

**UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

LUÍS HENRIQUE PUHL DE SOUZA

**HABILITANDO UM PRÉDIO A LOCALIZAR CONTEXTUALMENTE
DISPOSITIVOS UTILIZANDO REDES SEM FIO**

**BAURU
2016**

LUÍS HENRIQUE PUHL DE SOUZA

**HABILITANDO UM PRÉDIO A LOCALIZAR CONTEXTUALMENTE
DISPOSITIVOS UTILIZANDO REDES SEM FIO**

Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista “Júlio de Mesquita Filho” – UNESP, Câmpus de Bauru.

Orientador: Prof. Dr. Eduardo Martins Morigodo

Luís Henrique Puhl de Souza

Habilitando um Prédio a Localizar Contextualmente Dispositivos utilizando Redes Sem Fio/ Luís Henrique Puhl de Souza. – Bauru, 2016-
67 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Eduardo Martins Morgado

Monografia (Trabalho de Conclusão de Curso) –
Universidade Estadual Paulista “Júlio de Mesquita Filho”
Faculdade de Ciências - Campus Bauru
Departamento de Computação , 2016.

1. Localização. 2. Raspberry Pi. 3. Internet das Coisas. 4. Contexto I. Prof. Dr. Eduardo Martins Morgado. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". III. Faculdade de Ciências. IV. Habilitando um Prédio a Localizar Contextualmente Dispositivos utilizando Redes Sem Fio

LUÍS HENRIQUE PUHL DE SOUZA

HABILITANDO UM PRÉDIO A LOCALIZAR CONTEXTUALMENTE DISPOSITIVOS UTILIZANDO REDES SEM FIO

Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista “Júlio de Mesquita Filho” – UNESP, Câmpus de Bauru.

Aprovado em 13/02/2017.

BANCA EXAMINADORA

Prof. Dr. Eduardo Martins Morgado
Orientador

Profa. Dra. Simone das G. D. Prado

Profa. Dra. Roberta Spolon

RESUMO

ABSTRACT

LISTA DE FIGURAS

Figura 1 – Modelo das camadas	17
Figura 2 – Módulos ESP8266	28
Figura 3 – Sequência de ferramentas para implantação	30
Figura 4 – ESP-12f com regulador tensão e serial	31
Figura 5 – Código em C compilado e implantado em um ESP8266	32
Figura 6 – Raspberry Pi 3	34
Figura 7 – Antena cerâmica de Wi-Fi e Bluetooth do Raspberry Pi 3	37
Figura 8 – Adaptador Wi-Fi USB D-Link do laboratório	37
Figura 9 – Adaptador Wi-Fi USB Ralink Epub emprestado	37
Figura 10 – Adaptador Wi-Fi USB Ralink Epub comprado	37
Figura 11 – Interface do airodump-ng	38
Figura 12 – Arquitetura da aplicação	42
Figura 13 – MQTT Dashboard: Envio de mensagens	50
Figura 14 – MQTT Dashboard: Lista de inscrições	50
Figura 15 – MQTT Dashboard: Lista de mensagens no tópico	50
Figura 16 – MQTT.fx: Estatísticas do <i>Broker</i>	51
Figura 17 – mqtt-spy: Listagem de dispositivos	51
Figura 18 – Web APP	52
Figura 19 – Ambiente de teste	54
Figura 20 – Sinal em dBm por pacote capturado - 062722b3e5fb sensor 1	55
Figura 21 – Sinal em dBm por pacote capturado - 062722b3e5fb sensor 2	55
Figura 22 – Sinal em dBm por pacote capturado - 062722b3e5fe sensor 1	55
Figura 23 – Sinal em dBm por pacote capturado - 062722b3e5fe sensor 2	55
Figura 24 – dBm Pontos de acesso - Acumulado 8 horas	56
Figura 25 – Captura total (noise) - Teste 1	57
Figura 26 – Captura total (noise) - Teste 2	57
Figura 27 – dBm Motorola G4+ - Teste 1	58
Figura 28 – dBm Motorola G4+ - Teste 2	58

LISTA DE CÓDIGOS-FONTE

Código-fonte 4.1 – Ativação do modo monitor	36
Código-fonte 4.2 – iwconfig com modo monitor	36
Código-fonte 4.3 – TShark e opções	39
Código-fonte 4.4 – Adição do usuário pi ao grupo <i>wireshark</i>	39
Código-fonte 5.1 – TShark e opções executado pelo Node.js	43
Código-fonte 5.2 – Uso do fast-csv	44
Código-fonte 5.3 – Adição do pacote ao histórico do dispositivo	45
Código-fonte 5.4 – Extração das estatísticas do dispositivo	45
Código-fonte 5.5 – Cliente MQTT.js	46
Código-fonte 5.6 – Instalação e configuração do Mosquitto	49
Código-fonte 6.1 – TShark e redirecionamento da saída para arquivo assíncrono	53
Código-fonte 6.2 – TShark e redirecionamento da saída para arquivo	57

LISTA DE TABELAS

Tabela 1 – Descrição e custos de módulos ESP8266	29
Tabela 2 – Descrição e custos de acessórios para ESP8266	29
Tabela 3 – Ferramentas para desenvolvimento com ESP8266	30
Tabela 4 – Descrição e custos com Raspberry Pi 3	35
Tabela 5 – Comparação das plataformas ESP8266 e RPI3	40
Tabela 6 – Comparação de custos para o sensor da aplicação proposta em função da plataforma	41
Tabela 7 – Custos para o gateway da aplicação proposta	41

LISTA DE ABREVIATURAS E SIGLAS

AP	Ponto de Acesso Wi-Fi
API	<i>Application Programming Interface</i> , conjunto de definições para comunicação entre componentes de software
CSV	<i>comma separated values</i> - valores separados por vírgula
IoT	<i>Internet of Things</i> - Internet das Coisas
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
LTIA	Laboratório de Tecnologia da Informação Aplicada
MAC	<i>Media Access Control</i> - protocolo que coordena endereços de máquina a nível da camada de enlace de dados
RPI3	Raspberry Pi 3 model B
SSH	<i>Secure Shell</i> - Conexão segura entre terminais <i>bash</i>
Wi-Fi	Marca registrada da Wi-Fi Alliance. Rede local sem fios baseados no padrão IEEE 802.11

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Problema	13
1.1.1	Sobre Sistemas de Posicionamento	13
1.2	Motivação	16
1.3	Objetivos	18
1.3.1	Objetivo Geral	18
1.3.2	Objetivos Específicos	18
2	FUNDAMENTAÇÃO TEÓRICA	19
2.1	Internet das coisas (IoT)	19
2.2	Localização contextual de dispositivos	19
2.2.1	Localização contextual	20
2.2.2	Contexto de um dispositivo em um prédio	21
2.3	Localização baseada em redes sem fio	21
2.4	Trabalhos correlatos	22
2.4.1	Zebra	22
2.4.2	Outras tentativas	23
3	MÉTODO DE PESQUISA	24
4	PLATAFORMAS	26
4.1	ESP8266	27
4.1.1	Disponibilidade no mercado	27
4.1.2	Desenvolvimento e Implantação	28
4.1.3	Testes e resultados	30
4.2	Raspberry Pi	33
4.2.1	Disponibilidade no mercado	33
4.2.2	Desenvolvimento e implantação	35
4.2.3	Testes e resultados	36
4.3	Escolha e conclusão	40
5	CONSTRUÇÃO	42
5.1	Sensor	43
5.2	Gateway	49
5.3	Apresentação Web	52

6	RESULTADOS E DISCUSSÃO	53
6.1	Método de teste	53
6.2	Avaliação de ruído e consistência	54
6.3	Teste relação de distância com smartphone como objetivo	57
7	CONCLUSÃO	59
	REFERÊNCIAS	60
	APÊNDICES	65
	APÊNDICE A – COMPRAS MERCADO LIVRE	67

1 INTRODUÇÃO

Nos recentes anos de 2014 a 2016, a Internet das Coisas (IoT - *Internet of Things*) vem tomado o foco das atenções de empresas e entusiastas de Tecnologia da Informação (DZONE, 2015) e, como é esperado que uma quantia total de 6,4 bilhões de dispositivos conectados exista até o final de 2016 (GARTNER, 2015) e entre 26 bilhões (GARTNER, 2014) e 50 bilhões até 2020 com até 250 novas coisas conectando-se por segundo (Cisco Blog, 2013), as empresas líderes do segmento já incluem IoT como uma de suas áreas de atuação (IBM, 2016; ARM, 2016; MICROSOFT, 2016; INTEL, 2016; ORACLE, 2016; GOOGLE, 2016; AMAZON, 2016a).

Todo este movimento no mercado é justificado pelo baixo custo dos pequenos dispositivos computacionais (FOUNDATION, 2015; ESP8266.NET, 2016) e grandes serviços na nuvem (KAUFMANN; DOLAN, 2015; AMAZON, 2016b). Este baixo custo possibilita a computação ubíqua descrita por Weiser (1999) que nesta obra é entendida como “*computação onipresente diluída no dia-a-dia*”. Também nesta obra, esta onipresença diluída no plano de fundo é a base e a consequência para o conceito e área de IoT, sendo esta a realizadora da computação ubíqua.

Uma vez contextualizado o mercado e a oportunidade de implementação da computação ubíqua, percebe-se a necessidade de dar aos elementos cotidianos (coisas) a capacidade info-computacional, tornando-os sensores e atuadores conectados, unicamente identificáveis e acessíveis através da rede mundial de computadores (LEMOS, 2013; KRA-NENBURG, 2012). Para tanto, este trabalho propõe a construção de um sensor que, através da rede, identifica e localiza contextualmente os elementos cotidianos.

1.1 Problema

Tamanha quantidade de dispositivos conectados pouco acrescenta na vida diária se humanos ou coisas não puderem simplesmente se encontrar. Tanto em ambiente real quanto virtual é essencial o contato e conhecimento entre as partes envolvidas para que uma interação complexa seja executada. Portanto, para que uma aplicação IoT funcione corretamente, o conhecimento do contexto em que todos os interessados, sejam coisas ou pessoas, estão inseridos é indispensável. Para a maioria das aplicações, a informação contextual de maior relevância é a localização física.

Em situações em que a localização contextual é essencial para o bom funcionamento de uma aplicação IoT, destaca-se a necessidade da coleta desta informação através de sensores ativos sempre que a aplicação requisite a ciência deste contexto em suas tomadas de decisão. E, também, para que outros (sistemas, pessoas e coisas) saibam a localização de qualquer dispositivo ao qual têm interesse de interagir, distribuindo efetivamente essa informação coletada sobre o contexto com todos os que se encontram envolvidos no mesmo contexto.

Um exemplo desta necessidade de localização de dispositivos dentro de um prédio seria um profissional saber onde está o dispositivo em seu local de trabalho, seja ele um vendedor e seu *tablet* para demonstrar um produto fora de estoque em uma loja ou um médico e seu equipamento portátil.

1.1.1 Sobre Sistemas de Posicionamento

Sistemas de posicionamento (PS - *Positioning System*) são geralmente constituídos de um Ponto Origem Global escolhido (O) e um conjunto não vazio de Pontos de Referência (RP - *Reference Point*) cuja localização global em relação ao O é conhecida com uma certa precisão quando o sistema é construído - precisão interna. Então, para o usuário, um sistema de posicionamento oferece como resultado uma precisão de visualização menor que a sua precisão interna. Um PS tem interesse em determinar a posição de um ponto móvel (MU - *Mobile User*). Essa localização é feita encontrando um conjunto de distâncias associadas a cada um dos RPs em um sub-conjunto com dimensão variável de acordo com o método utilizado. Feito isso, é possível utilizar modelos matemáticos para, a partir das distâncias, encontrar uma posição do MU em relação aos RPs e uma nova transformação é aplicada para encontrar a posição relativa ao O .

Uma das maneiras de classificar PSs é entre as classes de Auto Posicionamento e Posicionamento Remoto. Os de Auto Posicionamento contém no MU todo aparato necessário para medir a distância dos RPs e calcular a posição em relação a O . Já os classificados como de Posicionamento Remoto tem o mínimo necessário na MU e todo o trabalho de cálculo de distância e posição global é feito nos RPs ou em uma unidade coordenadora

destes.

Para PSs eletrônicos baseados em radio-frequência (RF - *Radio Frequency*), geralmente, utilizam-se dois componentes básicos, Transmissores e Receptores, os quais assume-se que ao menos um destes está no RP e ao menos um outro no MU. Para calcular a distância entre MU e RP, utiliza-se as propriedades da comunicação por RF como tempo de chegada (TOA - *Time Of Arrival*), diferencial de tempo de chegada (TDOA - *Time Difference Of Arrival*) e ângulo de chegada de sinal (AOA - *Angle Of Arrival*).

Para maior precisão, é comum a utilização de múltiplas RPs geralmente com o número mínimo igual ao número de dimensões espaciais que deseja-se calcular. Nota que para sistemas distribuídos a sincronização de relógios é um problema intrínseco, então é fundamental que o tempo seja contado como dimensão.

Os sistemas classificados como “Sistema de Navegação Global por Satélite” (GNSS - *Global Navigation Satellite System*), como o tradicional estadunidense Sistema de Posicionamento Global (GPS - *Global Positioning System*), utilizam a técnica em que o dispositivo móvel contém o receptor e os transmissores são fixos em satélites na órbita terrestre (DJUK-NIC; RICHTON, 2001). Devido a posição e número de satélites, o GPS e seus correlatos estão sempre presentes do ponto de vista de um observador da superfície terrestre, sendo para este tipo de usuário um sistema ubíquo.

Entretanto, a força do sinal GNSS não é suficiente para penetrar a maioria dos prédios, uma vez que estes dependem de visão direta (LOS - *Line-Of-Sight*) entre os satélites e o receptor. A reflexão do sinal muitas vezes permite a leitura em ambientes fechados, porém o cálculo da posição não será confiável (CHEN; KOTZ, 2000). Portanto, apesar da ubiquidade dos GNSSs em ambientes abertos, são necessárias soluções diferentes para obter um Sistema de Posicionamento para Ambientes Fechados (IPS - *Indoor Positioning System*), sendo a ubiquidade deste essencial para conquistar o mesmo nível de confiança trazido pelos GNSSs.

Para implementar este IPS, propõem-se o uso de tecnologias já implantadas em dispositivos móveis e essenciais para o funcionamento dos mesmos, especialmente as de camadas de comunicação, que são ubíquas no ambiente dos dispositivos móveis, como *Wi-Fi* (padrão *IEEE 802.11*) e *Bluetooth* (padrão *Bluetooth SIG*), para que os objetos conectados os quais tem-se interesse de encontrar a localização contextual não necessitem de modificações.

Outros protocolos de comunicação sem fio ubíquos existem (em especial, os celulares em todas as gerações 2G, 3G, 4G), porém não oferecem a mesma flexibilidade por trabalharem em uma faixa de radio-frequência licenciada e por questões de propriedade da rede que serão abordadas na seção de Localização Contextual desta mesma obra.

De forma semelhante, existem protocolos mais flexíveis (nas faixas não licenciadas

como *NFC*, infra-vermelho, *ZigBee* ou *SIGFOX*), porém estes não estão presentes na maioria dos aparelhos utilizados, tanto globalmente quanto localmente, removendo a característica da forma de comunicação ubíqua que é foco deste trabalho.

Devido às restrições anteriores, justifica-se o foco nas tecnologias de comunicação *Wi-Fi* e *Bluetooth*, porém trabalhar com as duas tecnologias simultaneamente é um problema complexo por si só, então, a escolha de um ou outro deve ser feita, apesar de a nível global serem de equivalente importância para esta obra (ambas tem mesma importância e presença no mercado atual, permitem flexibilidade por possuírem protocolos conhecidos por todos em frequências livres de licenciamento, dentro da área de cobertura que são de nosso interesse e o usuário final já ser o proprietário da rede local criada). Esta escolha toma um único parâmetro como decisivo que é a observação do ambiente de teste do protótipo onde pouco existe o uso de *Bluetooth* que reflete o costume local de mantê-lo desligado em comparação com *Wi-Fi* que está sempre ligado em todos dispositivos, conectando estes diretamente à Internet. Portanto *Wi-Fi* é a tecnologia de maior interesse por pequena margem.

1.2 Motivação

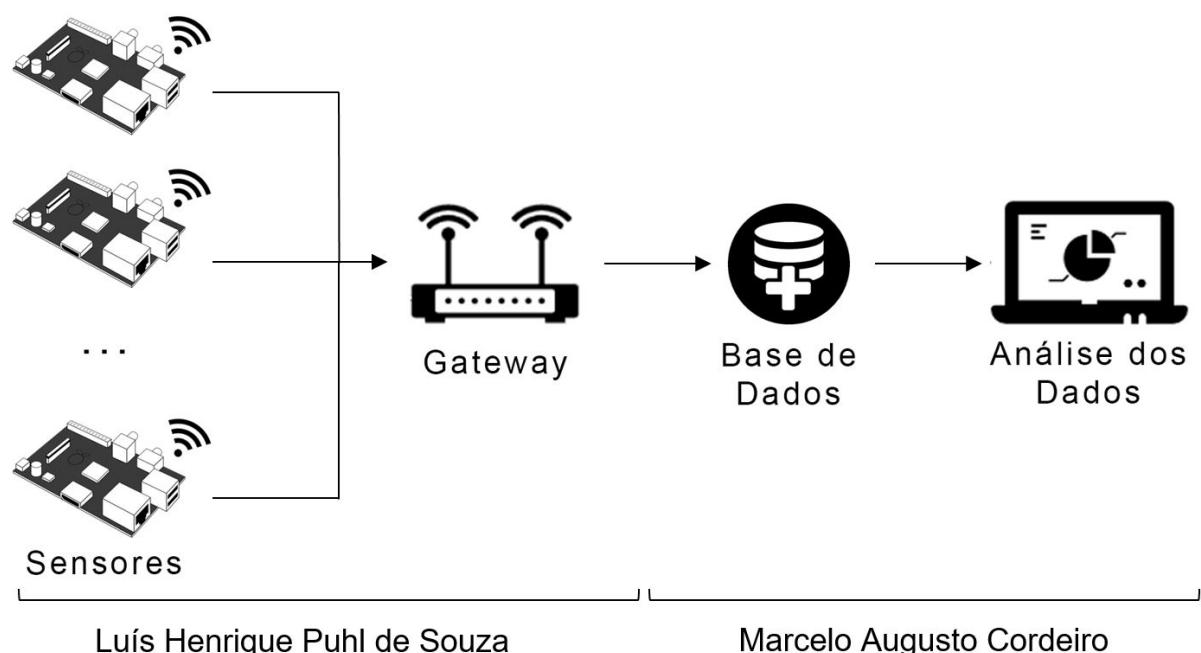
A proposta deste trabalho é criar um ambiente consciente, onde o contexto locativo oriundo do posicionamento remoto de cada dispositivo móvel é administrado e divulgado pelo prédio conectado ao invés da auto-localização do aparelho, pois:

- a) Uma vez encontrada a localização, é mais fácil propagar esta informação do ambiente para o aparelho em comparação ao autoposicionamento, pois a negociação entre o ambiente e o aparelho é nula quando o primeiro contém a informação - o ambiente sempre disponibilizará uma informação coletada para o gerador desta informação;
- b) Pode-se lidar com grande heterogeneidade de dispositivos, uma vez que cada um deles não precisa se adaptar para cada mudança de ambiente;
- c) Este tipo de informação já é contida nos históricos de cada Ponto de Acesso Wi-Fi (AP - *Access Point*), porém:
 - Geralmente sem uso - poucas são as aplicações que usam a localização obtida pelo AP;
 - Com granularidade insuficiente para uso em aplicações contextualizadas;
 - geralmente não disponibilizada pelos APs.
- d) Uma vez instalado um PS deste gênero, a quantia de dispositivos que ele pode localizar fica limitada apenas pela rede física anteriormente instalada;
- e) Economia de hardware quando menos é exigido de cada dispositivo móvel.

