmera, serão explorados os modos de captura e resolução listados na tabela 4. A fim de mais precisamente medir apenas o consumo da câmera, fez-se a subtração da medição de consumo em repouso. Os dados de consumo em repouso foram obtidos através da amostragem cinco minutos após a inicialização, processo que durou dez minutos. Calculou-se então a média das amostras neste período. Os dados foram capturados sem a presença de nenhuma conexão de rede.

Tabela 4: Medições de câmera

Modo	Resolução	Codificação
100 imagens	640x480	JPEG
100 imagens	1280x720	JPEG
100 imagens	1920x1080	JPEG
100 imagens	2592x1944	JPEG
100 imagens	640x480	YUV
100 imagens	1280x720	YUV
100 imagens	1920x1080	YUV
100 imagens	2592x1944	YUV
60 segundos de vídeo	640x480	H264
60 segundos de vídeo	1920x1080	H264
60 segundos de vídeo	640x480	MJPEG
60 segundos de vídeo	1920x1080	MJPEG

Fonte: Autor

3.3.4 Criptografia

Para a execução dos testes de criptografia escolheu-se a ferramenta `gpg` versão 1.4.22. Este software de código aberto implementa os seguintes tipos de algoritmos criptográficos de chave simétrica: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128, CAMELLIA192, CAMELLIA256. Destes algoritmos, foram escolhidos três: AES, AES256 e Twofish. O algoritmo AES (originalmente conhecido como Rjindael) foi o campeão do concurso para a escolha do algoritmo padrão de criptografia utilizado em todo o mundo (AES, do inglês Advanced Encryption Standard ou padrão de criptografia avançada em tradução livre) no ano de 2001. Sua versão AES256 utiliza 256 bits para o tamanho da chave (contra 128 bits na versão original). O algoritmo Twofish foi o segundo colocado no concurso e até o momento não foi encontrado nenhum tipo de vulnerabilidade.

A metodologia proposta tem o seguinte formato: Aplicar criptografia em arquivos de crescente tamanho e realizar as medições de consumo energético, buscando encontrar uma relação entre o tamanho do arquivo e a energia consumida no processo. Para cada formato, foram empregadas cinco medições seguindo a metodologia proposta na subseção 3.2.2 e registrado o consumo médio destas aferições. A sequência de comandos executada para este teste é a seguinte:

```
dd if=/dev/urandom of=./1mb.dat bs=4k /
  iflag=fullblock ,count_bytes count=1M
gpg --cipher-algo AES --passphrase password /
-o --c 1mb.dat > /dev/null
```

O primeiro comando cria um arquivo de tamanho 1MB e o salva em 1mb.dat. Este arquivo é constituído de conteúdo aleatório proveniente do dispositivo virtual `/dev/urandom`. Após isso, a ferramenta gpg é executada tendo como parâmetros uma senha para a criptografia de chave simétrica (password), o arquivo de origem (1mb.dat) e a saída direcionada para `/dev/null`, ou seja, desconsiderada. Note que é indesejável que o gpg escreva no cartão de memória para fins de medição de consumo, já que operações de escrita são computacionalmente custosas. Os comandos acima foram repetidos alterando os parâmetros correspondentes de tamanho de arquivo e tipo de algoritmo de criptografia.

3.3.5 Materiais

Fez-se o uso dos seguintes materiais utilizados para a realização deste trabalho:

- (1) Computador pessoal para a captura e sumarização dos dados de medição e escrita dos scripts de automação dos processos;
- (2) Raspberry Pi sendo o dispositivo sob teste;
- (3) Cartões de memória Secure Digital (SD) de tamanhos variados para proporcionar a instalação de diversos sistemas operacionais;
- (4) ódulo de câmera do Raspberry Pi, transformando o RPi num sensor visual e inserindo-o no contexto deste trabalho;
- (5) Módulo sensor de corrente INA219 (TI / Adafruit) que permitiu leituras precisas de consumo energético disponibilizando-os em um formato de fácil leitura;
- (6) Arduino Uno, plataforma de prototipagem que foi responsável pela interface com o INA219 e o Computador pessoal;
- (7) Fios / Jumpers para as ligações entre componentes; e
- (8) Fonte 5V/2A regulada para fornecer corrente ao Raspberry Pi. Esta fonte teve seu fio cortado e ligado diretamente ao INA219.

4 DESENVOLVIMENTO E RESULTADOS

O objetivo geral deste trabalho é avaliar se o Raspberry Pi é viável para ser utilizado como um nó de uma rede de sensores sem fio visual. Este objetivo deve ser alcançado tomando como base uma vertente energética: deseja-se testar se o processo de captura, compressão/codificação e criptografia possui um consumo de energia aceitável.

A revisão bibliográfica deu a entender que utilizar compressão e criptografia de grandes quantidades de dados não é uma tarefa que o nó sensor deve realizar. Os sensores devem ser capazes de registrar a informação e enviar para a base, para que sejam realizados os processos citados. Porém, apesar dos nós base terem energia virtualmente infinita, os recursos de processamento e rede ainda são limitados.

A seguir, detalham-se os passos executados para a investigação do problema citado e possíveis soluções.

