UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Mauricio Machado Lourenço

DESENVOLVIMENTO DO PROTÓTIPO DE UM SISTEMA DE LOCALIZAÇÃO COM MÓDULO WI-FI

Santa Maria, RS, Brasil
2016

Mauricio Machado Lourenço

DESENVOLVIMENTO DO PROTÓTIPO DE UM SISTEMA DE LOCALIZAÇÃO COM MÓDULO WI-FI

Monografia apresentada ao curso de Engenharia de Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Engenheiro de Computação.**

Orientador Prof. Dr. Carlos Henrique Barriquello

Santa Maria, RS
2016

Mauricio Machado Lourenço

DESENVOLVIMENTO DO PROTÓTIPO DE UM SISTEMA DE LOCALIZAÇÃO COM MÓDULO WI-FI

Monografia apresentada ao curso de Engenharia de Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Engenheiro de Computação.**

Aprovado em 13 de dezembro de 2016:

_____
Carlos Henrique Barriquello, Dr. (UFSM)
(Presidente/Orientador)

_____
José Eduardo Baggio, Dr. (UFSM)

_____
Mateus Beck Rutzig, Dr. (UFSM)

Santa Maria, RS
2016

RESUMO

DESENVOLVIMENTO DO PROTÓTIPO DE UM SISTEM DE LOCALIZAÇÃO COM MÓDULO WI-FI

AUTOR: MAURICIO MACHADO LOURENÇO
ORIENTADOR: CARLOS HENRIQUE BARRIQUELLO

Este trabalho tem como objetivo o desenvolvimento do protótipo de um sistema de localização constituído por um módulo Wi-Fi (ESP8266) e um aplicativo de celular *Android*, o qual detectará a presença desse módulo por meio do serviço de scan Wi-Fi do *Android* e calculará a distância entre o módulo e o celular, definida pela posição do GPS do aparelho até o módulo, através da potência do sinal do módulo recebida pelo celular, utilizando a técnica da Potência do Sinal Recebido (RSSI - *Received Signal Strength Indicator*). A principal motivação desse projeto é a criação de um produto para localizar animais domésticos, uma coleira com o módulo Wi-Fi "embarcado" a qual será localizada pelo aplicativo de celular desenvolvido. Porém, esse sistema pode ser utilizado para diversas outras aplicações, como localizar outros objetos em diversos ambientes, exemplo: bagagens em aeroportos. Essas aplicações estão diretamente relacionadas ao conceito da Internet das Coisas, que diz que objetos do nosso dia-a-dia podem se conectar à internet para realizar diversas funções. Foram feitos testes de consumo de bateria pelo módulo e de distância calculada pelo aplicativo para validar o funcionamento do sistema desenvolvido, com base nos resultados obtidos foi feito um ajuste para melhorar a precisão do sistema.

Palavras-chave: Sistema de Localização Wi-Fi. Andoid. Potência do Sinal Recebido. Internet das Coisas.

ABSTRACT

PROTOTYPE DEVELOPMENT OF A TRACKING SYSTEM WITH WI-FI MODULE

AUTHOR: MAURICIO MACHADO LOURENÇO
ADVISOR: CARLOS HENRIQUE BARRIQUELLO

This paper aims developing a prototype of a location system composed of by a Wi-Fi (*Wireless Fidelity)* module and a Android mobile application, which will detect the presence of this module by the Wi-Fi scan service of Android and will calculate the distance between the module and the cellphone, defined by the position of the smartphone to the module, through the signal strenght of the module received by the cellphone., using the technique of the Received Signal Strenght Indicator (RSSI). The main motivation of this Project is the creation of a product to find domestic animals, a leash with the Wi-Fi module “embedded” which will be find by the mobile application developed. However, this system can be used for several other applications, such as finding other objects in several environments, example: lugages at an airport. This applications are directly related with the concept of the Internet of Things, which says that objects of our dayly use can connect to the internet to perform various functions. Tests were perfomed, battery consuption by the module and calculed distance by the application, to validate the operation of the developed system, based on the results obtained an adjustment was made to improve the accuracy of the system.

Keywords: Wi-Fi Location System. Android. Received Signal Strength Indicator. Internet of the Things.

SUMÁRIO

1. INTRODUÇÃO

O presente trabalho tem como objetivo a criação do protótipo de um sistema de localização de baixo custo, em alternativa ao uso do sistema GPS (*Global Positioning System)*, que utiliza um módulo Wi-Fi (*Wireless Fidelity*), definido como *Access Point*, esse módulo será localizado pelo aplicativo implementado para o sistema de dispositivos móveis *Android*. A ideia é utilizar esse protótipo para criar uma coleira a qual será ser identificada pelo aplicativo desenvolvido, possibilitando assim que animais de estimação possam ser localizados por seus donos ou outras pessoas.