Nota-se também que mesmo com a quantidade prevista de 5 dispositivos IoT por pessoa em média, estes seriam beneficiados sempre que utilizados no ambiente conectado proposto.

A Figura 1 apresenta a arquitetura simplificada de uma aplicação IoT, e no detalhe inferior a relação deste projeto com o do aluno Marcelo Augusto Cordeiro, também do Bacharelado de Ciências da Computação, que é também membro do ambiente de testes LTIA (Laboratório de Tecnologia da Informação Aplicada) da Unesp de Bauru e do mesmo edital para obter o título de bacharel.

Figura 1 – Modelo das camadas



Fonte: Marcelo Augusto Cordeiro (??)

1.3 Objetivos

1.3.1 Objetivo Geral

Considerando características locais, propõem-se a construção de uma aplicação para localizar contextualmente dispositivos dentro de um prédio piloto e avaliar sua precisão.

Além da aplicação, é objetivo definir o custo do projeto piloto, incluindo esforço de pesquisa assim como definir um custo para replicação deste localizador contextual em outros prédios utilizando como fonte de ferramentas e recursos o mercado local.

1.3.2 Objetivos Específicos

- a) Estabelecer o estado da arte sobre a desenvolvimento de aplicações IoT;
- b) Identificar desafios locais para o desenvolvimento;
- c) Identificar provedores de serviços, dispositivos e ferramentas para o desenvolvimento;
- d) Construir sensores de identificação e localização (distância) de dispositivos cuja comunicação seja baseada em *Wi-Fi*;
- e) Posicionar estes sensores;
- f) Construir um dispositivo agregador de informações dos sensores (*gateway*) e sua interface web (*MQTT - MQ Telemetry Transport*);
- g) Estimar o custo total do projeto piloto incluindo esforço de pesquisa;
- h) Estimar o custo de replicação da aplicação em outros prédios utilizando fontes do mercado local.

2 FUNDAMENTAÇÃO TEÓRICA

Para conceituar, fundamentar e dar suporte teórico ao presente trabalho apresentam-se neste capítulo os tópicos e definições dos segmentos: IoT, localização contextual de dispositivos e localização baseada em redes sem fio.

2.1 Internet das coisas (IoT)

Uma das primeiras aplicações e definições de IoT foi feita simultaneamente por Kevin Ashton em 1999 para a *Procter & Gamble* (P&G) (ASHTON, 2009) e pelo laboratório Auto-ID Labs no Instituto de Tecnologia de Massachusetts (MIT - *Massachusetts Institute of Technology*) utilizando identificação por radio-frequência (RFID - *radio-frequency identification*) (ATZORI; IERA; MORABITO, 2010; FRIEDEMANN; FLOERKEMEIR, 2011). Desde então, a IoT cresceu ultrapassando o escopo da tecnologia RFID, porém sempre com as premissas de “uma infraestrutura global para a Sociedade da Informação, habilitando serviços avançados através da interconexão de coisas (físicas e virtuais) baseadas em tecnologias, existentes e evolutivas, de informação e comunicação” descrita por Wortmann e Flüchter (2015 apud International Telecommunication Union, 2012, p. 1, grifo e tradução nossa).

Hoje em dia, quase qualquer tecnologia de comunicação acessível a computadores pode ser utilizada como meio de comunicação entre dispositivos IoT, tornando o RFID mais uma, porém de grande importância, tecnologia info-comunicacional a disposição das coisas para sua conexão. Esta gama de tecnologias possibilita uma variedade equivalente de coisas conectadas. Se a coisa pode usar de uma tecnologia de conexão, considerando suas restrições de volume, custo e utilidade, muito provavelmente vai fazê-lo gerando ao menos uma identidade virtual representando seu objeto físico e seus atributos. Esta identidade virtual e atributos virtuais serão expostos para todos indivíduos, humanos ou coisas, que lhe forem convenientes de qualquer lugar do universo virtual, fazendo efetivamente parte da Internet.

2.2 Localização contextual de dispositivos

Em ciência da computação, os termos “*Contexto*” e “*Consciência de Contexto*” expressam uma ideia recente estudada nos campos de inteligência artificial e ciência cognitiva desde 1991. O tema “Contexto” ainda é considerado atual e promissor a ponto de mudar o cenário de negócios nos próximos 10 anos, mas sem definição simples. Tamanha é a falta de uma definição geral que realmente funcione para casos reais que existe uma

proposta de definir o termo utilizando uma nova metodologia de pesquisa holística através de mineração e agrupamento de texto advindo de publicações científicas (PASCALAU; NALEPA; KLUZA, 2013).

Mesmo sem uma definição permanente em vista, utilizou-se o que é considerado estado da arte para o termo "*Contexto*" que foi introduzido por Dey e Abowd (1999) e reforçado por Dey (2000):

Contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação.

Dey e Abowd (1999, p. 3) Tradução Nossa.

2.2.1 Localização contextual

Das informações contextuais que uma aplicação de cliente móvel pode obter, a localização é uma das mais importantes. Ajudar pessoas a navegar por mapas, encontrar objetos e pessoas com os quais tem interesse de interagir é sem dúvida uma boa meta a ser alcançada com a coleta da localização do cliente (BELLAVISTA; KÜPPER; HELAL, 2008).

Na categoria de Serviços Baseados em Localização (LBS - *Location-Based Services*) existem duas gerações. A primeira orientada a conteúdo que falhou, pois a informação de localização era armazenada pela rede (que geralmente era administrada por uma empresa de telecomunicações), podendo até ser vendida pelo provedor a terceiros, causando a sensação de *Spam* (conteúdo não solicitado) no usuário final ao receber conteúdo desta provedora. Já na segunda geração, a posse da informação foi movida para o cliente móvel, deixando a cargo do usuário escolher se ela seria compartilhada e com quem. Esta mudança trouxe maior engajamento do usuário, resultando numa maior aceitação dessa geração (BELLAVISTA; KÜPPER; HELAL, 2008).

Ao contrário das técnicas atuais, neste trabalho os humanos ou tomadores de decisão não estarão em posse do cliente móvel, e sim em posse do prédio. Portanto, a mesma informação, sem degradação em sua importância, passará a ser coletada e armazenada pelo provedor da rede como nos LBSs de primeira geração. Esta decisão garante o foco no usuário uma vez que este mudou, antes ele detinha um cliente móvel, agora ele detém múltiplos. Isso torna a detenção do todo (coisas dentro do prédio) mais precioso do que as partes (os clientes móveis) além da mudança da propriedade da rede para o usuário final, na comparação celular *versus Wi-Fi*.

Uma vez encontrada a localização de um dispositivo, metadados sobre o prédio são mesclados formando um conjunto rico contextualmente do ponto de vista da aplicação IoT Prédio como fornecedora principal dos dados para a Internet e, portanto, seus

usuários detentores. Essa riqueza é garantida com metadados sobre o dispositivo (identificação, nome, histórico, características) e sobre o prédio (ex.: mapa, estrutura de salas, humanos responsáveis e lista de equipamentos) que trazem possibilidades de extração de informação importantes para os detentores deste prédio e seu conteúdo. Esta capacidade do prédio deve-se pelo papel de coordenador de informações e controlador de meta-informações semelhante ao Coordenador em uma aplicação na arquitetura Modelo-Apresentação-Adaptador-Controlador-Coordenador (MPACC - *Model-PresentationAdapter-Controller-Coordinator*) proposto por Román e Campbell (2001).

2.2.2 Contexto de um dispositivo em um prédio

Para metadados agregados à informação de posição pelo prédio defini-se que, para uma aplicação IoT, o modelo de divulgação tem de conter além da posição do dispositivo informação sobre este (nome, histórico), informação da estrutura do prédio, ligação entre a estrutura do prédio e a localização do dispositivo e informação sobre o estado do prédio.

Este modelo visa prover fácil mineração e reutilização de informações por terceiros que é medida pela disponibilidade e relacionamento das informações providas. Essa métrica também será utilizada para avaliar o projeto.

Este foco em reusabilidade vem da definição de Web Semântica (*Semantic Web*) e de uma de suas realizadoras, a Ligação de Dados (*Linked Data*), que sugerem o uso de um formato padrão além de ser acessível e gerenciável pelas ferramentas de exploração. Desta forma a Web de Dados (*Web of Data*) é construída opondo uma simples coleção de dados (BIZER; HEATH; BERNERS-LEE, 2009).

2.3 Localização baseada em redes sem fio

Um sistema de posicionamento pode ser baseado em técnicas *n-lateração*? de distâncias adquiridas com a medição de características eletromagnéticas (ex.: potência de sinal) e dos protocolos (ex.: Tempo de chegada) que já foram explorados anteriormente (ABUSUBAIH; RATHKE; WOLISZ, 2007; BAHILLO et al., 2009; FELDMANN et al., 2003).

Portanto, os sensores seguem as especificações de *Wi-Fi IEEE 802.11* (CROW et al., 1997) e técnicas definidas para *Bluetooth Low Energy (BLE)* (HOSSAIN; SOH, 2007) devido a semelhança da área de cobertura (até 100 metros, geralmente utilizado até 20 metros) e frequência (no caso de 2.4GHz).

Para construir estes sensores uma plataforma de hardware adequada é necessária, para esta escolheu-se o Raspberry Pi (VUJOVIC et al., 2014; VUJOVIĆ; MAKSIMOVIĆ, 2015) que já foi provado funcional no caso de Localização através *Wi-Fi* por Ferreira (2016) especialmente a sua versão 3 que adiciona a capacidade de sensor *Wi-Fi* e *Bluetooth* em

sua placa principal sem necessidade de adaptadores externos destacando ainda mais sua escolha (RASPBERRY PI FOUNDATION, 2016). Em adição, na construção dos sensores foi testada a plataforma ESP8266 bem como outras alternativas que demonstraram afinidade com essas características.

2.4 Trabalhos correlatos

Neste subcapítulo, apresentaremos alguns projetos semelhantes em objetivo ao daqui proposto e que motivaram a construção do sensor resultante deste trabalho.

2.4.1 Zebra

A Zebra é um empresa estadunidense que fabrica e vende tecnologia de marcação, rastreamento e impressão por computador. Dentre os seus produtos, estão: computadores móveis, RFID, software, impressoras, tablets, leitores de códigos de barras, kiosks interativos, entre outros. Já na área de serviços, a empresa oferece desde o planejamento até a execução de projetos.

A Zebra realizou um estudo (Global Shopper Study) que indicou que os varejistas apostaram em recursos online que podem aumentar o envolvimento e fidelidade do consumidor, além, claro, do volume de vendas. Segundo o mesmo estudo, 51 compradores tem um forte interesse em serviços baseados em localização e *Wi-Fi* em lojas para cupons *mobile*, mapas de compras e receber assistência. Além disso, 64 mais itens se receberem um serviço melhor e mais atenção dos vendedores, enquanto mais da metade prefere que os varejistas usem a tecnologia para criar experiência de compra mais eficiente.

A empresa possui o projeto MPact que é um *indoor location* que unifica *Wi-Fi* e *Bluetooth*. Ele fornece a localização do consumidor em três níveis: presença, zona e posição. Com estas informações é possível saber sobre o indivíduo: quem é, onde está, quanto tempo fica em certas áreas e quais produtos está comprando. Esta tecnologia pode ser implementada independente do ambiente, através do *Wi-Fi*, ou do microposicionamento através do *Bluetooth*. Com a união dessas duas plataformas é possível saber o tempo exato e posição exata de onde alguém está.

Em 2016, a empresa implantou no Shopping Cidade Jardim, em São Paulo, uma rede LAN/WAN sem fio de alta velocidade, com a tecnologia MPact que proporciona aos seus clientes acesso gratuito ao *Wi-Fi*, juntamente com uma experiência de compra mais personalizada. Funciona assim, o consumidor assim que possível acessa o *Wi-Fi* do shopping que pede o *login* no Facebook ou Google. Assim que o *login* é feito, o MPact fornece visibilidade instantânea ao operador do shopping em relação aos locais dos compradores no shopping e habilita os operadores a enviar saudações pessoais, oferecer ofertas especiais ou fornecer instruções passo-a-passo para um venda ou promoção

específica. Hoje, o Cidade Jardim conta com 180 lojas numa área de 46.000 metros quadrados.

Segundo Claudio Bessa, diretor de marketing digital do shopping, este tipo de serviço fornece um excelente experiência para os compradores devido a alta velocidade do *Wi-Fi* e a cobertura. Além disso, segundo ele, os operadores e varejistas podem entender melhor o comportamento do consumidores, pois eles podem saber que parte do corredor ou de uma loja o cliente está, quanto tempo permanece na frente de uma loja e quais produtos mais vendem. Oferecer este tipo de serviço é uma maneira de ganhar e manter consumidores, crescer no número de satisfações e ajudar a monitorar os pontos de venda.

2.4.2 Outras tentativas

Outras tentativas bem sucedidas de localizar dispositivos móveis através da rede *Wi-Fi* são o caso de Vasisht, Kumar e Katabi (2016) e de Lanzisera, Zats e Pister (2011).

No primeiro exemplo, uma placa de rede sem fio Intel 5300 com três antenas calcula o tempo de vôo entre uma antena e outra além de utilizar técnicas de mitigação de multicaminho, mitigação de identificação de pacote entre outras características importantes do protocolo *Wi-fi*, como a frequência e sincronização de clientes para alcançar até 10 centímetros de precisão. Neste caso, as três antenas atuam como três sensores independentes justa posicionados para executar trilateração. Esta aplicação é implementada em uma placa instalada em um computador moderno através do barramento PCI Express com sistema operacional Ubuntu. Ela possui habilidade de injetar pacotes na rede o que difere muito das arquiteturas embarcadas que normalmente são encontradas no ambiente de IoT.

O segundo exemplo de aplicação bem sucedida se utiliza de modificações no hardware de um ponto de acesso do padrão *IEEE 802.15.4* e alcança precisões de 1 a 3 metros. Este protocolo é mais encontrado em comunicações de longa distância ou sensíveis a uso de energia que são frequentes em aplicações embarcadas.

3 MÉTODO DE PESQUISA

Abordagens para medir distâncias através de redes sem fio *Wi-Fi* (BAHILLO et al., 2009) e *Bluetooth* já existem e, propor novas maneiras não é o foco deste trabalho. Utilizando essas técnicas, constitui-se uma rede de nós sensores colaborativos fixos no ambiente onde deseja-se obter a localização dos dispositivos. As informações de distância são compartilhadas entre os nós para maior precisão da informação.

Para a implementação, utilizou-se os *softwares* de maior destaque recentemente nos ramos de comunicação de baixa energia (*MQTT*), serviços *Web* para geolocalização (*Google Maps*) e publicação (*NodeJS*), além de *softwares* para medição da distância sem interferir na comunicação (*Sniffing*) e das plataformas de *hardware* disponíveis e recomendadas para IoT com capacidade *Wi-Fi* (*Raspberry Pi 3* e *ESP-8266*).

Mesmo com a grande quantidade de dispositivos já conectados são poucos os documentos descrevendo boas práticas para concepção, construção e manutenção de aplicações IoT, especialmente sobre os cuidados tomados quanto a segurança e análise de custos para a implementação e manutenção. Além disso, a falta de referências neste sentido é agravada quando considera-se a implementação no interior do estado de São Paulo. Nesta região, poucas são as organizações atualizadas neste tema, levando a uma falta enorme de conteúdo escrito na linguagem local além de serviços e produtos disponíveis para construção de uma plataforma completa e competitiva na região.

Devido a falta de conteúdo e instrução, utiliza-se prototipagem ágil neste projeto, uma vez que esta metodologia de desenvolvimento é recomendada para projetos cujas especificações e definições não são claras, demandando muitas modificações das mesmas durante a etapa de execução. Esse método entra em contraste com metodologias clássicas, como a cascata, que apesar de previsíveis, não reagem bem a ambientes de extrema incerteza.

Mais especificamente, utiliza-se uma variante da metodologia *Scrum* (JAMES, 2016) que foi adaptada para o projeto. Nela, foram executadas iterações de uma semana em que a cada iteração, uma nova versão melhorada do produto completo (*hardware*, *software*, documentação e resultados) foi feita.

Dentro de cada iteração, as camadas da aplicação IoT foram escolhidas, implementadas, justificadas e avaliadas, sendo parte do processo registrado sob forma de vídeo (Youtube).

A cada iteração, cumpriu-se parte ou todo de cada objetivo proposto no trabalho, levando o projeto gradualmente para um estágio de completude. Cada iteração teve como

foco os objetivos a seguir, sendo seus resultados utilizados para tomar e justificar decisões durante a execução do projeto bem como servir de posterior documentação. Os objetivos de cada iteração são:

- a) Escolha de provedores de serviços, dispositivos e ferramentas para o desenvolvimento;
- b) Construir, avaliar, testar e manter os sensores;
- c) Construir o dispositivo agregador e sua API;
- d) Estimar o custo total do projeto piloto;
- e) Estimar o custo de replicação;
- f) Identificar os desafios para o desenvolvimento.

Desta forma, a liberdade necessária foi garantida para o projeto ser executado com sucesso, mesmo no ambiente de incerteza no qual o mercado local de IoT encontra-se, cumprindo as premissas de funcionamento, manutenção e segurança que são grande importância para os interessados na área.