4.1 AVALIAÇÃO DE SISTEMAS OPERACIONAIS

Definiu-se um conjunto de regras desejáveis para que um Sistema Operacional (SO) possa ser utilizado no Raspberry Pi, permitindo seu uso como nó de uma rede de sensores sem fio visual. A saber:

(1) Portabilidade: O tamanho da imagem (ISO) dita a capacidade que o cartão SD deverá ter. Imagens maiores necessitam de mais espaço, restringindo, por exemplo, a quantidade de dados que o nó sensor possa armazenar internamente, além de demandar custos com cartões maiores. Se uma imagem não estiver disponível, o SO deve ser fácil de instalar, com instruções claras que não necessitem de documentação extra.

(2) Consumo de memória em repouso: O consumo de memória RAM é mais importante que o espaço ocupado no cartão SD, já que não é possível substituir o chip de memória do Raspberry, limitando todas as aplicações a 1GB (no caso do RPi 2B, menos o total consumido pelo SO) e isso pode ser um fator limitante a depender da aplicação.

(3) Uso do processador em repouso: Esta característica indica a quantidade de serviços que executam para manter o SO funcionando. Mais serviços necessitam de mais processamento e o uso da CPU está linearmente relacionado com o consumo de energia. Quando a demanda energética é levada em consideração, é desejável que o SO contenha o menor processamento em repouso possível, ainda sendo possível realizar as tarefas necessárias ao funcionamento da aplicação (um exemplo é desativar o driver de vídeo para poupar energia).

(4) Tempo de inicialização: O uso do Raspberry em sistemas embarcados pode ser combinado com microcontroladores que consomem ordens de magnitude menos energia.

Os microcontroladores seriam responsáveis por estar cientes do estado da rede e enviar sinais para o Raspberry Pi, acordando-o e fazendo-o realizar tarefas as quais os requisitos de processamento e memória são elevados. Saber o tempo de boot é interessante pois a informação pode perder relevância com o passar do tempo.

(5) Ferramentas nativas: É interessante que o SO detenha de ferramentas para o gerenciamento e instalação das aplicações (package-manager), permitindo a replicação do estado com a execução de um script, fixando versões de pacotes que garantidamente funcionam para o fim desejado (isso descarta a possibilidade de que a atualização de algum pacote possa quebrar a aplicação). O SO deve possuir drivers de câmera, rede, USB etc disponíveis e de fácil configuração.

(6) Compatibilidade com ferramentas já consolidadas: Isso facilita o uso de linguagens de script (Python, Lua, Javascript), dando consistência e preparando a aplicação para uma possível atualização de hardware ou SO.

(7) Customizabilidade: É interessante que o SO tenha a capacidade de ser modificado para atender necessidades específicas (desativar certos componentes por padrão, incluir componentes instalados por padrão)

(8) Documentação: Listagem dos recursos disponíveis (gerenciador de pacotes, serviços, drivers etc), manuais de instalação

4.2 COMPARATIVO DE SISTEMAS OPERACIONAIS

Os sistemas operacionais escolhidos na pesquisa inicial foram testados e avaliados nas métricas apresentadas na seção anterior. A tabela 5 mostra os resultados obtidos com a instalação e execução dos SOs.

Tabela 5: Comparativo de sistemas operacionais

Sistema operacional	Tempo p/ iniciar	Reposta de rede	Download	ISO	RAM	Processos	Docs
Raspbian Jessie	19s	27s	1.6GB	4.3GB	111MB	884	Oficial
Raspbian Jessie (Lite)	15.7s	22s	284MB	1.26GB	73MB	657	Oficial
Minibian	12s	18s	225MB	794MB	31MB	351	Blog do autor
ArchLinux ARM	8.2s	14s	281MB	1.86GB	22MB	197	Arch Wiki
Fedberry Minimal	4.9s	10s	171MB	2.28GB	30MB	294	Site oficial Github

Fonte: Próprio autor

O sistema operacional Raspbian Jessie é a distribuição oficial da Raspberry Foundation. Ele se baseia na distribuição Debian e possui diversos recursos para tornar o Raspberry Pi um microcomputador utilizável para fins genéricos. A abundância de pacotes, drivers e serviços é refletida claramente nos dados coletados: Tamanhos de download e imagem descompactada grandes; alto tempo de inicialização e resposta de rede; alto uso de memória RAM e processos. Em compensação, este sistema operacional possui a melhor documentação (Site oficial dos criadores) e garantias de que as ferramentas disponibilizadas são totalmente compatíveis com o hardware.

O Raspbian Jessie Lite é a versão do Raspbian no qual excluiu-se a interface gráfica. O tamaho da imagem descompactada torna-se 1.4GB (versus 4.3GB da versão completa). Apesar do nome "lite", esta distribuição possui todos os recursos do Raspbian no que tange às ferramentas de configuração da placa - somente o X-server e seus componentes que não são instalados (responsáveis pela parte gráfica). Como a utilização do Raspberry Pi como nó sensor não necessita de acesso gráfico, a utilização do Raspbian Lite é recomendada.

Os sistemas operacionais não oficiais incluem - dentre outros - o Minibian, ArchLinux ARM e o Fedberry. Estes sistemas foram escolhidos pois, apesar de não oficiais, são capazes de realizar as tarefas propostas por este trabalho: captura, codificação e compressão de imagens. O Minibian é uma versão modificada do SO oficial (Raspbian), porém com muito mais pacotes removidos. O ArchLinux ARM e o Fedberry são, respectivamente, criações baseadas nas distribuições do Arch e do Fedora. Com o auxílio dos gerenciadores de pacotes, é possível instalar todas as ferramentas necessárias para a execução dos testes que serão realizados posteriormente ao longo da pesquisa.