Para localizar um objeto no espaço precisa-se de uma referência, por exemplo a latitude, longitude e altitude determinam a posição absoluta de um ponto na Terra e são referenciados em relação aos meridianos principais (Greenwich e Equador). O sistema desenvolvido neste projeto determina uma posição relativa em relação a uma posição absoluta do GPS do celular, sem levar em conta a altitude.

Esse sistema foi idealizado com o propósito de localizar animais perdidos, porém, o mesmo pode ser aplicado para localizar outros objetos, bastando apenas adaptar a aplicação para a nova função.

O aplicativo implementado apresentou resultados com uma precisão de distância calculada suficiente para as aplicações propostas e, uma boa autonomia de bateria para o módulo Wi-Fi. As informações de localização são enviadas para um servidor na nuvem, explorando a conectividade dos *smartphones*, onde os dados ficam armazenados de forma segura na nuvem.

1.1. MOTIVAÇÃO

Um sistema de localização tem como principal função rastrear um objeto e determinar sua posição geográfica no espaço. Inicialmente, o sistema de localização desenvolvido nesse trabalho foi idealizado com o objetivo de criar uma coleira para localizar animais perdidos para seus donos, porém, esse sistema pode ser usado para localizar outros objetos, bastando apenas adaptar o aplicativo para que ele funcione de acordo com o esperado.

Para que esse sistema funcione são necessários dois requisitos: um módulo Wi-Fi (ESP8266), que emitirá o sinal a ser detectado e um *smartphone* com sistema Android, e acesso à internet, o aplicativo identificará o sinal do módulo e enviará e/ou receberá as informações de localização para um servidor na nuvem. Como cada módulo possui um id único (conhecido por endereço MAC, *Media Access Control*) é possível identificar cada dispositivo de forma exclusiva. Assim, basta “embarcar” esse módulo a qualquer objeto que desejamos rastrear e utilizar a aplicação para localizá-lo.

Esse sistema pode ser utilizado de modo cooperativo, ou seja, outras pessoas utilizam o aplicativo para localizar os objetos, ou de forma particular, onde o próprio dono do objeto utiliza a aplicação para encontra-lo.

Possíveis usos para esse sistema de localização, além de localizar animais perdidos são: localizar bagagens em aeroportos; na Agropecuária para localizar gado em um campo; encontrar pessoas como crianças ou idosos em lugares com muito movimento, obter a localização de automóveis em estacionamentos, etc.

1.2. OBJETIVOS

1.2.1. Objetivo geral

O objetivo geral deste projeto é implementar o protótipo de um sistema de localização com o módulo Wi-Fi. Este sistema será constituído por um módulo Wi-Fi e um aplicativo *Android* desenvolvido.

1.2.2. Objetivos específicos

i. Programar o módulo configurando-o como *Access Point,* criando assim uma rede Wi-Fi a qual será detectada pelo aplicativo de celular através do SSID.
ii. Implementar a aplicação *Android* para detectar o sinal do módulo e calcular a distância através da potência do sinal recebido.
iii. Verificar a eficiência do sistema com testes de precisão e estimativa de consumo de bateria pelo módulo Wi-Fi.

1.3. ESTRUTURA DO TRABALHO

No capítulo 1, inicialmente é feita uma abordagem da motivação para o desenvolvimento deste trabalho, com uma explicação da importância de um sistema de localização e possíveis aplicações, em seguida, são apresentados os objetivos gerais e específicos.

No capítulo 2, é feita uma revisão teórica sobre os principais tópicos relacionados ao trabalho como Internet das Coisas (*Internet of Things*), uma apresentação do módulo ESP8266 com uma revisão sobre o protocolo Wi-Fi, como se localizar no espaço e as principais técnicas de localização em radiofrequência e por fim uma sobre o sistema GPS com detalhes de como surgiu e detalhes de funcionamento.

No capítulo 3, são apresentados aspectos sobre a metodologia do trabalho. Primeiramente é feita uma descrição do sistema desenvolvido, a seguir, são

apresentadas as APIs (*Application Programming Interface*) utilizadas no desenvolvimento do aplicativo Android e por fim são apresentados alguns diagramas utilizando a linguagem UML, tais como diagramas de casos de uso, diagrama de classes e diagrama de sequência das principais funcionalidades do sistema.

No capítulo 4, são apresentados os resultados obtidos através de testes de alcance e de consumo de bateria pelo módulo.

No capítulo 5, são feitas as conclusões como discussão dos aspectos do sistema implementado, resultados obtidos, possíveis melhorias e tópicos para trabalhos futuros.