4 PLATAFORMAS

Para a localização com os resíduos de comunicação *Wi-Fi* são necessários plataformas que possam capturar estes resíduos e processar qualquer informação capturada. Esta plataforma de sensor pode ser construída com qualquer plataforma computacional capaz de ser programada com comunicação *Wi-Fi*, porém o *hardware* de *Wi-Fi* e seu *software* controlador deve permitir o Modo Promíscuo.

Este Modo Promíscuo (*promiscuous mode*) é definido pela capacidade de uma Placa Adaptadora de Rede *Wi-Fi* (*Network Interface Card - NIC*) receber e interpretar todos os pacotes que trafegam em uma rede ou em todas as redes que estão em seu alcance, independentemente do destinatário do pacote. Em seu funcionamento normal, uma *NIC* descarta todos os pacotes que não são destinados a ela o mais cedo possível, evitando reprocessamento de dados indesejáveis, por este motivo não são todas as *NICs* que permitem o Modo Promíscuo. Essa funcionalidade elimina a necessidade de *hardware* ou *software* em cada um dos dispositivos rastreados.

Neste sentido, elegeu-se duas plataformas de notável importância no mercado atual e notável facilidade de acesso para qualquer interessado na área. As plataformas testadas foram o microcomputador *Raspberry Pi* e o microcontrolador *ESP8266*. Ambos foram escolhidos pelo domínio do segmento de Prototipação e Faça Você Mesmo (*Do It Yourself - DIY*) dentro do campo de IoT. Outro líder de segmento, o *Arduino* foi prontamente descartado por não conter nativamente a habilidade de conectar-se à *Internet* sendo constantemente combinado com um dos escolhidos para ganhar esta habilidade, demonstrando claramente menor afinidade a este projeto em comparação aos seus igualmente famosos concorrentes.

Após escolhidas as plataformas de interesse, alguns exemplares de cada uma delas foram adquiridos para implementar a aplicação proposta. Neste sentido, serão apresentadas cada uma dessas plataformas quanto as suas especificações técnicas, os produtos utilizados em conjunto para que elas pudessem funcionar e serem programadas e os motivos pela adoção ou não delas.

4.1 ESP8266

O ESP8266 é um SOC (*System On a Chip* - Sistema em um *Chip*), ou seja, é um chip com todos os componentes lógicos eletrônicos necessários e partes para um dado sistema em único circuito integrado. Este chip possui:

- a) *Wi-Fi* embutido com capacidade de 2,4 GHz (802.11 b/g/n);
- b) 16 GPIOs (*general-purpose input/output*) incluindo interfaces I2U, SPI, UART, entrada ADC, saída PWM;
- c) Arquitetura *RISC* de 32 bits;
- d) CPU que opera em 80 MHz, com possibilidade de operar em 160 MHz;
- e) 64 KB de ROM para *boot*;
- f) 64 KB de RAM para instruções;
- g) 96 KB de RAM para dados;
- h) Memória *Flash SPI* de 512 KB a 4 MB (dependente de módulo externo);
- i) Núcleo baseado no *IP Diamond Standard LX3* da *Tensilica*.

Para o mercado de prototipação, fabricantes constroem placas de diferentes configurações com este chip como elemento central, os chamados módulos. Estes módulos usam o ESP8266 com diferenças perceptíveis, por exemplo, quantidade de pinos, dimensões físicas, alguns podem até operar de modo *standalone* (sem outro *hardware* de suporte como reguladores de tensão e conversores serial-USB) e, especialmente, a Memória *Flash SPI*. Neste trabalho, foram usados os módulos: *ESP-01*, *LoLin*, *D1 mini* e *ESP-12f* com placa adaptadora de pinos. Cada um deles pode ser encontrado na Figura 2 a seguir.

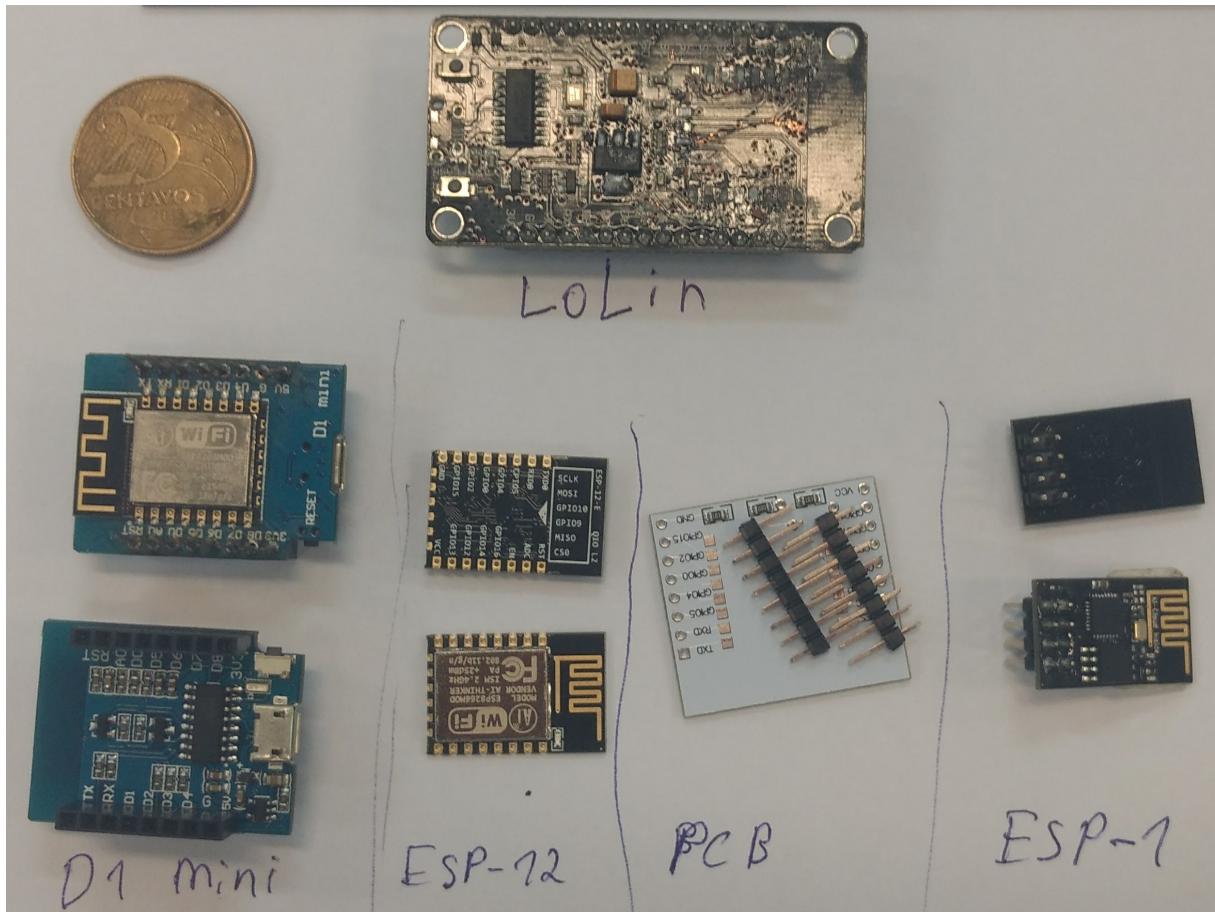
4.1.1 Disponibilidade no mercado

As diferentes especificações implicam em diferentes produtos e mercado para eles, isto resulta em diferentes custos em diferentes regiões.

O ESP8266 foi escolhido como primeira tentativa devido ao seu baixo custo e ao tamanho reduzido. No exterior, ele pode ser encontrado de USD\$ 1,76 a 2,2 (alibaba.com, 2017), e no Brasil por aproximadamente BRL R\$ 15,00 (mercadolivre.com.br, 2017a).

Devido ao seu tamanho, ele é de fácil integração com demais dispositivos, bastando o uso de uma comunicação serial. Já sobre a comunidade, há inúmeros projetos DIY (em inglês *do it yourself*, em português "faça você mesmo") que ensinam a como construir e manipular projetos que envolvem diferentes módulos. Além disso, a empresa idealizadora e fabricante do chip, Espressif, disponibiliza no GitHub projetos com documentação e código aberto.

Figura 2 – Módulos ESP8266



Fonte: Elaborada pelo autor

Para desenvolver na plataforma, os módulos ESP foram utilizados de formas diferentes dependendo das capacidades de um deles. Quando o módulo possuía regulador de tensão embarcado, utilizava-se o próprio conectado a uma porta USB. Quando o módulo não possuía tal, utilizava-se um circuito com fonte externa (pilhas ou USB) e um regulador de tensão conectados aos pinos *3v3* e *GND*. Dependendo da complexidade do circuito para ligar e ter acesso à serial do módulo, foi necessário o uso de uma placa *breadboard*, como na Figura 4. Para este trabalho foi utilizado o regular *AMS1117 3v3* e dois capacitores de $100\mu F$. Na Tabela 1 e Tabela 2, são apresentados os custos de cada módulo testado e dos acessórios utilizados em conjunto.

4.1.2 Desenvolvimento e Implantação

Todo código produzido em uma linguagem de programação é compilado por uma ferramenta e, então, carrega-se os arquivos binários para o ESP8266 através da serial, para que a execução do código seja iniciada. Na Figura 3, é apresentado um modelo de desenvolvimento e implantação desde o código até chegar no módulo ESP e, na Tabela 3,

Tabela 1 – Descrição e custos de módulos ESP8266

Módulo	Pinos de GPIO e conectores	Memória	Custo
ESP-01	8 pinos macho, incompatível com <i>breadboard</i> (GND, 3v3, TX, RX, CH _P D, RST, GPIO0, GPIO2)	1 MB	R\$ 16,80
ESP-12f	22 pontos para montagem em superfície, nenhum pino	4 MB	R\$ 14,90
D1 mini (ESP-12f)	16 + microUSB	4 MB	R\$ 12,56 ¹
LoLin (ESP-12f)	30 + microUSB	4 MB	R\$ 35,87

Fonte: Produzido pelo autor.

Nota 1: D1 mini (ESP-12f) foi adquirido do mercado chinês.

Tabela 2 – Descrição e custos de acessórios para ESP8266

Acessórios	Descrição	Custo
Esp8266 Placa Para Soldar Esp-07, Esp-08, Esp-12, Esp-12e	Placa com 16 pinos conectados aos pontos de superfície do ESP-12f	R\$ 3,45
Conversor Usb Serial Ch340 Rs232 - 3,3v 5v ¹	Fornece uma conexão serial-USB entre o ESP8266 e o computador de desenvolvimento	R\$ 6,87
Adaptador Usb Serial Ttl Conversor Cp2102 ²	Fornece uma conexão serial-USB entre o ESP8266 e o computador de desenvolvimento	R\$ 20,00
Ams1117 3,3v (3.3v) - Lm1117	Regula a tensão de uma USB ou pilhas para 3.3V 1A usado nos módulos	R\$ 1,50
Fonte Usb 5v 2a Celular Gps Android Ipod	Fonte de alimentação com padrão USB de 5V utilizada com D1 mini	R\$ 9,90

Fonte: Produzido pelo autor.

Nota 1: Compatível apenas com *Windows 7*.

Nota 2: Compatível com *Windows 10* e com o computador de desenvolvimento.

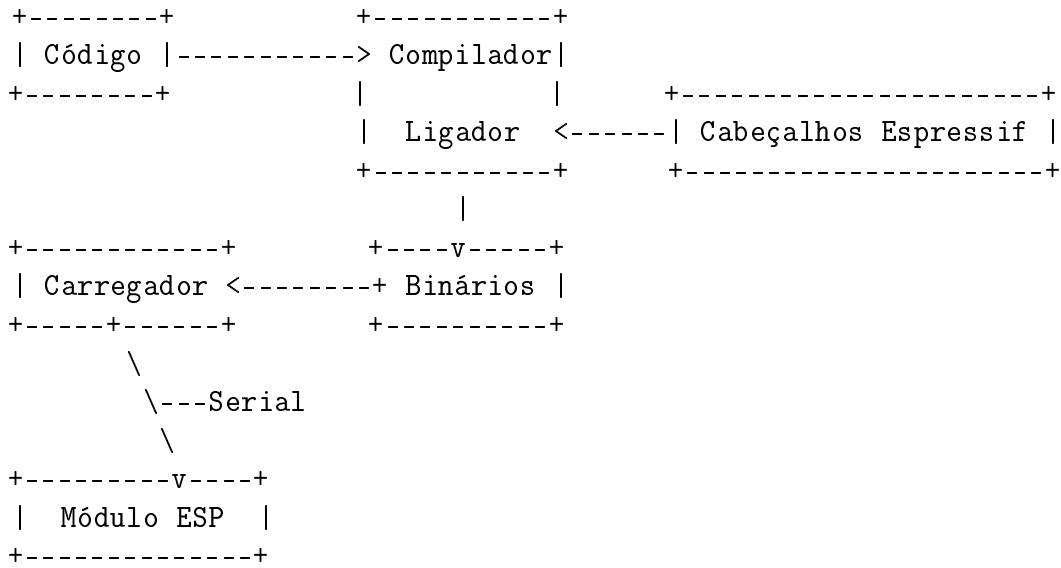
são apresentadas as ferramentas utilizadas como compiladores e carregadores.

Todo código produzido é carregado para o módulo ESP através de seu barramento serial. Alguns modelos, como o *LoLin* e *D1 mini*, já apresentam conversor serial para *micro-USB*. Para os que não possuem tal interface é necessário utilizar um conversor serial-USB externo, a Figura 4 demonstra esse método.

As *GPIOs* do *ESP-12f* são acessadas somente através de placas de circuito impresso, então uma foi adquirida para a programação do mesmo.

Dos conversores serial-USB adquiridos, o modelo *CH340G* não funcionou por não ter driver compatível com o *Windows 10*, em contraste com o modelo *CP2102* que funcionou no mesmo sistema operacional.

Figura 3 – Sequência de ferramentas para implantação



Fonte: Elaborada pelo autor

Tabela 3 – Ferramentas para desenvolvimento com ESP8266

Ferramenta	Editor	Compilador e Ligador	Carregador
Arduino IDE	Sim	arduino C	Sim, mas não carrega binários pré compilados
ESPlorer	Sim	NodeMCU Lua, MicroPython, AT e RN243	Não, conta com firmware específico
esptool.py	Não	Não	Somente binários pré compilados
ESP8266 Flash Downloader	Não	Não	Somente binários pré compilados
NodeMCU Firmware Programmer	Não	Não	Somente binários pré compilados

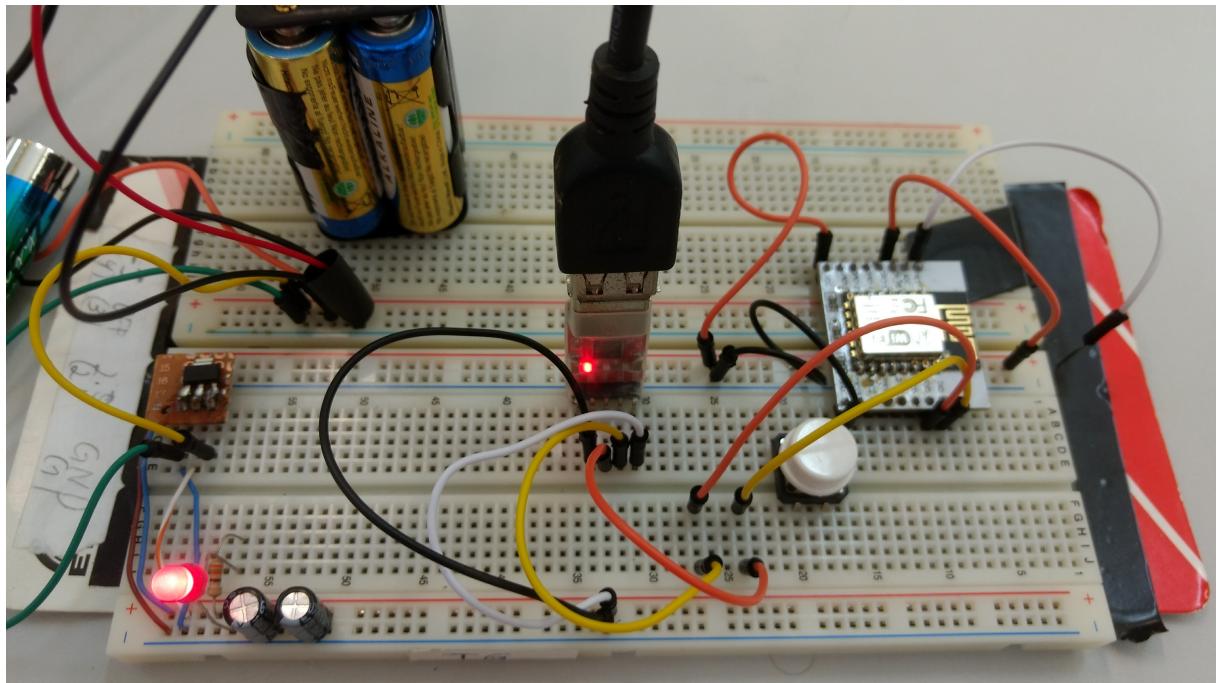
Fonte: Produzido pelo autor.

4.1.3 Testes e resultados

O primeiro objetivo durante a programação dos módulos ESP8266 foi cumprir a premissa estabelecida no início deste capítulo de acessar o Modo Promíscuo da interface Wi-Fi. Neste caso, procurou-se pelo ponto da API de *hardware* do ESP8266 onde os pacotes destinados a outros dispositivos são descartados, desativar este filtro, capturar e avaliar o pacote para localizar o seu emissor.

A princípio, com o *firmware AT*, que é o padrão do módulo ESP-01, e com o emulador de serial da *Arduino IDE* ou a aplicação *Cool Term* é possível configurar e utilizar o módulo por completo apenas com instuções AT enviadas através da conexão serial. A primeira

Figura 4 – ESP-12f com regulador tensão e serial



Fonte: Elaborada pelo autor

investigação sobre a API do protocolo AT indicou Room-15 (2015) como uma fonte sucinta da documentação oficial fornecida por Espressif Systems (2014) do *firmware AT* e não revelou nenhuma capacidade de ativar o Modo Promíscuo.

Também utilizou-se a linguagem C que foi compilada na *Arduino IDE* e enviada ao ESP8266 com a extenção *esp8266 by ESP8266 Community* que inclui os cabeçalhos de funções para que o compilador padrão da *Arduino IDE* gere código executável pelo ESP8266. Mesmo nesta API, nenhuma capacidade de ativar o Modo Promíscuo foi encontrada.