Em termos de tempo de inicialização, o sistema que se mostrou mais rápido nos testes foi o Fedberry. O ganho de performance foi considerável, se comparado ao Raspbian (cerca de 4 vezes mais rápido para inicializar e responder a comandos de rede). Se comparado à versão "lite"do Raspbian, o sistema apresentou um desempenho 3.4 vezes melhor. Quando comparado ao segundo melhor sistema neste quesito, o Fedberry ainda possui um desempenho surpreendentemente superior (quase 2 vezes mais rápido).

O uso de memória RAM em repouso também é um quesito a se observar. O sistema que se saiu melhor nos testes foi o ArchLinux, apresentando um consumo de apenas 22 megabytes. Isso é importante pois parte da memória RAM do Raspberry Pi é compartilhada com a unidade de processamento gráfico, responsável pela codificação das imagens da câmera. Além disso, a utilização dos algoritmos de criptografia também requerem certa quantidade de memória.

A quantidade de processos em execução logo após inicializar o sistema também

é uma métrica importante. Mais processos requerem mais carga do processador e, consequentemente, maior consumo energético. Mais uma vez, o ArchLinux mostrou-se o mais "leve" neste quesito. Quando comparado ao Raspbian oficial, há 4.5 vezes menos processos executando em segundo plano. O Fedberry, apesar de otimizado, ainda possui cerca de 100 processos a mais que o Arch.

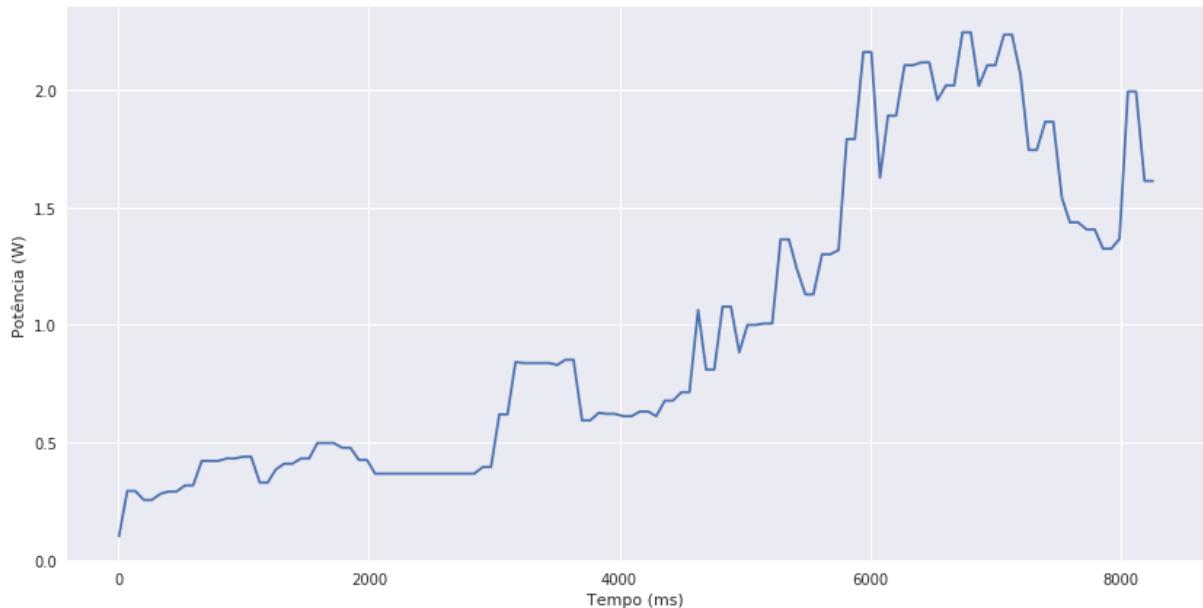
4.3 MEDIÇÕES DE CONSUMO

Seguindo a metodologia proposta na seção 3.2, procedeu-se com as medições de consumo. Os dados obtidos foram processados utilizando as bibliotecas `pandas` e `seaborn` do Python para análise e visualização dos dados. As subseções a seguir demonstram os dados de consumo de inicialização, repouso, captura de imagens e vídeos, compressão de arquivos, criptografia de arquivos e transferência de rede.

4.3.1 Inicialização

Os dados de consumo da inicialização do sistema operacional foram capturados de forma individual para os sistemas operacionais listados na tabela 5. As figuras 12 e 13 mostram, respectivamente, a plotagem do consumo de potência durante a inicialização e o gráfico de energia consumida para o sistema operacional Fedberry.