Uma nova tentativa para a programação dos módulos escolhidos foi feita através de *toolchains* (conjunto de ferramentas para desenvolvimento de software) da empresa *Espressif* e de um usuário do *Github*, muito utilizado para projetos de ESPs, Sokolovsky (2017). Ambas as *toolchains* são *SDKs* de código aberto. Os *scripts* foram feitos na linguagem C, compilados nessas *SDKs* e transferidos para os módulos ESP. Neste caso, a configuração delas mostrou-se um desafio, pois requisitavam uma versão específica do *Ubuntu Linux* que a máquina utilizada para o desenvolvimento não suporta. Também foi testada a utilização de máquinas virtuais mas, novamente, a máquina do desenvolvedor não possui virtualização, impossibilitando esta opção.

Em conclusão, apesar do baixo custo e da documentação da comunidade aberta, o ESP8266 não foi adotado como sensor, pois não foi possível colocá-lo em modo prosmícuo, essencial para detectar pacotes entre dispositivo e os pontos de acesso, inviabilizando

Figura 5 – Código em C compilado e implantado em um ESP8266

```

CheckFlashConfig running
Flash real id: 001440E0
Flash real size: 1048576

Flash ide size: 1048576
Flash ide speed: 40000000
Flash ide mode: DIO
Flash Chip configuration ok.

CheckFlashConfig running
Flash real id: 001440E0
Flash real size: 1048576

Flash ide size: 1048576
Flash ide speed: 40000000
Flash ide mode: DIO
Flash Chip configuration ok.

CheckFlashConfig running
Flash real id: 001440E0
Flash real size: 1048576

Flash ide size: 1048576
Flash ide speed: 40000000
Flash ide mode: DIO
Flash Chip configuration ok.

Done uploading
Build options changed, rebuilding all
C:\Users\luizg\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266\2.3.0\cores\esp8266\Schedule.cpp:17:13: warning: 'void init_lists()' defined but not used [-Wunused-function]
    static void init_lists()
    ^
Sketch uses 225,905 bytes (4%) of program storage space. Maximum is 499,696 bytes.
Global variables use 32,500 bytes (3%) of dynamic memory, leaving 49,420 bytes for local variables. Maximum is 81,920 bytes.
Uploading 230946 bytes from C:\Users\luizg\AppData\Local\Temp\build8a8d1e059820a47038110d4b6d768.tmp\CheckFlashConfig.ino.bin to flash at 0x00000000
.....[ 35% ].....[ 71% ].....[ 100% ]

```

Esquerda: Comandos AT no emulador de serial da *Arduino IDE*.

Direita: Editor da *Arduino IDE* com código C.

Abaixo em preto: Processo de *upload* do firmware escrito em C.

Fonte: Elaborado pelo autor.

completamente o uso desta plataforma mesmo sendo a mais adequada e promissora no ponto de vista da construção de um produto final por seu extremo baixo custo.

4.2 Raspberry Pi

Após constatado que o ESP8266 não oferece modo promíscuo, foi testado e desenvolvido software para transformar o Raspberry Pi em uma plataforma para hospedar o sensor. Sua principal diferença é o sistema operacional linux (inexistente no ESP8266) que favorece esta plataforma, porém seu alto custo a desfavorece. Em média, no exterior, o Raspberry Pi é vendido por USD \$ 35,00 (RASPBERRY PI FOUNDATION, 2016) e, no Brasil, entre R\$ 270 em Março de 2016 e R\$ 190 em Janeiro de 2017 (mercadolivre.com.br, 2017b).

As vantagens de ter um computador moderno completo sobreponem seu custo em muitas vezes, dentre as quais destaca-se o poder computacional e a interface "amigável" com usuário devido ao sistema operacional oferecendo maior nível de abstração (bastando apenas alguns comandos para acessá-los e realizar tarefas complexas). Além deste recurso a nível de sistema, a comunidade e o número de projetos "faça você mesmo" é muito maior que a do ESP8266, devido a sua simplicidade em conectar-se a um monitor e construir protótipos e aplicações.

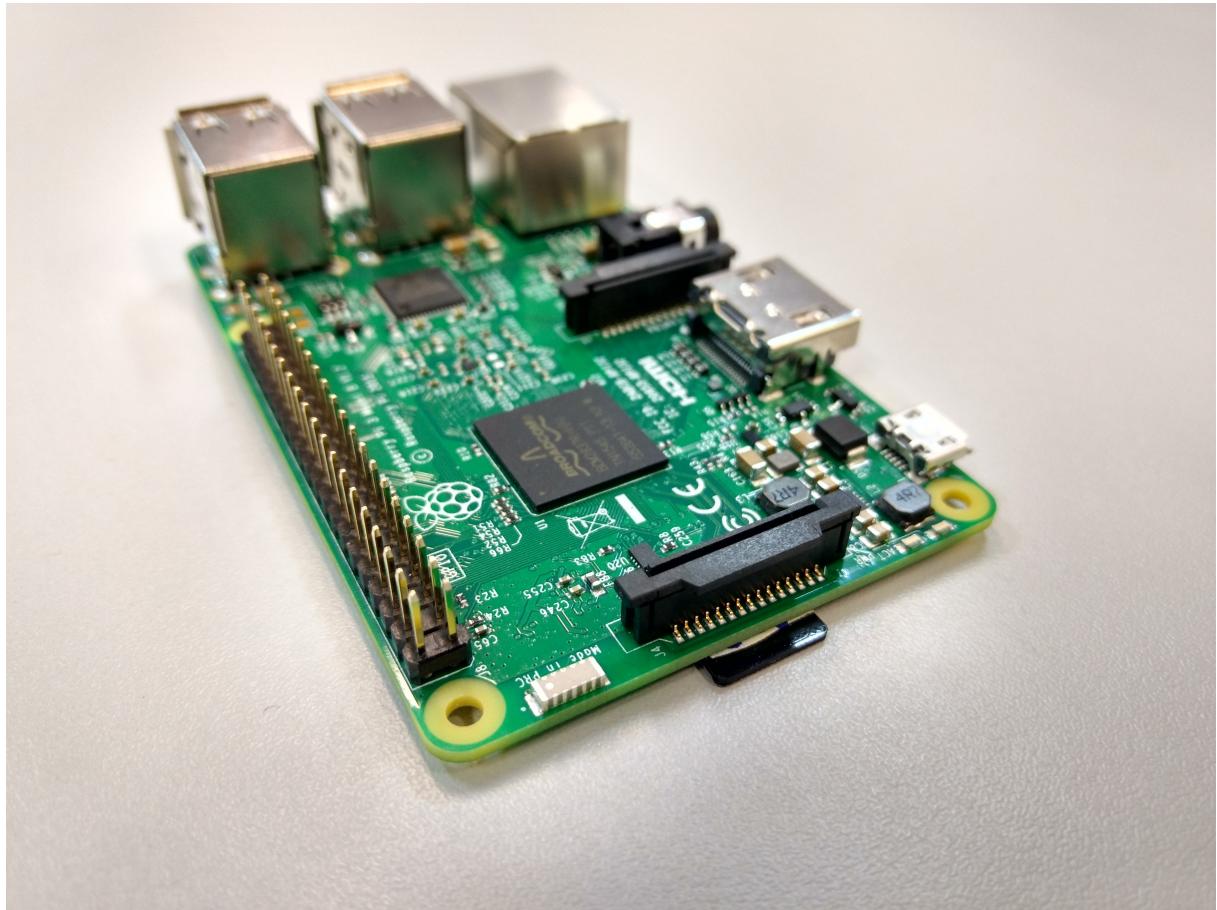
O RPi3 (*Raspberry PI 3 Model B*) é um computador *single-board* (única placa) que tem o tamanho próximo ao de um cartão de crédito. Foi desenvolvido pela *Raspberry Pi Foundation* para promover o ensino da computação nas escolas. Este computador possui:

- a) 1 GB RAM;
- b) Processador Gráfico *VideoCore IV 3D*;
- c) ARM CPU de 1.2 GHz quad-core 64-bit;
- d) 4 portas USB;
- e) 40 pinos GPIOs;
- f) Porta HDMI;
- g) Porta *Megabit Ethernet*;
- h) Saída de áudio e vídeo 3.5 mm;
- i) Interface para câmera (CSI) e monitor (DSI);
- j) Leitor para cartão *micro SD*;
- k) *Wi-Fi LAN* embutida 802.11n;
- l) *Bluetooth 4.1* e *Bluetooth Low Energy* (BLE).

4.2.1 Disponibilidade no mercado

Para abordar a disponibilidade no mercado deve-se também contar os periféricos que são necessários para desenvolver na plataforma RPi3 da mesma maneira que foi feito com o ESP8266.

Figura 6 – Raspberry Pi 3



Fonte: Elaborada pelo autor

O RPI3 é ligado por uma fonte de 2A, 5V e 10W através de uma entrada micro USB. Para ligá-lo, foi adquirido uma fonte USB tipo A para iPad, pois além de poder desconectar o cabo da fonte, facilitando a manutenção, fornece a quantidade exata de amperagem que o computador precisa. A primeira aquisição foi de um carregador de *smartphone* que não forneceu os amperes necessários.

Em comparação com a anterior, esta plataforma tem uma exigência energética maior, muito disto é devido a *Wi-Fi* integrado que é um destaque. A antena de cerâmica do adaptador integrado pode ser vista no primeiro plano da Figura 6. Contudo, o adaptador não possui modo promíscuo e fez-se necessário o uso de adaptadores *Wi-Fi* USB. As recomendações da comunidade quanto a escolha do adaptador USB (também conhecido como *dongle Wi-Fi*) são o *Edimax EW-7811Un* que não é tão comum no Brasil e o *EDUP EP-N85xx* que tem muitos genéricos no mercado nacional.

Como camada de *software*, o RPI3 comporta diversos sistemas operacionais que são carregados de seu cartão *microSD*. Alguns exemplos de sistemas compatíveis são

Archlinux, OpenELECE, Raspbian, Risc OS, Pidora, Kali Linux, Windows 10 IoT, entre outros. Para este trabalho, foi utilizado o *Raspbian*.

Portanto, para funcionamento e desenvolvimento de aplicações com RPI3 são necessários componentes extra que são demonstrados na Tabela 4.

Tabela 4 – Descrição e custos com Raspberry Pi 3

Produto	Descrição e utilização	Custo
Novo Raspberry Pi 3 (pi3) Quadcore 1.2ghz (10x+rapido) 1gb	Computador hospedeiro do sensor	R\$ 269,99
Fonte Carregador Original Usb Apple Iphone 3 4 4s Ipad 1 2	Fonte com conector USB tipo A que supriu o consumo elétrico do RPI3	R\$ 13,99
Cabo USB com conectores <i>A</i> e <i>Micro-B</i>	Para conectar a fonte ao RPI3	R\$ 2,00 ¹
Cartão Micro Sdhc 16gb Ultra Sd Sandisk Classe 10 30mb/s	Armazena o SO e outros arquivos, a classe indica a velocidade do cartão que implica na velocidade do SO	R\$ 21,99
Mini Adaptador Wireless Wifi Edup Usb 150mbps Raspberry Pi	Adaptador externo Wi-Fi que permite modo promíscuo	R\$ 16,88

Fonte: Produzido pelo autor.

Nota 1: Os cabos USB foram reutilizados de outras aplicações.

4.2.2 Desenvolvimento e implantação

Para desenvolver com o RPI3 é necessário instalar um sistema operacional em seu cartão SD, esse processo é simplificado com o uso do *bootloader noobs* que pode ser encontrado no site oficial do Raspberry para download (raspberrypi.org). Após feito o download, os arquivos são extraídos do arquivo comprimido e colocados na pasta raiz do cartão SD. Os próximos passos são conectar o cartão SD, a fonte, monitor, teclado e mouse no RPi e ligar a fonte na tomada para que imediatamente o computador ligue. Na tela inicial deve-se escolher uma rede com ou sem fio. Após conectado, é possível escolher o sistema operacional que será baixado e instalado no próprio cartão SD.

O sistema operacional escolhido para a construção da plataforma de sensor foi o Raspbian que é a distribuição Linux recomendada para o RPI. Nela, já estão instaladas e configuradas muitas ferramentas utilizadas para o desenvolvimento e a implementação de projetos, como *git*, *SSH* e *nodejs* que foram utilizados para a construção da aplicação como será discutido no próximo capítulo.

Para tornar o sistema completamente funcional para o desenvolvimento, é necessário somente executar os passos de segurança e dar acesso remoto ao sistema. Na interface gráfica de configuração do Raspbian deve ser alterado a senha do usuário ‘pi’ e ativado o

serviço SSH que permite acesso remoto através de um terminal. Os últimos três passos são a atualização da lista de pacotes, a atualização dos pacotes instalados e uma reinicialização do sistema para garantir o bom funcionamento e segurança do mesmo.

Feito isto, qualquer desenvolvimento e implantação pode ser realizado com riscos e falhas minimizados. Este processo é fundamental para a segurança da aplicação, usuários e construtores, pois, como foi revelado após os ataques de 21 de Outubro de 2016, dispositivos IoT atualmente não oferecem estes níveis mínimos de segurança (*software* atualizado e senhas seguras), tornando-se um terreno fértil para a construção de *botnets*, como foi o caso do *malware Mirai* utilizado para infectar milhares de dispositivos e causar o maior ataque *DDoS* até o momento com 1,2 terabits por segundo (WOOLF, 2016) (PERLROTH, 2016).

4.2.3 Testes e resultados

De maneira análoga à feita com o ESP8266, analisou-se a capacidade do Raspberry Pi de operar com sua *Wi-Fi* em modo promíscuo, porém, devido a diferença de camada de software envolvida, diferentes ferramentas foram utilizadas.

Neste caso, utilizou-se as ferramentas *airodump-ng* (aircrack-ng.org) e *TShark* além das ferramentas de *Wi-Fi* padrões do sistema operacional *Raspbian*. Para verificar o modo promíscuo no ambiente *Raspbian*, utiliza-se os comandos *ifconfig*, *iwconfig* e *iw*, que são padrão do sistema operacional *Debian*, como demonstrado a seguir.

Código-fonte 4.1 – Ativação do modo monitor

```
1 pi@sensor-01:~ $ sudo ifconfig wlan0 down
2 pi@sensor-01:~ $ sudo iwconfig wlan0 mode monitor
3 pi@sensor-01:~ $ sudo ifconfig wlan0 up
```

O resultado pode ser observado com o comando a seguir.

Código-fonte 4.2 – iwconfig com modo monitor

```
1 pi@sensor-01:~ $ sudo iwconfig wlan0
2 wlan0 IEEE 802.11bgn Mode:Monitor Frequency:2.412 GHz Tx-Power=20 dBm
3      Retry short limit:7 RTS thr:off Fragment thr:off
4      Power Management:off
```

Quando este processo foi realizado utilizando somente o adaptador de *Wi-Fi* integrado no RPI3, cuja antena de cerâmica está em destaque na Figura 7, o resultado foi negativo, portanto outros adaptadores foram necessários.

No ambiente do laboratório (LTIA- Laboratório de Tecnologia da Informação Aplicada), encontrou-se adaptadores *Wi-Fi USB D-link* (Figura 8), porém executando o mesmo teste neles não foi possível ativar o modo promíscuo. Um terceiro adaptador emprestado foi o modelo *Edup* (Figura 9) da fabricante *Ralink* que teve resultado positivo. Para a construção

dos dois sensores, foi necessária a aquisição, e subsequente teste, de mais um modelo de adaptador que está listado na Tabela 4 que também tem como fabricante a *Ralink*, porém a sua aparência externa difere do anterior e pode ser visualizada na Figura 10, mesmo que a aquisição foi do mesmo anúncio.

Figura 7 – Antena cerâmica de Wi-Fi e Bluetooth do Raspberry Pi 3 Figura 8 – Adaptador Wi-Fi USB D-Link do laboratório

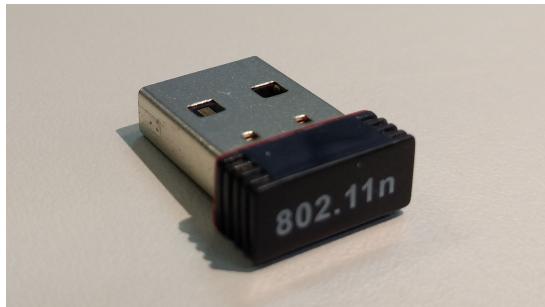


Fonte: Produzido pelo autor



Fonte: Produzido pelo autor

Figura 9 – Adaptador Wi-Fi USB Ralink Epub emprestado



Fonte: Produzido pelo autor

Figura 10 – Adaptador Wi-Fi USB Ralink Epub comprado



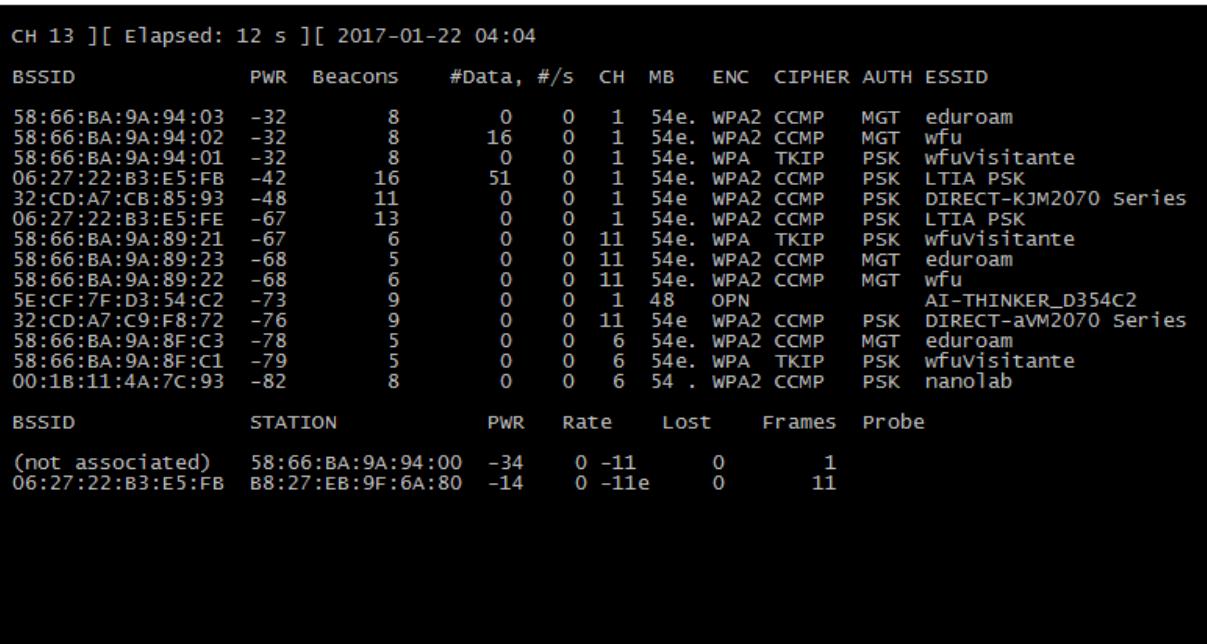
Fonte: Produzido pelo autor

Para capturar e avaliar pacotes uma ferramenta é necessária. Na área de segurança da informação pode-se encontrar o *airodump-ng* que é utilizado para avaliar e explorar vulnerabilidades de segurança em redes *Wi-Fi*. Outra área que forneceu uma ferramenta adequada foi a área de qualidade de serviço em redes de computadores (QoS) onde o software *Wireshark* é bem popular, uma interface alternativa do mesmo feita para uso em terminal é chamada *TShark*.

Para testar a viabilidade do sensor, utilizou-se o *airodump-ng* que é uma ferramenta de terminal interativa, como vista na Figura 11, onde é demonstrado a capacidade de capturar pacotes do tipo *Beacon* (tabela com BSSID, PWR e Beacons na parte superior da Figura 11) que anunciam a presença de AP (*Access Point* - ponto de acesso) e o nome da rede que ele está servindo. Além disso, é possível observar também os pacotes entre os dispositivos (*Stations*) associados a um AP (tabela com BSSID, STATION e PWR na

parte inferior da Figura 11). Em ambos os casos, é mostrado um valor de potência de sinal (PWR) associado a cada transmissor, portanto, demonstrando a viabilidade do sensor com a informação de potência de sinal.

Figura 11 – Interface do airodump-ng



```

pi@ltia-pi-01: ~
CH 13 ][ Elapsed: 12 s ][ 2017-01-22 04:04
          BSSID      PWR  Beacons    #Data, #/s   CH   MB   ENC   CIPHER AUTH ESSID
          58:66:BA:9A:94:03 -32       8        0   0   54e. WPA2 CCMP   MGT eduroam
          58:66:BA:9A:94:02 -32       8       16   0   1   54e. WPA2 CCMP   MGT wfu
          58:66:BA:9A:94:01 -32       8        0   0   1   54e. WPA  TKIP   PSK wfuvisitante
          06:27:22:B3:E5:FB -42      16       51   0   1   54e. WPA2 CCMP   MGT LTIA PSK
          32:CD:A7:CB:85:93 -48       11       0   0   1   54e. WPA2 CCMP   PSK DIRECT-KJM2070 Series
          06:27:22:B3:E5:FE -67      13       0   0   1   54e. WPA2 CCMP   PSK LTIA PSK
          58:66:BA:9A:89:21 -67       6       0   0   11  54e. WPA  TKIP   PSK wfuvisitante
          58:66:BA:9A:89:23 -68       5       0   0   11  54e. WPA2 CCMP   MGT eduroam
          58:66:BA:9A:89:22 -68       6       0   0   11  54e. WPA2 CCMP   MGT wfu
          5E:CF:7F:D3:54:C2 -73       9       0   0   1   48   OPN
          32:CD:A7:c9:F8:72 -76       9       0   0   11  54e. WPA2 CCMP   PSK DIRECT-avM2070 Series
          58:66:BA:9A:8F:C3 -78       5       0   0   6   54e. WPA2 CCMP   MGT eduroam
          58:66:BA:9A:8F:C1 -79       5       0   0   6   54e. WPA  TKIP   PSK wfuvisitante
          00:1B:11:4A:7C:93 -82       8       0   0   6   54e. WPA2 CCMP   PSK nanolab

          BSSID      STATION      PWR  Rate   Lost   Frames   Probe
          (not associated) 58:66:BA:9A:94:00 -34   0 -11    0       1
          06:27:22:B3:E5:FB B8:27:EB:9F:6A:80 -14   0 -11e   0       11

```

Fonte: Elaborada pelo autor

Após a avaliação de viabilidade, fez-se necessário o uso de uma aplicação mais flexível do que o *airodump-ng* onde fosse possível escolher campos e gerar relatórios mais flexíveis. Para essa tarefa o software *TShark* mostrou-se ideal.

Nele, pode-se escolher, através de argumentos na execução por terminal, a interface com a opção “-i wlan0”, o modo monitor com opção “-l”, a opção “-T fields” que altera o funcionamento normal dele para que com as opções “-e field.field_child” seja permitido escolher os campos mostrados e juntamente com as opções “-E separator=, -E quote=d” o formato do relatório gerado torna-se CSV (*comma separated values* - valores separados por vírgula).

Desta maneira, todos os pacotes capturados são processados pelo *TShark* e escritos na saída padrão do terminal (*stdout*), dando a possibilidade de usar a ferramenta de criação de arquivo, acrescentar em arquivo e redirecionar para outro processo do terminal Linux (respectivamente ‘>’, ‘>>’ e ‘|’). Esta capacidade, ausente no *airodump-ng*, é essencial para este trabalho.

Para este trabalho, o comando mais utilizado foi o que mostra os endereços MAC de origem e transmissão (wlan.sa, wlan.ta), os mesmos endereços, porém com nome de

fabricante como prefixo (wlan.sa_resolved, wlan.ta_resolved), a potência de sinal (radiotap.dbm_antsignal) e o nome da rede anunciada se o pacote for um Beacon, da mesma maneira que o *airodump-ng* mostra.

Código-fonte 4.3 – TShark e opções

```
1 pi@sensor-01:~ $ tshark -l -i wlan0 -T fields -E header=y -E quote=d \
2 -e wlan.sa -e wlan.sa_resolved -e wlan.ta -e wlan.ta_resolved \
3 -e radiotap.dbm_antsignal -e wlan_mgt.ssid
```

Por último, a configuração padrão do *TShark* não permite execução em modo supervisor (*root*), portanto, por segurança, somente usuários do grupo *WireShark* podem executar o programa. Para isso, é necessário adicionar o usuário "pi" ao grupo *wireshark* com o seguinte comando.

Código-fonte 4.4 – Adição do usuário pi ao grupo *wireshark*

```
1 pi@sensor-01:~ $ sudo usermod -a -G wireshark pi
```

Este modo de operação permite executar a aplicação final sem necessidade de elevação de privilégios (*root*), tornando-a mais segura, pois mesmo que aplicação seja subvertida o sistema operacional não poderá ser comprometido.

4.3 Escolha e conclusão

Em comparação com o ESP8266, o RPI3 compensou seu custo elevado devido a facilidade de programação, acesso aos seus recursos e acesso a recursos externos uma vez que foi possível chegar ao modo promíscuo facilmente através de recursos nativos do sistema operacional. Para uma comparação das plataformas veja a Tabela 5.

Tabela 5 – Comparação das plataformas ESP8266 e RPI3

Aspecto	Raspberry Pi 3 model B	ESP-12f
GPIO	27 GPIOs (0 a 26) digitais	17 GPIOs digitais e analógicos
Número de pinos	40 pinos	22 pinos
Processamento	ARMv8 64-bit quad-core 1.2 GHz e VideoCore IV 3D GPU	Tensilica L106 32-bit (MCU) com 80 ou 160 MHz e instruções 16-bit RSIC
Memória RAM	1 GB	RAM < 50 kB
Memória longo termo	cartão SD (usualmente 8 ou 16 GB)	SPI flash de 4 MB
Tamanho físico	85x56mm	24x13mm
Rede embutida	Megabit Ethernet, 802.11 b/g/n	802.11 b/g/n
Expansão	USB, DSI, CSI e GPIO	Somente GPIO
Sistema operacional	Qualquer linux/windows/risc compilado em ARMv8	Não possui
Custo	de R\$ 190,00 a R\$ 270,00	de R\$ 12,56 a R\$ 35,87

Fonte: Produzido pelo autor.

O RPI3 foi adotado como plataforma para o sensor de detecção de dispositivos, pois o modo promíscuo (*monitor mode*) conseguiu ser acessado através de adaptador USB Wi-Fi. Apesar do esforço para ativação do modo promíscuo na plataforma ESP8266, o uso desta reduziria significativamente o custo de cada sensor. Para comparação do custo da aplicação aqui proposta, veja a Tabela 6.

É importante notar a proporção de custo entre as duas plataformas onde o RPI3 custa aproximadamente 15 vezes mais por sensor do que a plataforma ESP8266. Isto justifica o esforço realizado durante o desenvolvimento deste trabalho para explorar e, com esperança, ativar o modo promíscuo no pequeno dispositivo que infelizmente não rendeu frutos.

Para a construção do *Gateway IoT*, as exigências de *hardware* são capacidades mínimas de processamento, armazenamento e comunicação, e a exigência de *software* é um sistema operacional que suporte um *MQTT Broker*. Portanto, para o *gateway*, a plataforma ESP8266 claramente não é adequada, então um RPI3 representa o custo mínimo.

Tabela 6 – Comparação de custos para o sensor da aplicação proposta em função da plataforma

Sensor					
Plataforma	Raspberry Pi		ESP8266		
Item	Descrição	Custo em R\$	Descrição	Custo em R\$	
Plataforma	RPI3	269,99	D1 mini (ESP-12f)	12,56	
Fonte de alimentação	Fonte Usb iPad	13,99	Fonte Usb Celular com cabo		7,85
	Cabo Usb A-micro	2,00			
Adaptador Wi-Fi	Edup Usb	16,88			
Memória	SD c10 16GB	21,99			
Total por Sensor		324,85			20,41

Fonte: Produzido pelo autor.

Tabela 7 – Custos para o gateway da aplicação proposta

Gateway			
Item	Descrição	Custo em R\$	
Plataforma	RPI3	269,99	
Fonte de alimentação	Fonte Usb iPad	13,99	
	Cabo Usb A-micro	2,00	
Memória	SD c10 16GB	21,99	
Total por Gateway		307,97	

Fonte: Produzido pelo autor.

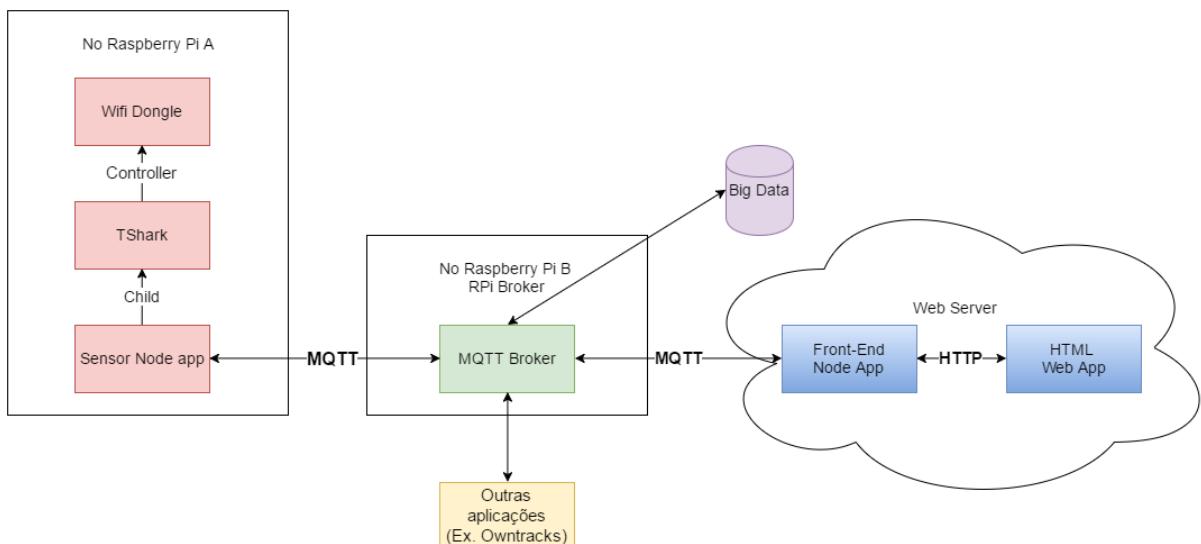
5 CONSTRUÇÃO

Para a construção do *software* aplicativo, foi utilizado uma arquitetura em três camadas: sensor, distribuidor de acesso (*IoT gateway*) e apresentação (*Web*). Nesta divisão, os sensores capturam as informações dos dispositivos e repassam para a camada seguinte, no *gateway* todas as partes se encontram para fornecer e solicitar informações e, por último, a camada de apresentação coleta o que é enviado dos sensores e gera uma página *Web* para visualização dos dados capturados.

Esta divisão está de acordo com o padrão encontrado em outras aplicações *IoT* onde a última camada usualmente varia entre apresentação e mineração de dados (*Data Mining*).

A camada de sensor utilizou as tecnologias *Node.js*, *TShark* parte do *Wireshark* e *MQTT.js*. A camada *gateway* foi composta basicamente pelo *MQTT Broker Mosquitto*. Por fim a camada de apresentação utilizou as tecnologias *Node.js*, *MQTT.js*, *html*, *css*, *javascript*, *Bootstrap* e *Google Maps API*.

Figura 12 – Arquitetura da aplicação



Fonte: Elaborada pelo autor

5.1 Sensor

A aplicação sensor tem como requisitos funcionais capturar, avaliar e classificar pacotes de *Wi-Fi*, inferir estatísticas de dispositivos e fornecer estas informações para os interessados através do *gateway*.

Para fazer a captura dos pacotes na aplicação final, diferente do que foi demonstrado na subseção 4.2.3, em especial o *airodump-ng* e sua interface demonstrada na Figura 11, foi utilizado o programa *TShark* cujo modo de operação serve melhor para a construção dos *Streams* que serão abordados em breve.

TShark é uma versão orientada ao terminal do Wireshark projetada para capturar e exibir pacotes quando uma interface de usuário interativa não é necessária ou disponível. Ele suporta as mesmas opções como wireshark.

Wireshark.org (2014) Tradução Nossa.

Como foi estabelecido no capítulo anterior, *TShark* utiliza a saída padrão do terminal (*stdout*) como sua saída principal, esta característica foi explorada com a aplicação *nodejs*. Mais especificamente, com o módulo *child_process*, que provê uma API que permite a criação e controle de processos filhos do processo *nodejs*.

Node.js é uma estrutura em tempo de execução construída sobre o motor de execução JavaScript V8 do Chrome. Node.js utiliza um modelo orientado a evento, de entrada e saída não bloqueante que o faz leve e eficiente. O ecossistema de pacotes do Node.js, npm, é o maior ecossistema de bibliotecas de código livre no mundo.

Nodejs.org (2016b) Tradução Nossa.

Como também foi estabelecido anteriormente, o *TShark* é executado com o comando e argumentos como mostrado a seguir, a diferença em relação aos testes e na escolha da plataforma é a forma de execução, na maneira mostrada, o processo é criado utilizando o módulo *child_process* (NODEJS.ORG, 2016a) e os argumentos são passados como um vetor (*Array*).

Código-fonte 5.1 – TShark e opções executado pelo Node.js

```
1 const spawn = require('child_process').spawn;
2 const tsharkProcessoFilho = spawn(
3   'tshark', [
4     '-l',
5     '-i', childInterface,
6     '-T', 'fields',
7     '-E', 'separator=,',
8     '-E', 'quote=d',
9     '-e', 'wlan.sa',
10    '-e', 'wlan.sa_resolved',
```

```

11     '-e', 'wlan.ta',
12     '-e', 'wlan.ta_resolved',
13     '-e', 'radiotap.dbm_antsignal',
14     '-e', 'wlan_mgt.ssid',
15     '-Y', 'wlan.sa'
16 ]);

```

Para utilizar o resultado gerado pelo *TShark*, utilizou-se outro método do módulo *child_process* juntamente com a estrutura de *Stream* (NODEJS.ORG, 2016c) que provê o método *pipe(destination[, options])* que permite, de maneira análoga ao operador '|' no terminal também chamado de *pipe*, redirecionar a saída de um processo ou *stream* de leitura para outro processo ou *stream* de escrita.

Com isso, ainda falta um *stream* de escrita que receba a saída do *TShark* que foi definida no formato CSV. Para isso, uma biblioteca extra, *fast-csv*, deve ser instalada. Com ela, pode-se criar o *stream* necessário e configurá-lo para interpretar os resultados (C2FO.COM, 2016).

Código-fonte 5.2 – Uso do fast-csv

```

1 const csv = require("fast-csv");
2 let csvStream = csv()
3   .on("data", function(data){
4     let packet = new Packet(
5       data[0], // sender address
6       data[1], // sender address resolved
7       data[2], // transmitter address
8       data[3], // transmitter address resolved
9       data[4], // potencia de sinal (rss)
10      data[5] // nome da rede no pacote Beacon
11    );
12    processarPacote(packet);
13  })
14  .on("end", function(){
15    console.log("done with tshark");
16  });
17 tsharkProcessoFilho.stdout.setEncoding('utf8');
18 tsharkProcessoFilho.stdout.pipe(csvStream);

```

Na linha 18, pode-se observar a operação de redirecionamento (*pipe*) de *stream* da saída padrão do processo filho (*stdout*) para o *stream* de escrita descrito e configurado com o *fast-csv*. A parte faltante é o processamento dos pacotes e envio das inferências do sensor para o *gateway*.

Para as inferências e estatísticas, foi criado um objeto para armazenamento indexado pelo endereço MAC e uma classe onde as informações sobre cada dispositivo são agregadas. Desta forma, cada novo pacote pode ser acrescentado ao histórico de cada

dispositivo.

Código-fonte 5.3 – Adição do pacote ao histórico do dispositivo

```

1 let devices = {}; // lista indexada de dispositivos
2 class Device {
3   [...]
4   appendPacket(packet){
5     let curTime = ( new Date() ).toISOString();
6     this.rssHistory.push(packet.radiotap.dbm_antsignal);
7     this.ssidHistory[packet.wlan_mgt.ssid] = curTime;
8     this.taHistory[packet.wlan.ta] = {
9       ta      : packet.wlan.ta,
10      ta_resolved : packet.wlan.ta_resolved,
11    };
12  }
13 }
14 function processarPacote(packet) {
15   let sa = packet.wlan.sa;
16   if (!devices[sa]){
17     devices[sa] = new Device(sa, packet.wlan.sa_resolved);
18   }
19   devices[sa].appendPacket(packet);
20 }
```

A partir desta lista, é possível calcular a média e o desvio padrão de potência de sinal para cada dispositivo descoberto.