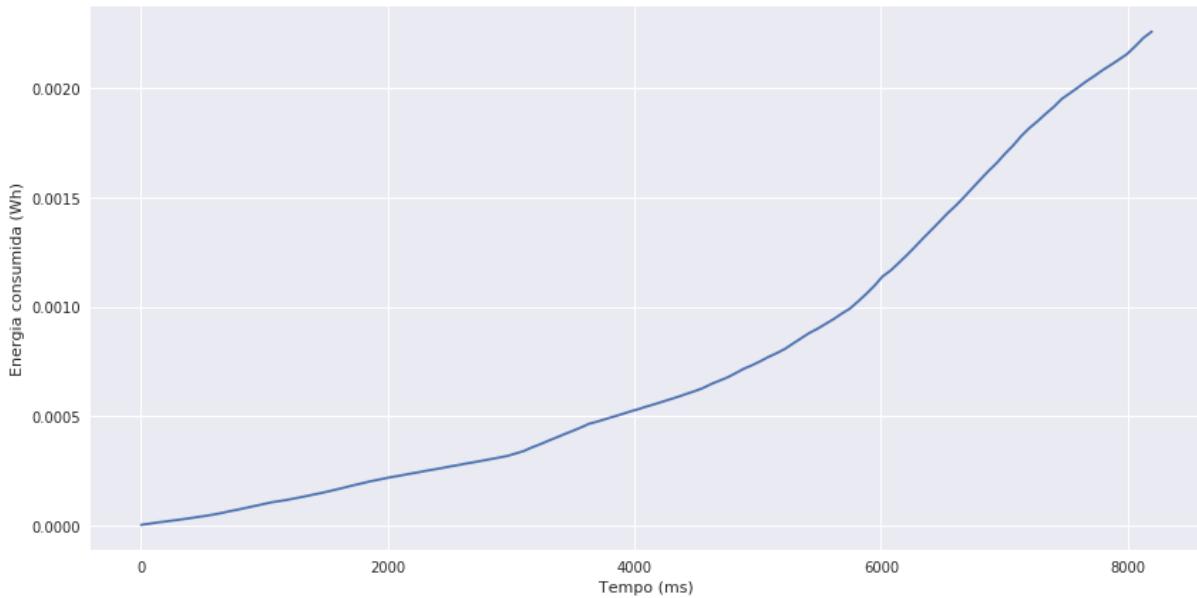
Figura 12: Fedberry - Diagrama de potência de inicialização



Fonte: Autor

As figuras 14 e 15 mostram, respectivamente, a plotagem do consumo de potência durante a inicialização e o gráfico de energia consumida do sistema operacional ArchLinux

Figura 13: Fedberry - Diagrama de energia de inicialização



Fonte: Autor

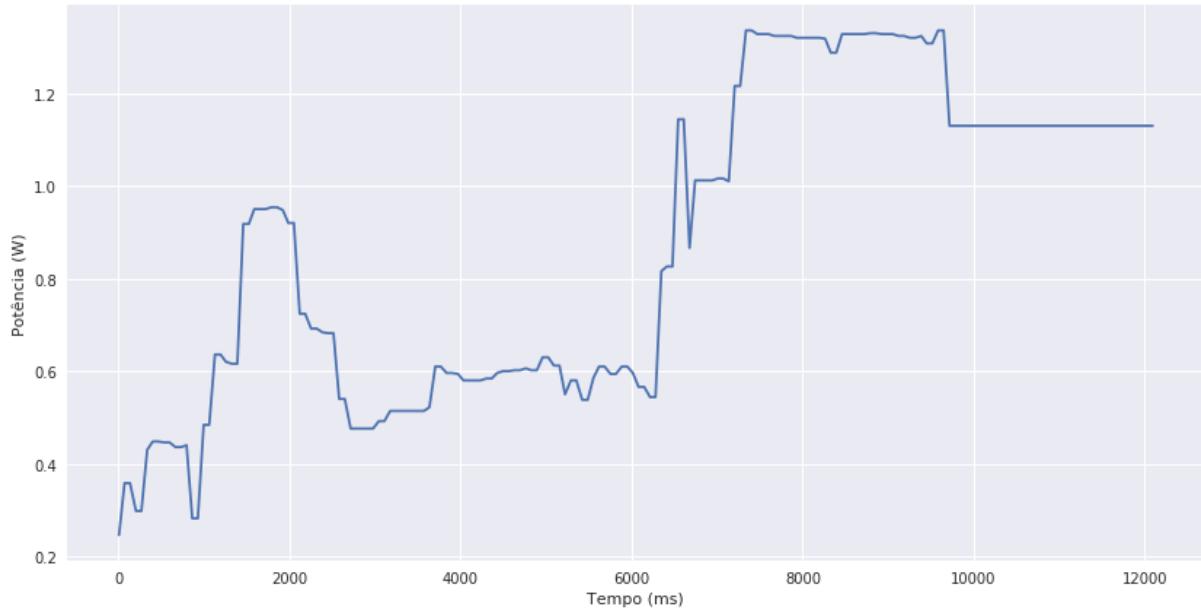
ARM. Este sistema, apesar de possuir um tempo de boot maior que o Fedberry, mantém a potência abaixo dos 1.3 Watts e seu consumo energético acumulado é parecido com o primeiro SO avaliado. Isso se deve ao fato que o ArchLinux foi projetado para ser o mais limpo possível, sem funções desnecessárias para o funcionamento fundamental do dispositivo e as ferramentas de desenvolvimento geralmente devem ser instaladas posteriormente.

As figuras 16 e 17 mostram, respectivamente, a plotagem do consumo de potência durante a inicialização e o gráfico de energia consumida do sistema operacional Raspbian Jessie Lite. Esta versão minimalista do Raspbian exibe um padrão de consumo característico: por volta de 12s o sistema aparentemente finalizou a sequência de inicialização, porém o serviço de cliente DHCP está aguardando a atribuição do endereço de IP.

As figuras 18 e 19 mostram, respectivamente, a plotagem do consumo de potência durante a inicialização e o gráfico de energia consumida do sistema operacional Raspbian. Este SO é o que consome mais recursos, sendo cerca de 5 vezes mais oneroso quando comparado ao Fedberry. Este não é um sistema operacional recomendado para a utilização como nó de uma rede de sensores sem fio, por contar com diversos pacotes que acabam elevando a latência das ações do sistema.

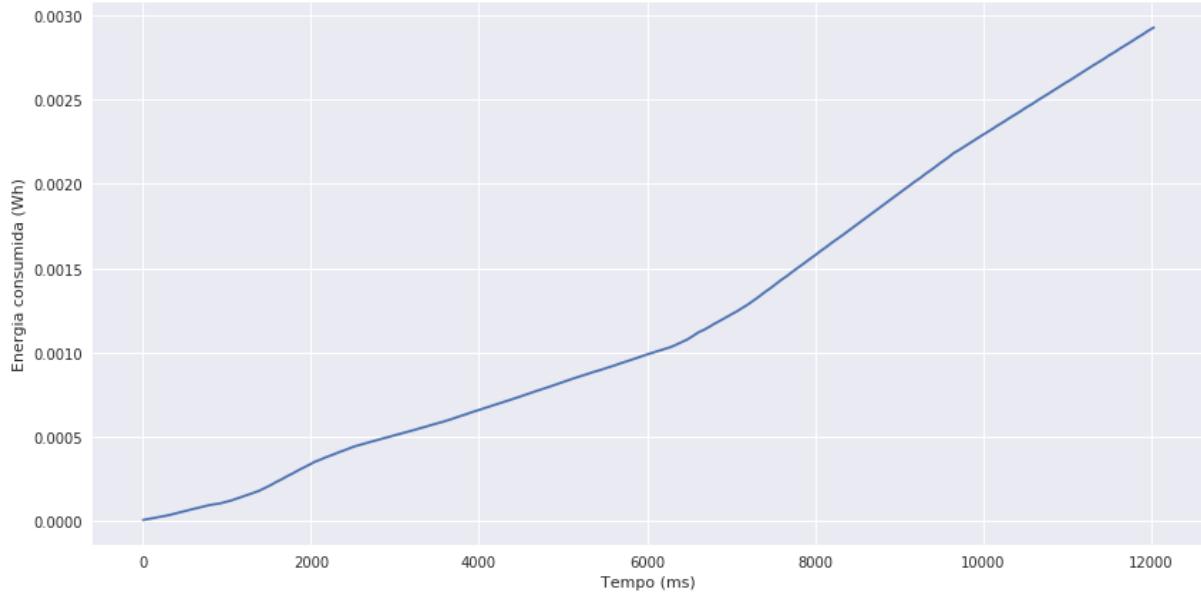
A figura 20 traz um comparativo do consumo energético para os diferentes tipos de conectividade disponíveis (WiFi, ethernet, offline). As figuras de 12 a 19 mostram os

Figura 14: ArchLinux ARM - Diagrama de potência de inicialização



Fonte: Autor

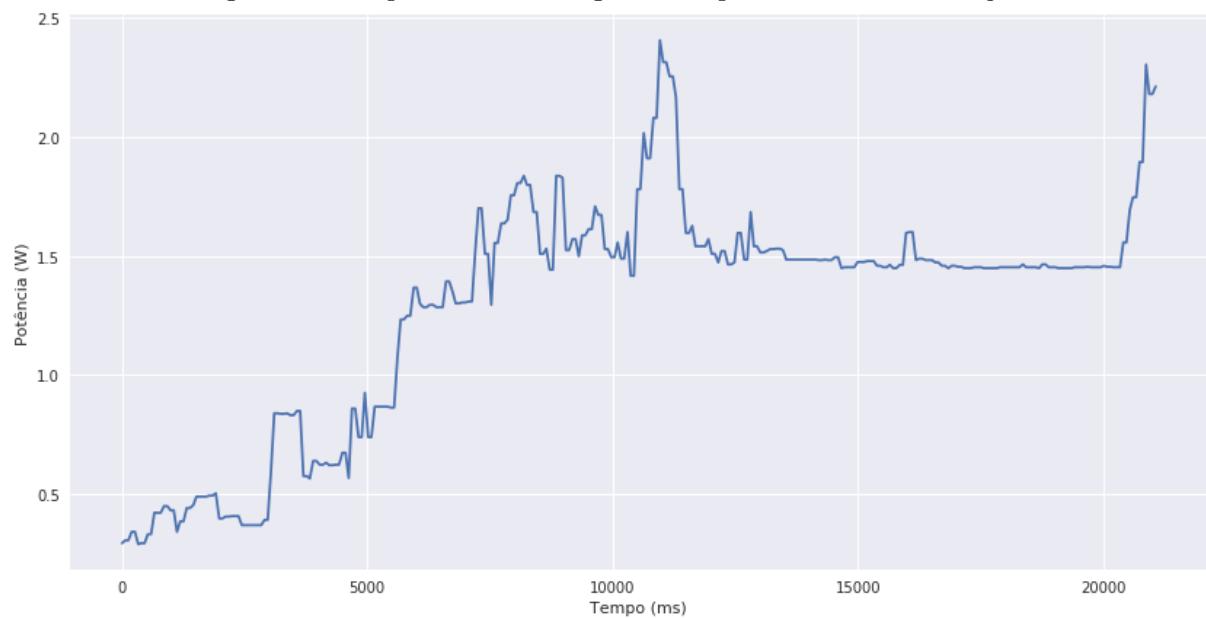
Figura 15: ArchLinux ARM - Diagrama de energia de inicialização