Código-fonte 5.4 – Extração das estatísticas do dispositivo

```

1 class Device {
2   [...]
3   get rssStatistics(){
4     let sum = 0, avg = 0, variance = 0, stdDeviation = 0;
5     if (this.rssHistory.length > 0){
6       for (let rss of this.rssHistory){
7         sum += rss;
8       }
9       avg = sum / this.rssHistory.length;
10      sum = 0;
11      for (let rss of this.rssHistory){
12        sum += Math.pow(( rss - avg ), 2);
13      }
14      variance = sum / (this.rssHistory.length - 1);
15      stdDeviation = Math.sqrt(variance);
16    }
17    return {
18      size      : this.rssHistory.length,
19      avg       : avg,
20      stdDeviation : stdDeviation,
```

```

21      };
22    }
23  }

```

Por fim, é necessário comunicar aos interessados nessas inferências, o que é feito através do módulo [github.com/mqttjs](https://github.com/mqttjs/mqttjs) (2016).

Código-fonte 5.5 – Cliente MQTT.js

```

1  const mqtt = require('mqtt');
2  const mqttBrok = 'mqtt://${config.mqttHost}:${config.mqttPort}';
3  let clientMqtt = mqtt.connect(mqttBrok, {
4    username : config.mqttUser,
5    password : config.mqttPwd,
6  })
7  clientMqtt.on('connect', function () {
8    clientMqtt.subscribe('devices');
9    startTshark();
10 });
11 clientMqtt.on('message', function (topic, message) {
12   if (topic.toString() == 'devices') {
13     switch (message.toString()) {
14       case 'list':
15         clientMqtt.publish(
16           'devices/report',
17           JSON.stringify( tshark.getDevices() )
18         );
19         break;
20       case 'report':
21         clientMqtt.publish(
22           'devices/report',
23           JSON.stringify( tshark.getReport() )
24         );
25         break;
26       default:
27         let mac = message.toString();
28         clientMqtt.publish(
29           'devices/report',
30           JSON.stringify( tshark.getDeviceReport(mac) )
31         );
32         break;
33     }
34   }
35 });

```

No caso desta listagem de código, quando a aplicação é iniciada ela conecta-se ao *MQTT Broker*. Quando a conexão é estabelecida, ela solicita ao *MQTT Broker* a inscrição para receber as publicações no tópico 'devices' e o processo filho *TShark* é iniciado. O

processo filho é executado, os resultados são capturados e guardados.

Quando uma mensagem no tópico 'devices' é recebida, três reações podem acontecer: a lista de dispositivos deve ser fornecida, o relatório do sensor deve ser fornecido ou um relatório sobre um dispositivo específico deve ser fornecido. Para todos os casos, uma resposta adequada é imediatamente processada e enviada para o tópico 'devices/report'.

Em conclusão, a aplicação sensor instancia o processo de aquisição de dados *TShark* assincronamente e captura, classifica e armazena todos os pacotes que ficam acessíveis através de requisições ao tópico *MQTT* 'devices'. Isso cobre os requisitos funcionais desta aplicação.

Os requisitos não-funcionais desta aplicação estão ligados com o ambiente de implantação que é um RPI3 remoto, sem outro dispositivo de entrada e saída exceto a comunicação *MQTT*, portanto, sem nenhum aspecto permitindo monitoração de um humano que garanta que o aplicativo continue funcionando.

Este ambiente se assemelha muito a aplicações em núvem, onde não há acesso físico ao computador onde a aplicação é executada. Neste ambiente, procura-se garantir que a aplicação seja executada constantemente e atualizada automaticamente sempre que uma nova versão é construída e testada. Estas garantias podem ser alcançadas com ferramentas de integração contínua (*Continuous Integration - CI*). No caso do Github, a escolha de hospedagem de código deste projeto, diversas opções deste tipo de ferramenta podem ser encontradas em (GITHUB.COM, 2016).

No entanto, para esta aplicação, construiu-se uma solução extremamente simplificada destas ferramentas utilizando a arquitetura existente do programa *git* para garantir que novas versões fossem instaladas assim que possível. A implementação consiste da adição, nas primeiras instruções do aplicativo, uma rotina que executa sincronamente um programa externo através do terminal (diferente da execução do *TShark* que é assíncrona). Este programa é o *git* com a opção *pull* que, quando executado em um diretório que é repositório, solicita ao servidor configurado, como *origin*, a atualização do código do *branch* atual. Se a mensagem encontrada na saída padrão (*stdout*) resultante deste comando for diferente de "Already up-to-date.", o programa finaliza imediatamente, pois uma nova versão foi baixada e uma reinicialização da aplicação é necessária. Em resumo, uma vez instalada a aplicação utilizando a ferramenta *git*, ela será atualizada com o mesmo processo com que foi instalada.

O outro requisito é que a aplicação seja executada constantemente. Para isso, pode-se utilizar uma aplicação também escrita em *Node.js* e disponível no gerenciador de pacotes *npm*: *forever-service*. Ela registra um novo serviço no sistema operacional linux para que a infra-estrutura de gerenciamento de serviços existente seja aproveitada (por exemplo: início automático após o início do sistema). Além disso, a sua dependência *forever* garante que a

aplicação seja reinicializada sempre que finaliza, executando a aplicação constantemente (Zappy Inc, 2016).

5.2 Gateway

Para a construção do *Gateway* foi feita a instalação e a configuração do *MQTT Broker Mosquitto* em um dos RPI3.

Eclipse Mosquitto™ é um distribuidor de mensagens de código aberto (EPL/EDL licenciado) que implementa o protocolo MQTT versões 3.1 e 3.1.1. O MQTT fornece um método leve de transmitir mensagens usando um modelo de publicação/inscrição. Isso o torna adequado para mensagens "Internet das Coisas", como com sensores de baixa potência ou dispositivos móveis, como telefones, computadores embutidos ou microcontroladores como o Arduino.

Mosquitto.org (2016) Tradução Nossa.

A instalação é realizada com o gerenciador de pacotes *apt-get* padrão do *raspbian* como mostrado na primeira linha da listagem abaixo.

Código-fonte 5.6 – Instalação e configuração do Mosquitto

```

1 pi@broker:~ $ sudo apt-get install mosquitto
2 pi@broker:~ $ sudo sh -c 'echo "password_file /etc/mosquitto/passwd"
3      >> /etc/mosquitto/mosquitto.conf'
4 pi@broker:~ $ sudo mosquitto_passwd -c /etc/mosquitto/passwd user
5 Password:
6 Reenter password:
7 pi@broker:~ $
```

Nas linhas 2 e 3, adiciona-se ao arquivo padrão de configuração padrão do *Mosquitto* a linha 'password_file /etc/mosquitto/passwd' que indica o arquivo de senhas a ser utilizado para autenticação. Na linha 4, é adicionado o usuário 'user' ao arquivo, a senha é solicitada nas linhas 5 e 6.

Feito isso, o acesso ao *Broker* está limitado aos usuários e senhas configurados neste processo. As demais configurações permaneceram sem alterações.

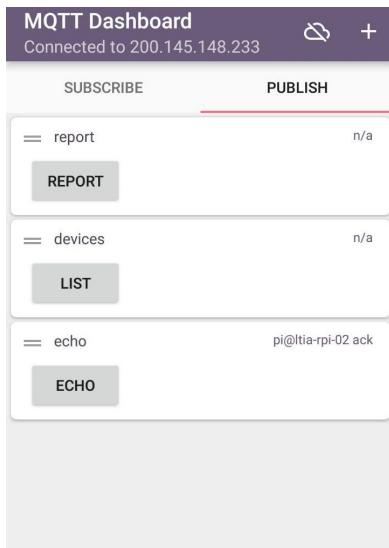
Na configuração do sensor, deve ser adicionado o endereço (nome ou IP), a porta 1883 e o par usuário e senha para que o sensor acesse o *Gateway* com sucesso.

Para verificar o funcionamento do conjunto sensor e distribuidor (*Gateway*), utiliza-se um cliente *MQTT*, como o aplicativo para a plataforma *Android MQTT Dashboard* (Nghia TH, 2016) ou os aplicativos para as plataformas *Java mqtt-spy* (eclipse, 2016) e *Windows MQTT.fx* (DETERS, 2017).

Nas figuras 13, 14 e 15, o aplicativo *MQTT Dashboard* é utilizado para verificar quais sensores estão online com a mensagem 'echo' publicada no tópico 'ADMIN'. Na Figura 13, os botões enviam mensagens com simplicidade. Na Figura 14, é visível a lista de tópicos e a última mensagem recebida em cada um deles. Na Figura 15, fica evidente a resposta de confirmação (ack) de cada um dos sensores.

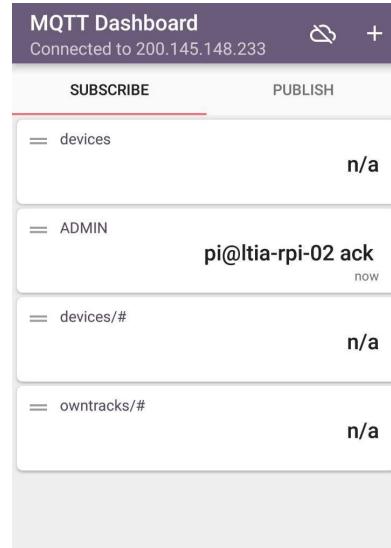
A aplicação *MQTT.fx*, em especial a tela mostrada na Figura 16, revela estatísticas do *Broker*, como a versão, tempo *online*, número de clientes, mensagens e utilização de rede. Na aplicação *mqtt-spy*, o comando de listagem (mencionado na página 46) é demonstrado.

Figura 13 – MQTT Dashboard: Envio de mensagens



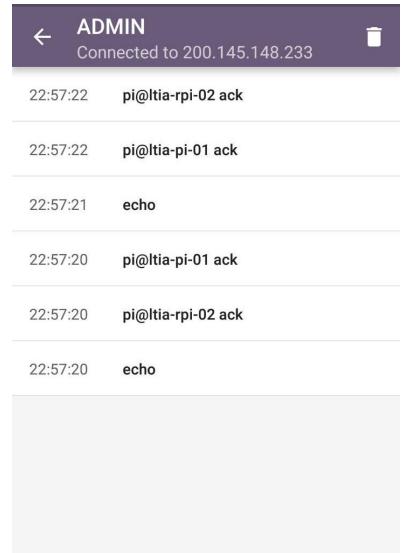
Fonte: Produzido pelo autor

Figura 14 – MQTT Dashboard: Lista de inscrições



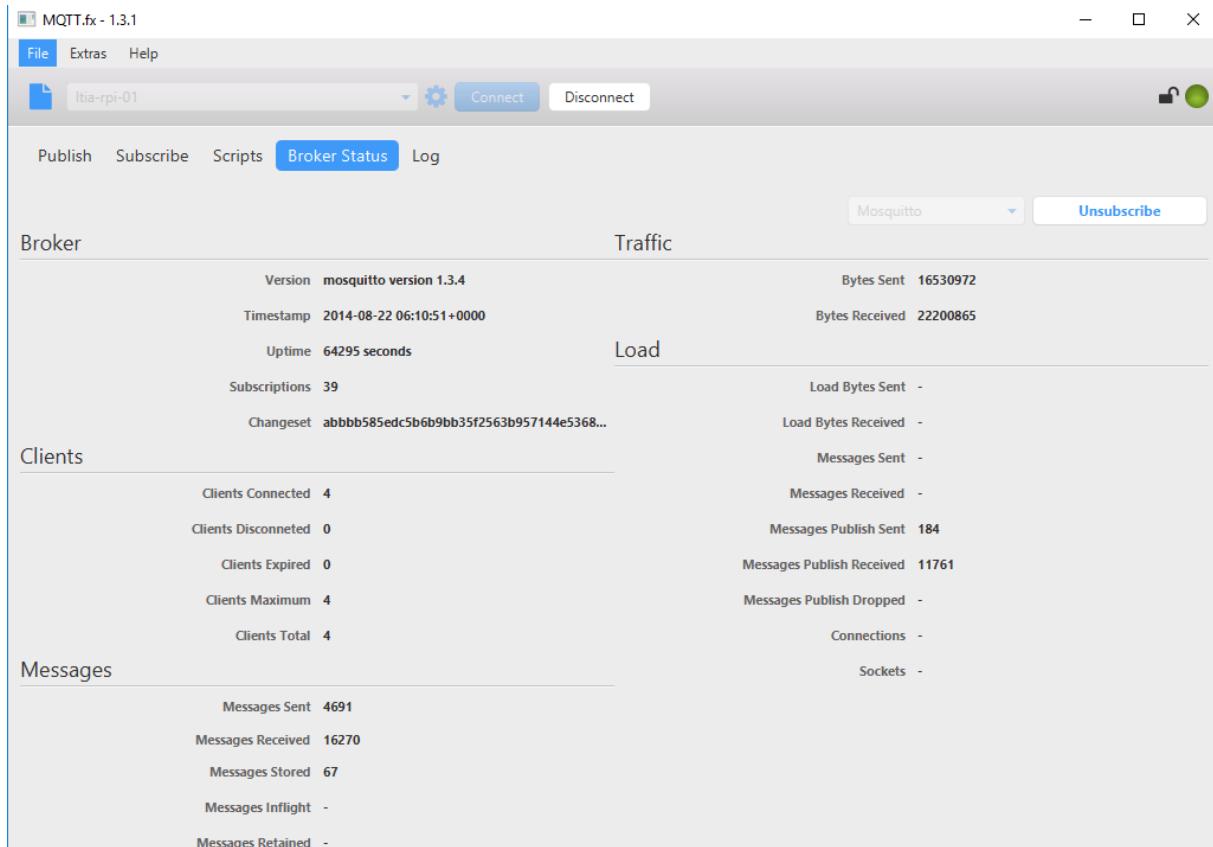
Fonte: Produzido pelo autor

Figura 15 – MQTT Dashboard: Lista de mensagens no tópico



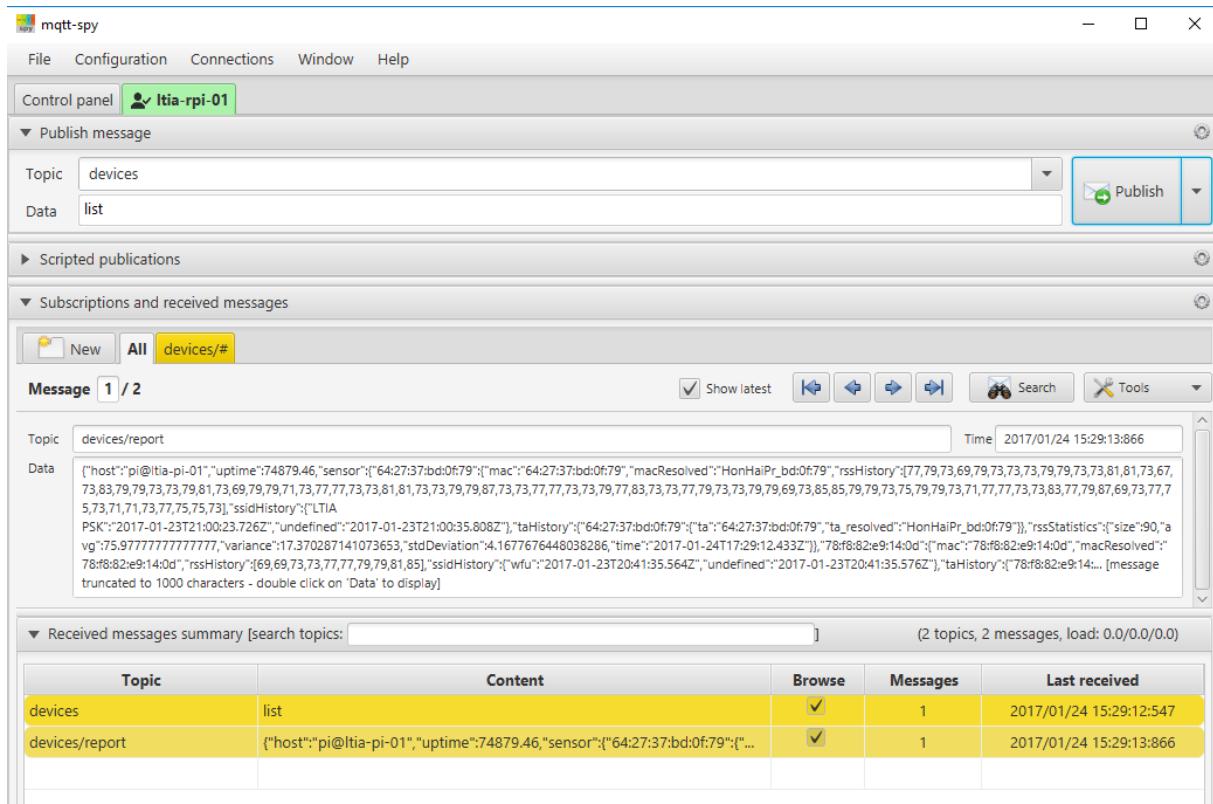
Fonte: Produzido pelo autor

Figura 16 – MQTT.fx: Estatísticas do Broker



Fonte: Produzido pelo autor

Figura 17 – mqtt-spy: Listagem de dispositivos



Fonte: Produzido pelo autor

5.3 Apresentação Web

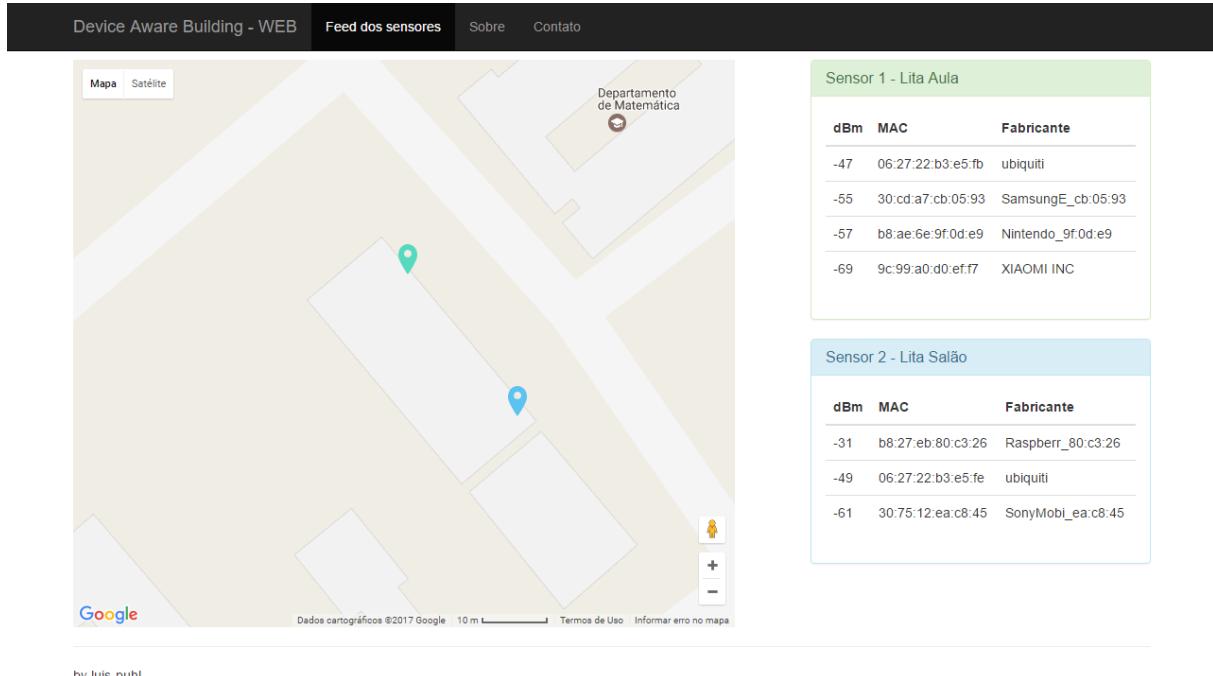
Como já mencionado, para a apresentação das informações de captura de maneira acessível, foi construída uma aplicação *Web* utilizando as tecnologias *Node.js*, *MQTT.js*, *html*, *css*, *javascript*, *Bootstrap* e *Google Maps API*.