Fonte: Autor

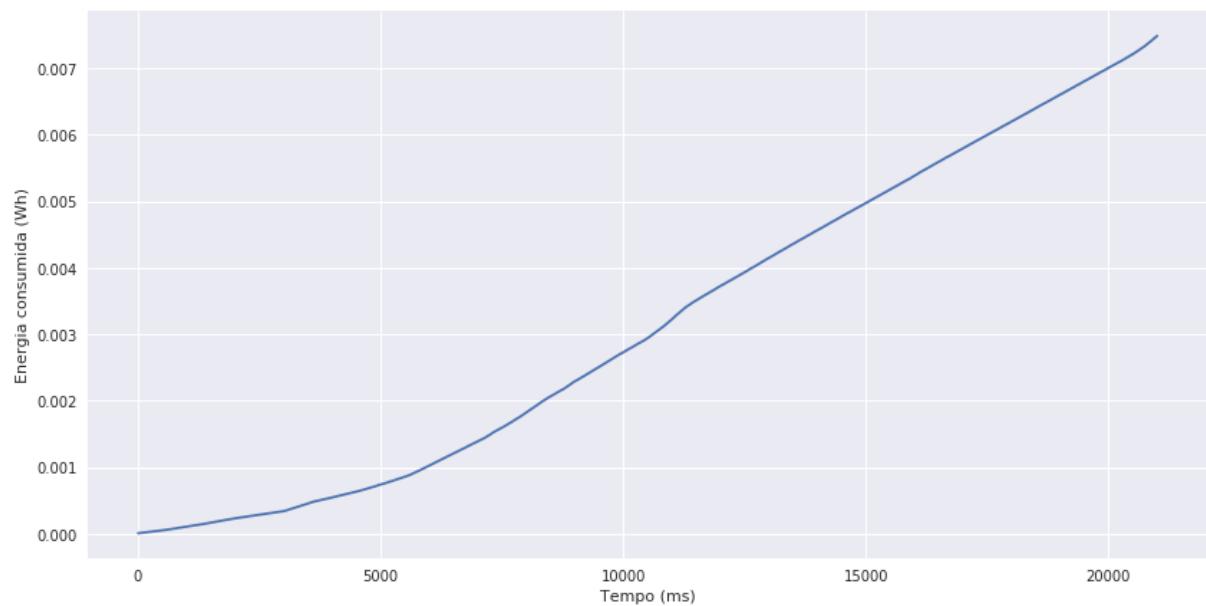
gráficos de consumo para o caso específico do uso do adaptador WiFi, já que, se tratando de RSSF, o sistema, pelo menos, deve ser capaz de se comunicar sem fios.

Figura 16: Raspbian Lite - Diagrama de potência de inicialização



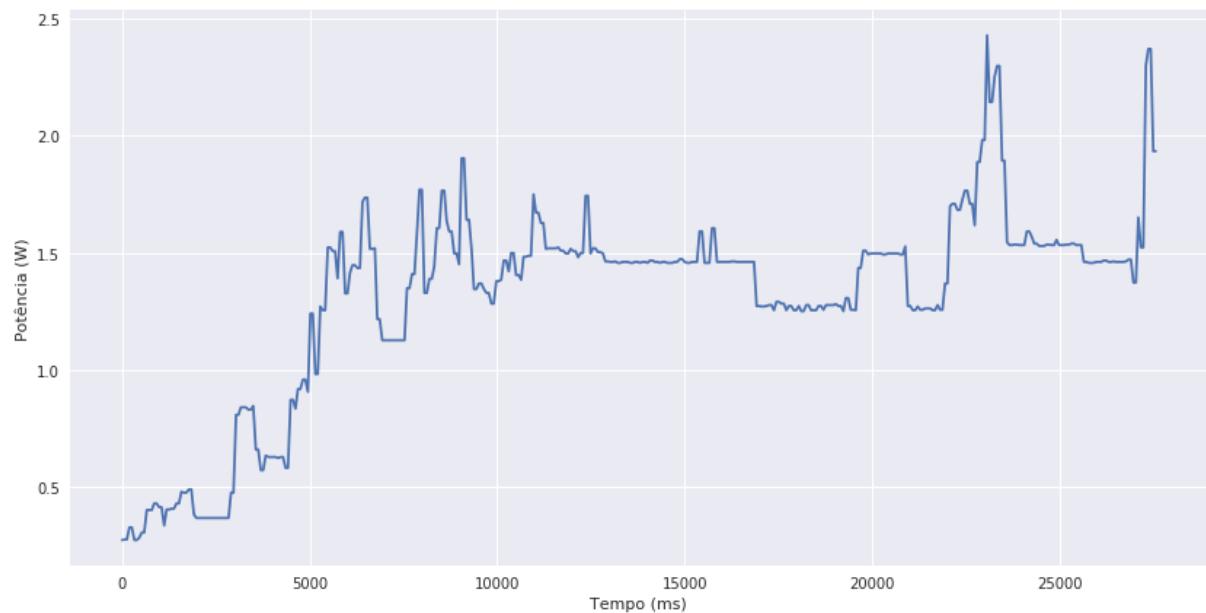
Fonte: Autor

Figura 17: Raspbian Lite - Diagrama de energia de inicialização



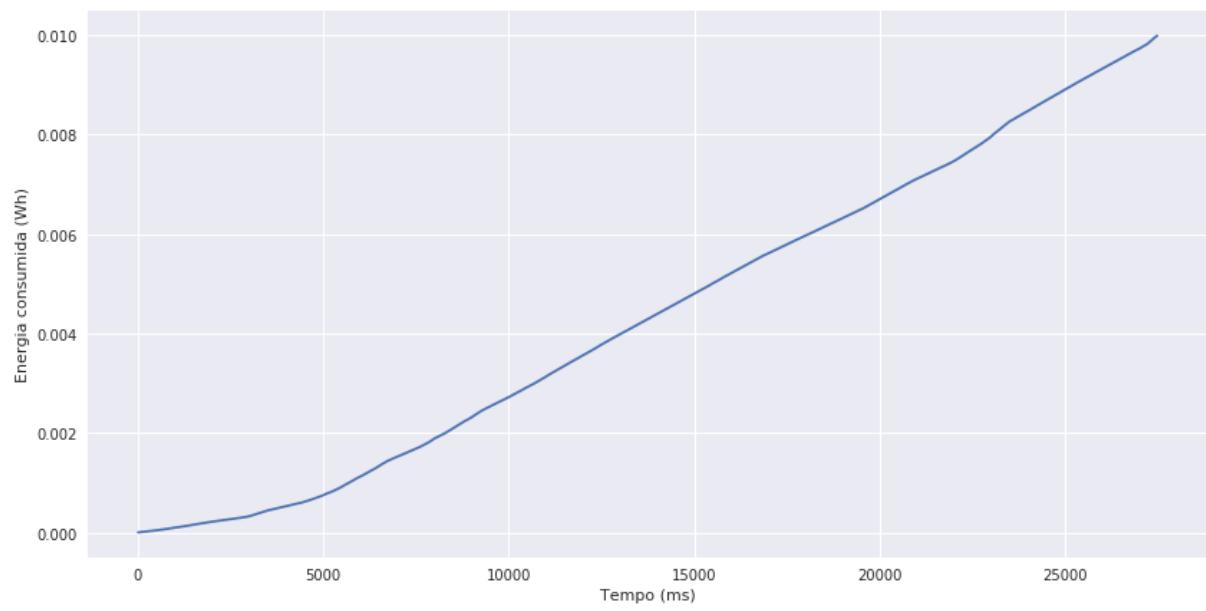
Fonte: Autor

Figura 18: Raspbian - Diagrama de potência de inicialização



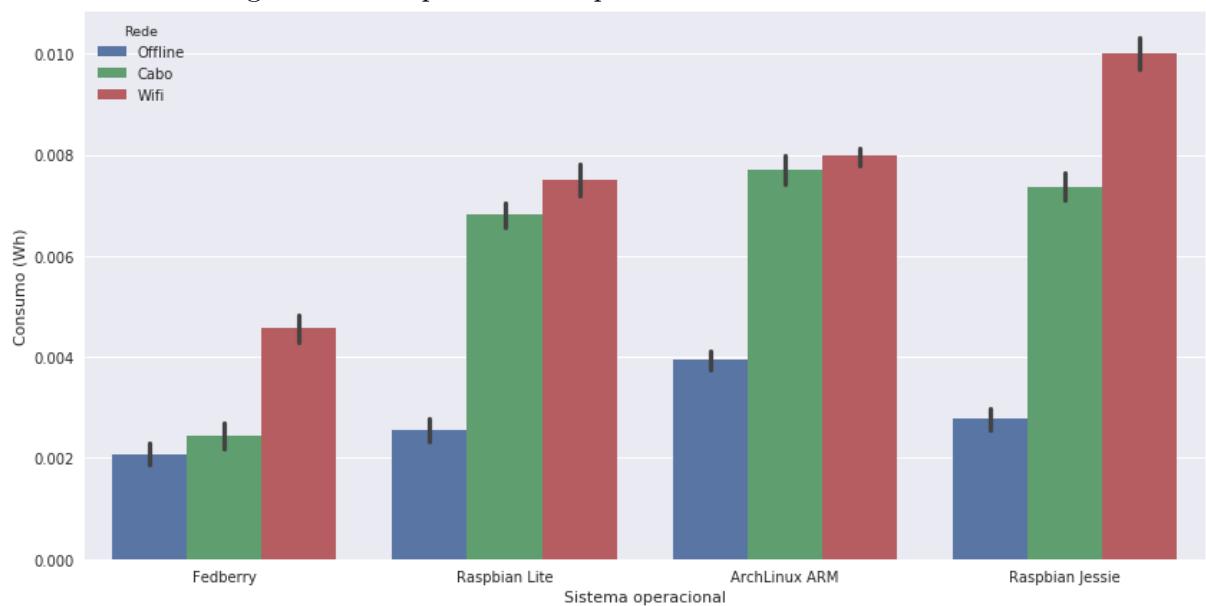
Fonte: Autor

Figura 19: Raspbian - Diagrama de energia de inicialização



Fonte: Autor

Figura 20: Comparativo de tipos de conectividade e consumo



Fonte: Autor

4.3.2 Conectividade

Para comparar os tipos de conexão que o Raspberry dispõe, utilizou-se o upload e download de um arquivo de 5 megabytes para um servidor próprio desenvolvido para testes.