Node.js e a biblioteca de cliente *MQTT.js* foram utilizados para, da mesma forma exposta no Código-fonte 5.5, conectar uma aplicação escrita em *javascript* com o *MQTT Broker*. Esta aplicação recupera as informações sobre os dispositivos descobertos e as classifica por proximidade de cada sensor para compor a lista de dispositivos por sensor vista no lado direito da Figura 18.

Já o *html*, *css*, *javascript* e *Bootstrap* foram utilizados para estruturar, estilizar, inflar e animar as informações. Em especial, o *Bootstrap* forneceu a estrutura de cabeçalho, rodapé e colunas, além do esquema de cores.

A *Google Maps API* juntamente com um pouco de *css* e *javascript* fornece o mapa visto no lado direito da Figura 18. Nele estão representadas as localizações geográficas de cada sensor representados pelos marcadores nas cores azul e verde.

Figura 18 – Web APP



Fonte: Elaborada pelo autor

6 RESULTADOS E DISCUSSÃO

Neste capítulo são abordados, analisados e discutidos os resultados encontrados durante a exploração do tema e das plataformas, durante da implementação das aplicações além de verificar a precisão atingida com a aplicação implementada.

6.1 Método de teste

Como discutido no Capítulo 5, a arquitetura da aplicação total mostra que a precisão vista na aplicação *Web* depende dos resultados encontrados pela aplicação sensor e, por sua vez, este depende do par de capacidades combinadas do *hardware* adaptador Wi-Fi e do *software TShark*, portanto, a metodologia de testes empregada neste capítulo é analisar diretamente as capacidades deste último par. Esta decisão também se deve pela facilidade de armazenar e analisar arquivos CSV gerados pelo *TShark*.

A captura foi executada com o comando

Código-fonte 6.1 – TShark e redirecionamento da saída para arquivo assíncrono

```

1 pi@sensor-01:~ $ tshark -l -i wlan0 -T fields -E header=y -E quote=d \
2 -e wlan.sa -e wlan.sa_resolved -e wlan.ta -e wlan.ta_resolved \
3 -e radiotap.dbm_antsignal -e wlan_mgt.ssid \
4 >> 2017-01-17--02-48--rpi-02.csv &
5 pi@sensor-01:~ $ exit

```

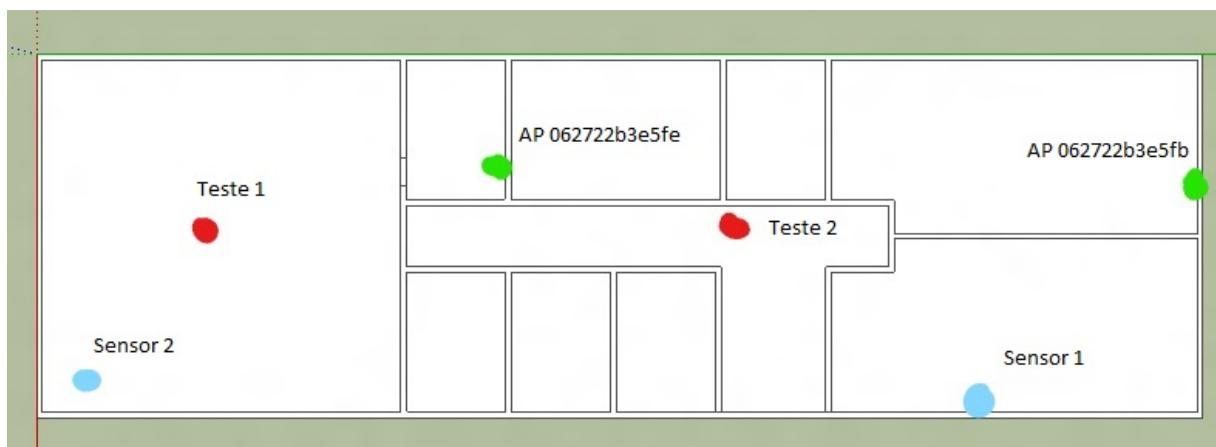
Neste modo de uso os resultados são direcionadas para a saída padrão (stdout) do terminal e podem ser capturados por outro programa no formato de valores separados por vírgula (csv). os campos escolhidos para captura são *wlan.sa*, *wlan.sa_resolved*, *wlan.ta*, *wlan.ta_resolved*, *radiotap.dbm_antsignal* e *wlan_mgt.ssid*.

6.2 Avaliação de ruído e consistência

Para entender o ambiente onde a aplicação foi desenvolvida e testada no âmbito de ruído wi-fi e pontos de referência wi-fi foi executada uma captura de teste durante a noite quando ninguém habitava o prédio protótipo.

Nesta captura dois sensores foram utilizados posicionados a menos de 10 centímetros de distância um do outro sobre uma mesa a um metro do chão a captura ocorreu de 2:50 até aproximadamente 11:25 totalizando aproximadamente 8 horas de captura

Figura 19 – Ambiente de teste



Fonte: Elaborada pelo autor

Em azul: Sensores da aplicação

Em verde: Pontos de acesso da rede Wi-Fi do laboratório

Em vermelho: Pontos do dispositivo teste

A análise de dados foi feita com a função *summary* da ferramenta *Ron's editor* e a filtragem com a função *Filter* da ferramenta *RecCsvEditor*

<https://www.ronsplace.eu/Products/RonsEditor>

<http://recsveditor.sourceforge.net/>

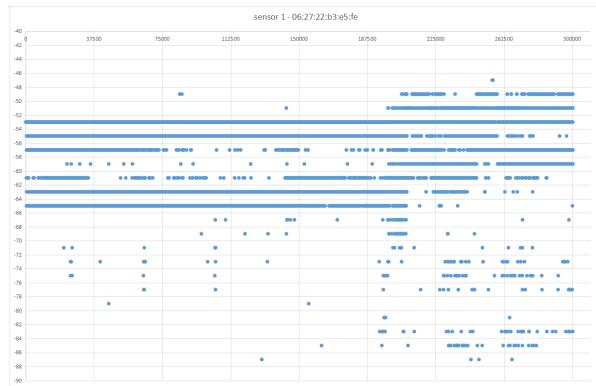
Para o primeiro sensor foram detidos 1 729 624 pacotes num arquivo de 155MB com 88 transmissores únicos dos quais se destacaram dois endereços MAC que são os pontos de acesso para rede wi-fi do laboratório.

Figura 20 – Sinal em dBm por pacote capturado - 062722b3e5fb sensor 1



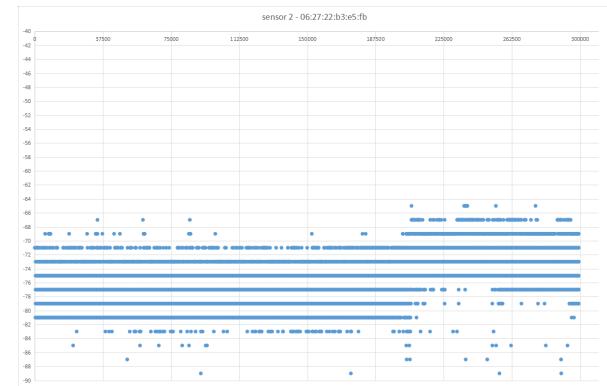
Fonte: Elaborada pelo autor

Figura 22 – Sinal em dBm por pacote capturado - 062722b3e5fe sensor 1



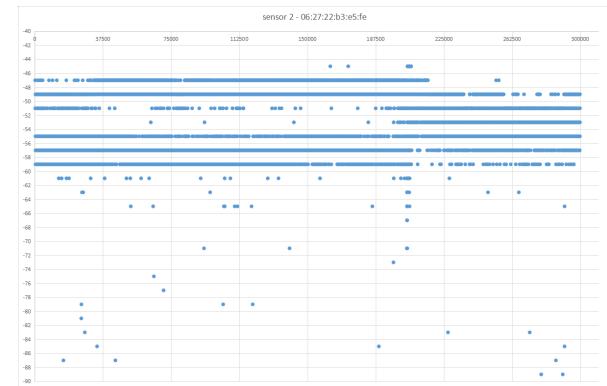
Fonte: Elaborada pelo autor

Figura 21 – Sinal em dBm por pacote capturado - 062722b3e5fb sensor 2



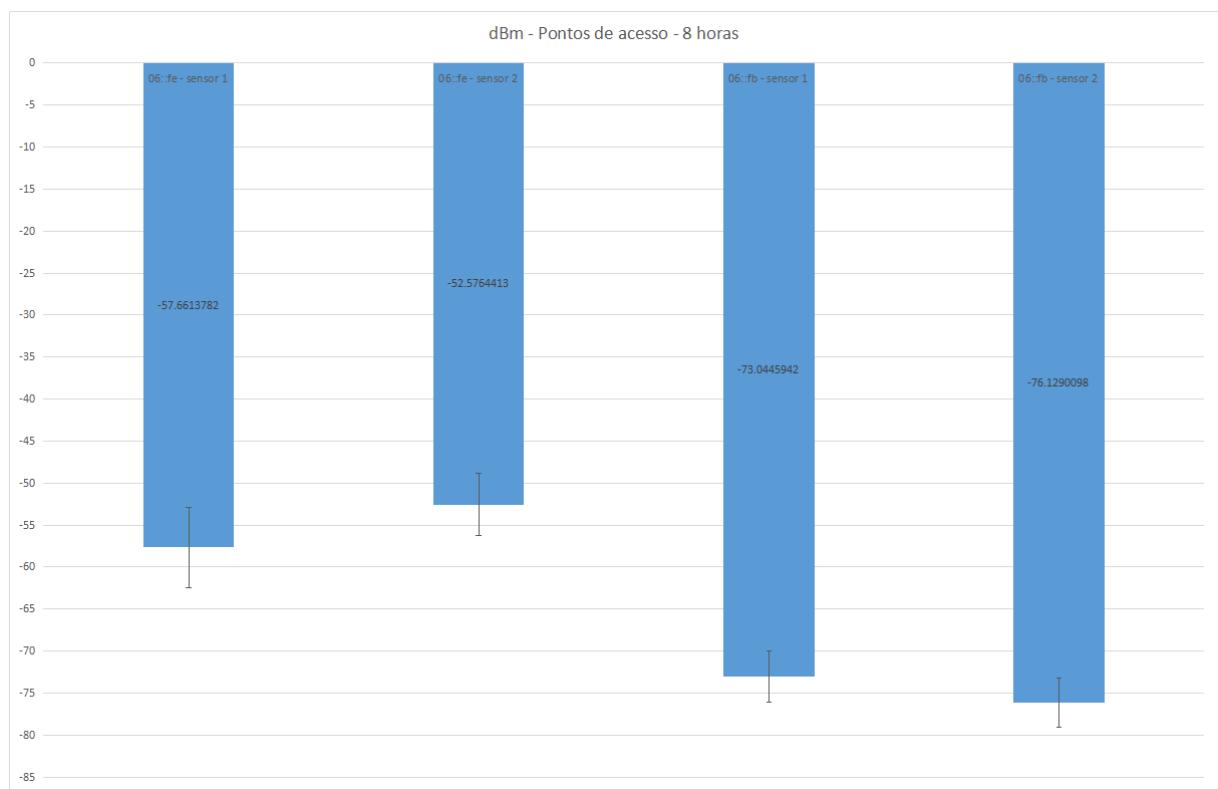
Fonte: Elaborada pelo autor

Figura 23 – Sinal em dBm por pacote capturado - 062722b3e5fe sensor 2



Fonte: Elaborada pelo autor

Figura 24 – dBm Pontos de acesso - Acumulado 8 horas



Fonte: Elaborada pelo autor

6.3 Teste relação de distância com smartphone como objetivo

Para verificar a capacidade do sensores de localizar contextualmente um dispositivo móvel, smartphone, foi utilizado. Este foi posicionado em duas salas diferentes, em cada uma das salas foi executada uma captura de 10 minutos. Para que houvesse tráfego na rede o dispositivo móvel foi configurado para receber um stream de vídeo no aplicativo *Netflix*.

A captura foi realizada utilizando a ferramenta TShark da mesma maneira que é utilizada no aplicativo. A descrição deste modo de operação pode ser encontrada no capítulo de construção.

Código-fonte 6.2 – TShark e redirecionamento da saída para arquivo

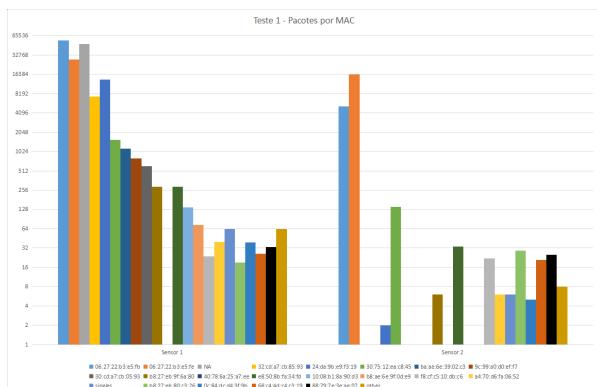
```
1 pi@sensor-01:~ $ tshark -l -i wlan1 -T fields -E separator=, -E quote=d \
2 -e wlan.sa -e wlan.sa_resolved -e wlan.ta -e wlan.ta_resolved \
3 -e radiotap.dbm_antsignal -e wlan_mgt.ssid >> sensor-02-distance-test-01.csv
4 pi@sensor-01:~ $
```

Para o primeiro caso o dispositivo estava na mesma sala do sensor número 2 que é a maior sala do prédio com 12 metros de comprimento por 10 metros De largura. neste caso foram capturados 157736 pacotes totalizando 9.7 megabytes pelo sensor 1 21974 pacotes totalizando 1.9 megabytes de captura pelo sensor 2.

No segundo teste o dispositivo móvel estava posicionado no corredor fora da sala do sensor 1 e distante do sensor 2. neste teste o sensor 1 capturou 103555 pacotes totalizando 6.4 megabytes de captura e o sensor 2 capturou 22635 pacotes totalizando 2 megabytes de captura

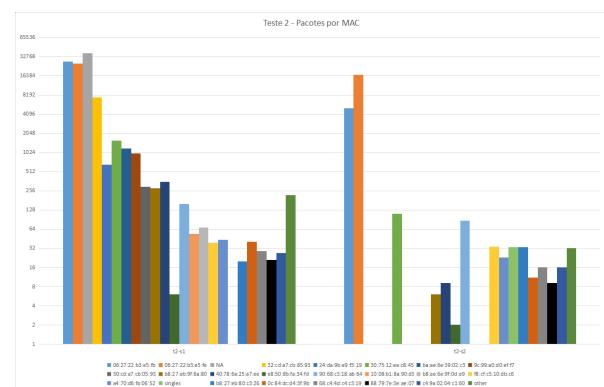
Posteriormente os arquivos de captura foram analisados com a ferramenta *Ron's Editor* para que um sumário fosse construído.

Figura 25 – Captura total (noise) - Teste 1



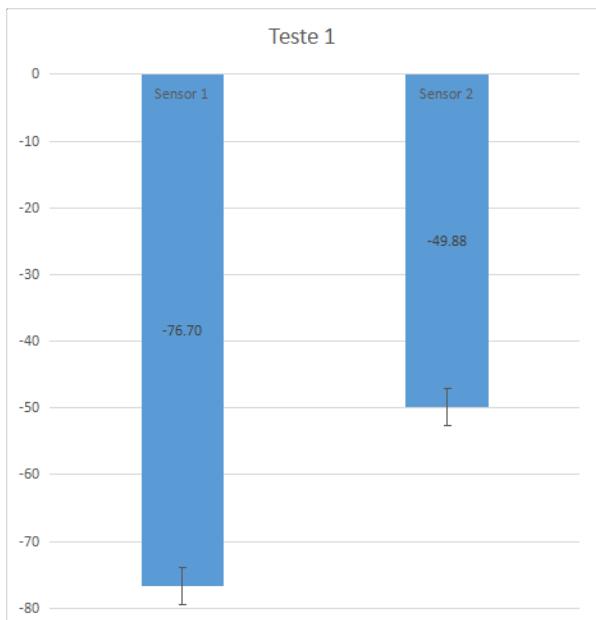
Fonte: Elaborada pelo autor

Figura 26 – Captura total (noise) - Teste 2



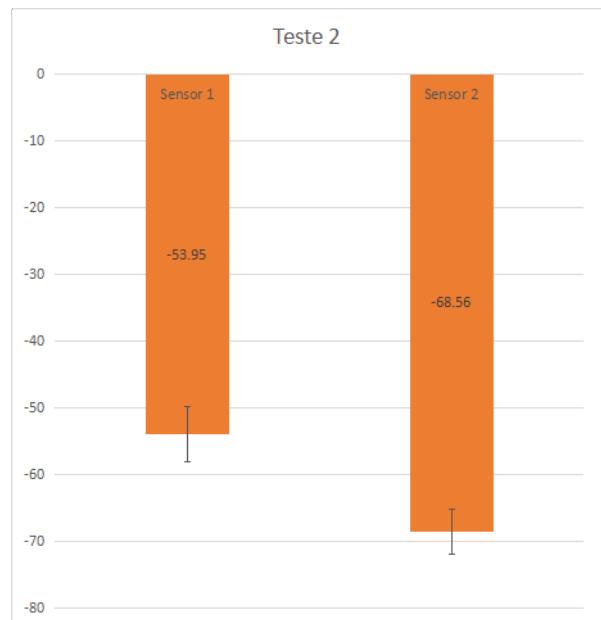
Fonte: Elaborada pelo autor

Figura 27 – dBm Motorola G4+ - Teste 1



Fonte: Elaborada pelo autor

Figura 28 – dBm Motorola G4+ - Teste 2



Fonte: Elaborada pelo autor

7 CONCLUSÃO

Durante a exploração do tema foram encontradas diversas implementações de localizadores baseados em Wi-Fi mas a implementação aqui executada mais se assemelha com a de (FERREIRA, 2016) onde a mesma plataforma (Raspberry Pi porém na sua versão 2 modelo B+), tipo de adaptador (Wi-Fi USB) e *software* (*TShark*) foram utilizadas. A principal diferença são os objetivos, enquanto a localização que Ferreira (2016) buscou é do tipo geográfica, neste trabalho buscou-se o objetivo mais simples de encontrar o grau de presença do dispositivo no mesmo contexto (sala) do sensor. Outras diferenças são que alguns desafios propostos por Ferreira (2016) foram atacados com certo nível de sucesso, entre eles: mais de um dispositivo sensor, coleta e processamento simultâneos (*online*), registro do histórico. Outros que não renderam frutos também merecem atenção, como a exploração adicional da plataforma ESP8266 que aqui propõem-se como trabalho futuro.

REFERÊNCIAS

- ABUSUBAIH, M.; RATHKE, B.; WOLISZ, A. A Dual Distance Measurement Scheme for Indoor IEEE 802 . 11 Wireless Local Area Networks *. In: *2007 9th IFIP International Conference on Mobile Wireless Communications Networks*. [s.n.], 2007. p. 121–125. ISBN 9781424417209. ISSN 978-1-4244-1719-3. Disponível em: <<http://ieeexplore.ieee.org/document/4668193/>>. 21
- alibaba.com. *Esp-12f Esp8266 Remote Serial Port Wifi Transceiver Wireless Module - Buy Esp8266, Wifi Module, Esp12f Product on Alibaba.com*. 2017. Disponível em: <https://www.alibaba.com/product-detail/ESP-12F-Esp8266-Remote-Serial-Port_60518607068.html?s=p>. Acesso em: 2017-01-19. 27
- AMAZON. *AWS IoT*. 2016. 1–8 p. Disponível em: <<https://aws.amazon.com/pt/iot/>>. 12
- AMAZON. *Definição de preço do AWS IoT – Amazon Web Services*. 2016. 2 p. Disponível em: <<https://aws.amazon.com/pt/iot/pricing/>>. Acesso em: 2016-01-01. 12
- ARM. *Welcome to mbed*. 2016. Disponível em: <<https://www.mbed.com/en/>>. Acesso em: 2016-04-19. 12
- ASHTON, K. That internet of things thing. *RFID Journal*, v. 22, n. 7, p. 4986, 2009. Disponível em: <<http://www.rfidjournal.com/articles/view?4986>>. 19
- ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. *Computer Networks*, Elsevier B.V., v. 54, n. 15, p. 2787–2805, 2010. ISSN 13891286. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S1389128610001568>>. 19
- BAHILLO, A. et al. IEEE 802.11 distance estimation based on RTS/CTS two-frame exchange mechanism. In: *IEEE. Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*. 2009. p. 1–5. Disponível em: <<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5073583>>. 21, 24
- BELLAVISTA, P.; KÜPPER, A.; HELAL, S. Location-based services: Back to the future. *IEEE Pervasive Computing*, v. 7, n. 2, p. 85–89, 2008. ISSN 15361268. 20
- BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, IGI Global, v. 5, n. 3, p. 1–22, jan 2009. ISSN 1552-6283. Disponível em: <<http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf>>. 21
- C2FO.COM. *C2FO/fast-csv: CSV parser for node*. 2016. Disponível em: <<http://c2fo.io/fast-csv/>>. Acesso em: 2017-01-23. 44
- CHEN, G.; KOTZ, D. *A Survey of Context-Aware Mobile Computing Research*. [S.I.], 2000. v. 3755, n. TR2000-381, 1–16 p. Disponível em: <<http://www.cs.dartmouth.edu/reports/abstracts/TR2000-381/>>. 14
- Cisco Blog. *How Many Internet Connections are in the World? Right. Now.* 2013. 2 p. Disponível em: <<http://blogs.cisco.com/news/cisco-connections-counter>>. Acesso em: 2016-04-20. 12

- CROW, B. et al. IEEE 802.11 Wireless Local Area Networks. *IEEE Communications Magazine*, v. 35, n. 9, p. 116–126, 1997. ISSN 01636804. Disponível em: <<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=620533>>. 21
- DETERS, J. *MQTT.fx*. 2017. Disponível em: <<http://mqttfx.org/>>. Acesso em: 2017-01-23. 49
- DEY, A. K. Providing Architectural Support for Building Context-Aware Applications. *Sensors Peterborough NH*, v. 16, n. November, p. 97–166, 2000. ISSN 07370024. Disponível em: <<http://portal.acm.org/citation.cfm?id=932974>>. 20
- DEY, A. K.; ABOWD, G. D. Towards a Better Understanding of Context and Context-Awareness. *Computing Systems*, v. 40, n. 3, p. 304–307, 1999. ISSN 00219266. 20
- DJUKNIC, G. M.; RICHTON, R. E. Geolocation and assisted GPS. *Computer*, v. 34, n. 2, p. 123–125, 2001. ISSN 0018-9162. Disponível em: <<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=901174>>. 14
- DZONE. THE DZONE GUIDE TO THE INTERNET OF THINGS. *DZone*, p. 32, 2015. Disponível em: <<https://dzone.com/guides/internet-of-things-1>>. 12
- eclipse. *eclipse/paho.mqtt-spy: mqtt-spy is an open source desktop and command line utility intended to help you with monitoring activity on MQTT topics*. 2016. Disponível em: <<https://github.com/eclipse/paho.mqtt-spy>>. Acesso em: 2017-01-23. 49
- ESP8266.NET. *ESP8266.net home*. 2016. Disponível em: <<http://esp8266.net/>>. Acesso em: 2016-03-29. 12
- Espressif Systems. *Home - espressif/ESP8266_AT Wiki*. 2014. 4–5 p. Disponível em: <https://github.com/espressif/ESP8266{_}AT/w>. Acesso em: 2017-01-20. 31
- FELDMANN, S. et al. An indoor Bluetooth-based positioning system : concept , Implementation and experimental evaluation. *International Conference on Wireless Networks*, n. JANUARY 2003, p. 109–113, 2003. ISSN 09621105. Disponível em: <https://www.researchgate.net/publication/220719209{_}An{_}Indoor{_}Bluetooth-Based{_}Positioning{_}System{_}Concept{_}Implementation{_}and>. 21
- FERREIRA, L. C. P. *Sistema localizador interior de baixo custo*. 2016. Disponível em: <<http://repositorio.ipl.pt/handle/10400.21/6162>>. 21, 59
- FRIEDEMANN, M.; FLOERKEMEIR, C. From the Internet to the Internet of Things. *From Active Data Management to Event-Based Systems and More*, p. 242–259, 2011. ISSN 0302-9743. Disponível em: <<http://www.ulb.tu-darmstadt.de/tocs/79304567.pdf>>. 19
- FUNDATION, R. *Raspberry Pi Zero: the \$5 computer*. 2015. 2 p. Disponível em: <<https://www.raspberrypi.org/blog/raspberry-pi-zero/>>. Acesso em: 2016-04-19. 12
- GARTNER. *Gartner Says the Internet of Things Will Transform the Data Center*. 2014. 5 p. Disponível em: <<http://www.gartner.com/newsroom/id/2684616>>. Acesso em: 2016-04-20. 12

- GARTNER. *Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015.* 2015. 1 p. Disponível em: <<http://www.gartner.com/newsroom/id/3165317>>. Acesso em: 2016-03-28. 12
- GITHUB.COM. *Integrations with Deployment.* 2016. Disponível em: <<https://github.com/integrations/feature/deployment>>. Acesso em: 2017-01-23. 47
- GITHUB.COM/MQTTJS. *mqttjs/MQTT.js: The MQTT client for Node.js and the browser.* 2016. Disponível em: <<https://github.com/mqttjs/MQTT.js>>. Acesso em: 2017-01-23. 46
- GOOGLE. *Google for Internet of Things.* 2016. 1–5 p. Disponível em: <<https://cloud.google.com/solutions/iot/>>. 12
- HOSSAIN, A. K. M. M.; SOH, W.-S. A Comprehensive Study of Bluetooth Signal Parameters for Localization. In: *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications.* IEEE, 2007. p. 1–5. ISBN 978-1-4244-1143-6. ISSN 2166-9570. Disponível em: <<http://ieeexplore.ieee.org/xpl/login.jsp?tp={&}arnumber=853>>. 21
- IBM. *IBM IoT.* 2016. 1–5 p. Disponível em: <<http://www.ibm.com/internet-of-things/>>. Acesso em: 2016-01-01. 12
- INTEL. *IoT Solutions / IntelDeveloper Zone.* 2016. 1–4 p. Disponível em: <<https://software.intel.com/en-us/articles/a-fast-flexible-and-scalable-path-to-commercial-iot-solutions>>. 12
- International Telecommunication Union. Overview of the Internet of things. Series Y: *Global information infrastructure, internet protocol aspects and next-generation networks - Frameworks and functional architecture models*, p. 22, 2012. Disponível em: <<http://handle.itu.int/11.1002/1000/11559>>. 19
- JAMES, M. The Ultimate Scrum Reference Card. *Dzone*, p. 6, 2016. Disponível em: <<https://dzone.com/refcardz/scrum>>. 24
- KAUFMANN, A.; DOLAN, K. *Price Comparison: Google Cloud vs AWS.* [S.I.], 2015. 16 p. Disponível em: <<https://cloud.google.com/files/esg-whitepaper.pdf>>. 12
- KRANENBURG, R. van. The Sensing Planet: Why The Internet Of Things Is The Biggest Next Big Thing. *Co.CREATE*, p. 1–8, 2012. Disponível em: <<http://www.fastcocreate.com/1681563/the-sensing-planet-why-the-internet-of-things-is-the-biggest-next-big-thing>>. 12
- LANZISERA, S.; ZATS, D.; PISTER, K. S. J. Radio Frequency Time-of-Flight Distance Measurement for Low-Cost Wireless Sensor Localization. *IEEE Sensors Journal*, v. 11, n. 3, p. 837–845, mar 2011. ISSN 1530-437X. Disponível em: <<http://ieeexplore.ieee.org/document/5701645/>>. 23
- LEMOS, A. A Comunicação das Coisas: Internet das Coisas e Teoria Ator-Rede. p. 1–23, 2013. 12
- mercadolivre.com.br. *Modulo Esp8266 Esp-12e Wireless Wifi em Mercado Livre.* 2017. Disponível em: <<https://produto.mercadolivre.com.br/MLB-788100073-modulo-esp8266-esp-12e-wireless-wifi>>. Acesso em: 2017-01-20. 27

- mercadolivre.com.br. *Novo Raspberry Pi 3 (pi3) Quadcore 1.2ghz (10x+rapido) 1gb - RS 189,99 em Mercado Livre*. 2017. Disponível em: <<https://produto.mercadolivre.com.br/MLB-745485654-novo-raspberry-pi-3-pi3-quadcore-12ghz-10xrapido-1gb>>. Acesso em: 2017-01-20. 33
- MICROSOFT. *The Internet of Your Things*. 2016. Disponível em: <<https://dev.windows.com/en-US/iot>>. 12
- MOSQUITTO.ORG. *An Open Source MQTT v3.1 Broker*. 2016. Disponível em: <<http://mosquitto.org/>>. Acesso em: 2017-01-23. 49
- Nghia TH. *IoT MQTT Dashboard - Android Apps on Google Play*. 2016. Disponível em: <<https://play.google.com/store/apps/details?id=com.thn.iotmqtdashboard>>. Acesso em: 2017-01-23. 49
- NODEJS.ORG. *Child Process / Node.js v6.9.4 Documentation*. 2016. Disponível em: <<https://nodejs.org/dist/latest-v6.x/docs/api/childprocess.html>>. Acesso em: 2017-01-23. 43
- NODEJS.ORG. *Node.js*. 2016. Disponível em: <<https://nodejs.org/en/>>. Acesso em: 2017-01-22. 43
- NODEJS.ORG. *Stream / Node.js v6.9.4 Documentation*. 2016. Disponível em: <<https://nodejs.org/dist/latest-v6.x/docs/api/stream.html>>. Acesso em: 2017-01-23. 44
- ORACLE. *Oracle IoT*. 2016. 3–5 p. Disponível em: <<https://www.oracle.com/solutions/internet-of-things/index.html>>. 12
- PASCALAU, E.; NALEPA, G. J.; KLUZA, K. Towards a Better Understanding of. *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, p. 959–962, 2013. ISSN 1743-9213. 20
- PERLROTH, N. *Hackers Used New Weapons to Disrupt Major Websites Across U.S.* *The New York Times*. 2016. Disponível em: <<https://www.nytimes.com/2016/10/22/business/internet-problems-attack.html>>. Acesso em: 2017-01-20. 36
- RASPBERRY PI FOUNDATION. *Raspberry Pi 3 Model B - Raspberry Pi*. 2016. Disponível em: <<https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>>. Acesso em: 2016-05-16. 22, 33
- ROMÁN, M.; CAMPBELL, R. H. A Model for Ubiquitous Applications. 2001. Disponível em: <<http://www.ncstrl.org:8900/ncstrl/servlet/search>>. 21
- ROOM-15. *ESP8266 - AT Command Reference*. 2015. 1–21 p. Disponível em: <<https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/>>. Acesso em: 2017-01-20. 31
- SOKOLOVSKY, P. *pfalcon/esp-open-sdk: Free and open (as much as possible) integrated SDK for ESP8266/ESP8285 chips*. 2017. 1–4 p. Acesso em: 2017-01-20. 31
- VASISHT, D.; KUMAR, S.; KATABI, D. Decimeter-Level Localization with a Single WiFi Access Point This paper is included in the Proceedings of the Decimeter-Level Localization with a Single WiFi Access Point. *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)*, 2016. Disponível em: <<https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/vasisht>>. 23

- VUJOVIĆ, V.; MAKSIMOVIĆ, M. Raspberry Pi as a Sensor Web node for home automation. *Computers & Electrical Engineering*, v. 44, p. 153–171, feb 2015. ISSN 00457906. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0045790615000257>>. 21
- VUJOVIC, V. et al. Raspberry Pi as Internet of Things hardware : Performances and Constraints Raspberry Pi as Internet of Things hardware : Performances and Constraints. n. JUNE, 2014. 21
- WEISER, M. The Computer for the 21st Century. *Scientific American*, v. 265, n. 3, p. 3–11, sep 1999. ISSN 0036-8733. Disponível em: <<http://www.nature.com/doifinder/10.1038/scientificamerican0991-94><http://dl.acm.org/citation.cfm?doid=329124.329126>>. 12
- WIRESHARK.ORG. *Wireshark User's Guide*. 2014. Disponível em: <https://www.wireshark.org/docs/wsug_html/#AppToolstshark>. Acesso em: 2017-01-23. 43
- WOOLF, N. *DDoS attack that disrupted internet was largest of its kind in history, experts say Technology* *The Guardian*. 2016. Disponível em: <<https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet>>. Acesso em: 2017-01-20. 36
- WORTMANN, F.; FLÜCHTER, K. *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. [s.n.], 2015. v. 57. 221–224 p. ISSN 1867-0202. Disponível em: <<http://dx.doi.org/10.1007/s12599-015-0383-3>>. 19
- Zappy Inc. *zapty/forever-service: Provision node script as a service via forever, allowing it to automatically start on boot, working across various Linux distros and OS*. 2016. Disponível em: <<https://github.com/zappy/forever-service>>. Acesso em: 2017-01-23. 48

Apêndices

APÊNDICE A – COMPRAS MERCADO LIVRE

1/20/2017

Compras - Mercado Livre

 <input type="text"/> Search			
Compras			
Compras:Todas			
Fernando Moreni 11 981029845 Enviar mensagem		Esp8266 Placa Para Soldar - Esp-07, Esp-08, Esp-12, Esp-12e R\$ 3,45 x 5 unidades	 Pago  Entregue <input type="checkbox"/> Você o qualificou positivo
Fernando Moreni 11 981029845 Enviar mensagem		Módulo Esp8266 Esp-12e Wireless Wifi R\$ 14,90 x 5 unidades	 Pago  Entregue <input type="checkbox"/> Você o qualificou positivo
Filipe William Pinto fwp@fwp.com.br 19 3203-6397		Ams1117 3,3v (3.3v) - Lm1117 (10 Unidades) - Frete Grátis R\$ 14,99 x 1 unidade	 Pago  A caminho <input type="checkbox"/> Você o qualificou positivo
Jose Barreto Da Paixao - Me jkeletronicos1@gmail.com 1156111482		Kit Lentes Celular Fisheye Macro Wide Moto G Z3 S5 Note 4 5c R\$ 6,99 x 1 unidade	 Pago  Entregue <input type="checkbox"/> Você o qualificou positivo
Gabriel Fernandes Dos Santos gblimport10@hotmail.com 11 28879260		Mini Adaptador Wireless Wifi Edup Usb 150mbps Raspberry Pi R\$ 16,88 x 3 unidades	 Pago  Entregue <input type="checkbox"/> Você o qualificou positivo
Michelle Alves rgomes@outlook.com.br 11972329379		Esp-01 - Módulo Transceptor Serial Wifi Esp8266 R\$ 16,80 x 1 unidade	 Pago  Entregue <input type="checkbox"/> Você o qualificou positivo
Vision Adapter 11959463754 Enviar mensagem		Adaptador Usb Serial Ttl Conversor Cp2102 R\$ 20,00 x 1 unidade	 Pago  Entregue <input type="checkbox"/> Você o qualificou positivo
Clóvis Fritzen clovisf@gmail.com 47 9204-6606		Conversor Usb Serial Ch340 Rs232 - 3,3v 5v R\$ 6,87 x 1 unidade	 Pago  Entregue <input type="checkbox"/> Você o qualificou positivo