REFERÊNCIAS

- AKYILDIZ, I. F.; MELODIA, T.; CHOWDHURY, K. R. A survey on wireless multimedia sensor networks. *Computer networks*, Elsevier, v. 51, n. 4, p. 921–960, 2007.
- ALMALKAWI, I. T. et al. Wireless multimedia sensor networks: current trends and future directions. *Sensors, Molecular Diversity Preservation International*, v. 10, n. 7, p. 6662–6717, 2010.
- BARONTI, P. et al. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards. *Computer communications*, Elsevier, v. 30, n. 7, p. 1655–1695, 2007.
- BROADCOM. BCM2836 ARM Peripherals. 2012. Disponível em: <raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/QA7_rev3.4.pdf>. Acesso em: 13 abr. de 2016.
- BURATTI, C. et al. An overview on wireless sensor networks technology and evolution. *Sensors, Molecular Diversity Preservation International*, v. 9, n. 9, p. 6869–6896, 2009.
- CALIN, D. The Raspberry Pi camera guide. 2015. Disponível em: <intorobotics.com/raspberry-pi-camera-guide>. Acesso em: 12 abr. de 2016.
- FISHER, R. et al. Open hardware: a role to play in wireless sensor networks? *Sensors, Multidisciplinary Digital Publishing Institute*, v. 15, n. 3, p. 6818–6844, 2015.
- GONÇALVES, D. d. O.; COSTA, D. G. A survey of image security in wireless sensor networks. *Journal of Imaging*, Multidisciplinary Digital Publishing Institute, v. 1, n. 1, p. 4–30, 2015.
- KAUP, F.; GOTTSCHLING, P.; HAUSHEER, D. Powerpi: Measuring and modeling the power consumption of the raspberry pi. In: IEEE. Local Computer Networks (LCN), 2014 IEEE 39th Conference on. [S.l.], 2014. p. 236–243.
- MAMMERI, A.; HADJOU, B.; KHOUMSI, A. A survey of image compression algorithms for visual sensor networks. *ISRN Sensor Networks*, Hindawi Publishing Corporation, v. 2012, 2012.
- MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A. *Handbook of applied cryptography*. [S.l.]: CRC press, 1996.
- MOOSAVI, S.; SADEGHI-NIARAKI, A. A survey of smart electrical boards in ubiquitous sensor networks for geomatics applications. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Copernicus GmbH, v. 40, n. 1, p. 503, 2015.
- NAVEENKUMAR, S.; PANDURANGA, H. et al. Partial image encryption for smart camera. In: IEEE. Recent Trends in Information Technology (ICRTIT), 2013 International Conference on. [S.l.], 2013. p. 126–132.
- PAPAGEORGIOU, P. Literature survey on wireless sensor networks. [S.l.]: Citeseer, 2003.

- PHAM, C. Low cost wireless image sensor networks for visual surveillance and intrusion detection applications. In: IEEE. Networking, Sensing and Control (ICNSC), 2015 IEEE 12th International Conference on. [S.l.], 2015. p. 376–381.
- POTTIE, G. J. Wireless sensor networks. In: IEEE. Information Theory Workshop, 1998. [S.l.], 1998. p. 139–140.
- RASPBERRY PI FOUNDATION. About the Raspberry Pi. 2016. Disponível em: <raspberrypi.org/about>. Acesso em: 13 abr. de 2016.
- RASPBERRY PI FOUNDATION. Raspberry Pi 2 Model B. 2016. Disponível em: <raspberrypi.org/products/raspberry-pi-2-model-b>. Acesso em: 13 abr. de 2016.
- SJOGELID, S. Raspberry Pi for Secret Agents. [S.l.]: Packt Publishing Ltd, 2015.
- SORO, S.; HEINZELMAN, W. A survey of visual sensor networks. Advances in Multimedia, Hindawi Publishing Corporation, v. 2009, 2009.
- TEXAS INSTRUMENTS. INA219: Zero-Drift, Bidirectional Current/Power Monitor With I₂C Interface. 2015. Disponível em: <<http://www.ti.com/lit/ds/symlink/ina219.pdf>>. Acesso em: 27 dez. de 2017.
- VUJOVIC, V.; MAKSIMOVIC, M. Raspberry pi as a wireless sensor node: Performances and constraints. In: IEEE. Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on. [S.l.], 2014. p. 1013–1018.
- VUJOVIĆ, V.; MAKSIMOVIĆ, M. Raspberry pi as a sensor web node for home automation. Computers & Electrical Engineering, Elsevier, v. 44, p. 153–171, 2015.
- WINKLER, T.; RINNER, B. Security and privacy protection in visual sensor networks: A survey. ACM Computing Surveys (CSUR), ACM, v. 47, n. 1, p. 2, 2014.
- YAP, F. G.; YEN, H.-H. A survey on sensor coverage and visual data capturing/processing/transmission in wireless visual sensor networks. Sensors, Multidisciplinary Digital Publishing Institute, v. 14, n. 2, p. 3506–3527, 2014.
- YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. Computer networks, Elsevier, v. 52, n. 12, p. 2292–2330, 2008.
- ZAINELDIN, H.; ELHOSSEINI, M. A.; ALI, H. A. Image compression algorithms in wireless multimedia sensor networks: A survey. Ain Shams Engineering Journal, Elsevier, v. 6, n. 2, p. 481–490, 2015.