2. REVISÃO TEÓRICA

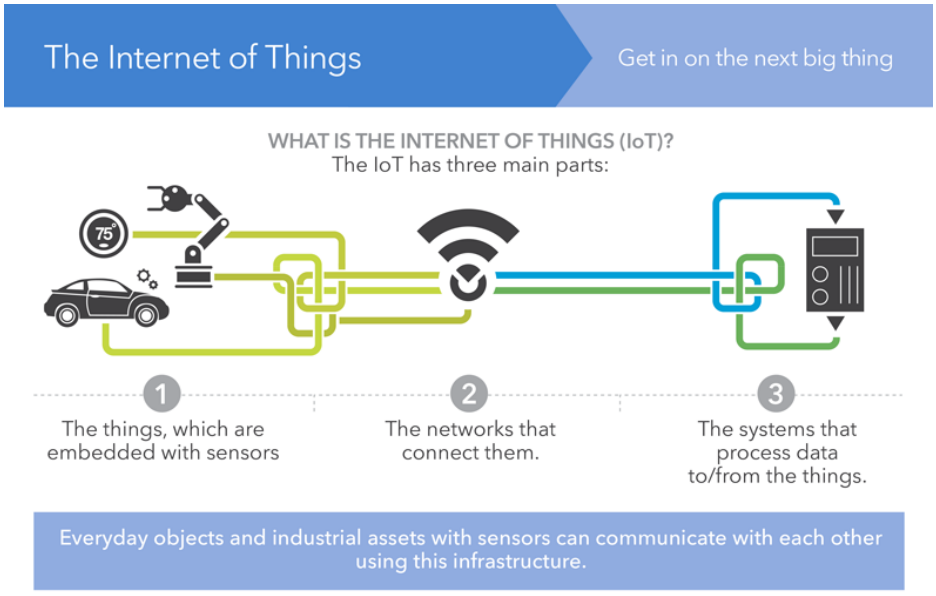
2.1. INTERNET DAS COISAS

Atualmente, a Internet das Coisas (*Internet of Things* - IoT) vem ganhando grande destaque no cenário das telecomunicações e está sendo considerada a revolução tecnológica que representa o futuro da computação e comunicação (Tan e Wang 2010; Atzori et al. 2010 apud França et al. 2011).

A ideia central do paradigma da Internet das Coisas é permitir que objetos que nos rodeiam em nosso dia-a-dia se conectem à internet. Esses objetos podem ser quaisquer dispositivos, tais como eletrodomésticos, pneus, sensores, atuadores, telefones celulares, entre outros, que possam ser identificados e interligados a internet para trocar informações e tomar decisões para atingir objetivos comuns (Atzori et al. 2010 apud França et al. 2011).

De acordo com o infográfico da figura 2.1.1, a Internet das coisas é formada por três partes principais: As coisas, as quais possuem sensores embarcados; as redes que conectam os objetos e, os sistemas que processam dados de/para as coisas.

Figura 2.1.1 – A internet das Coisas



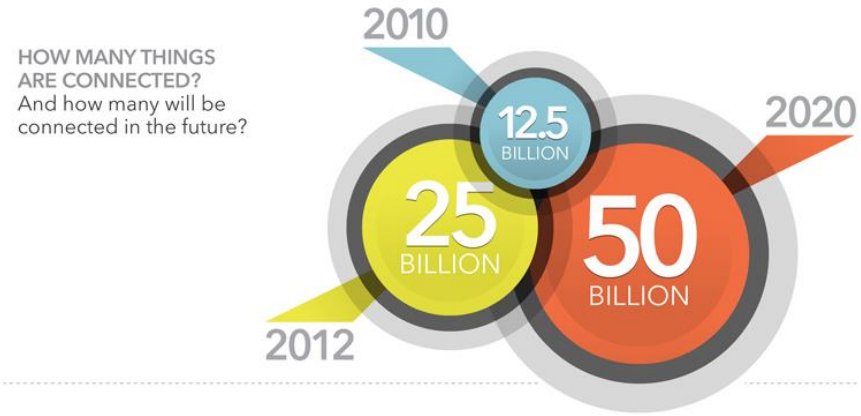
(Fonte: Infográfico SAS).

Para Almeida (2015, p. 8)

> A empolgação atual com IoT é fruto da convergência de diversas tecnologias. Em primeiro lugar, a miniaturização e popularização de sensores viabilizam a coleta e transmissão de dados, com estimativa de mais de 40 bilhões de dispositivos conectados em 2020 (ABI Research, 2013). Tal conectividade é viabilizada pelo avanço das redes sem fio, tornando onipresente o acesso e a transmissão dos dados para a Internet.

O infográfico da figura 2.1.2 mostra que em 2020 teremos 50 bilhões de objetos conectados à internet, o que representa um futuro promissor na área de dispositivos embarcados e aplicações relacionadas à Internet das Coisas.

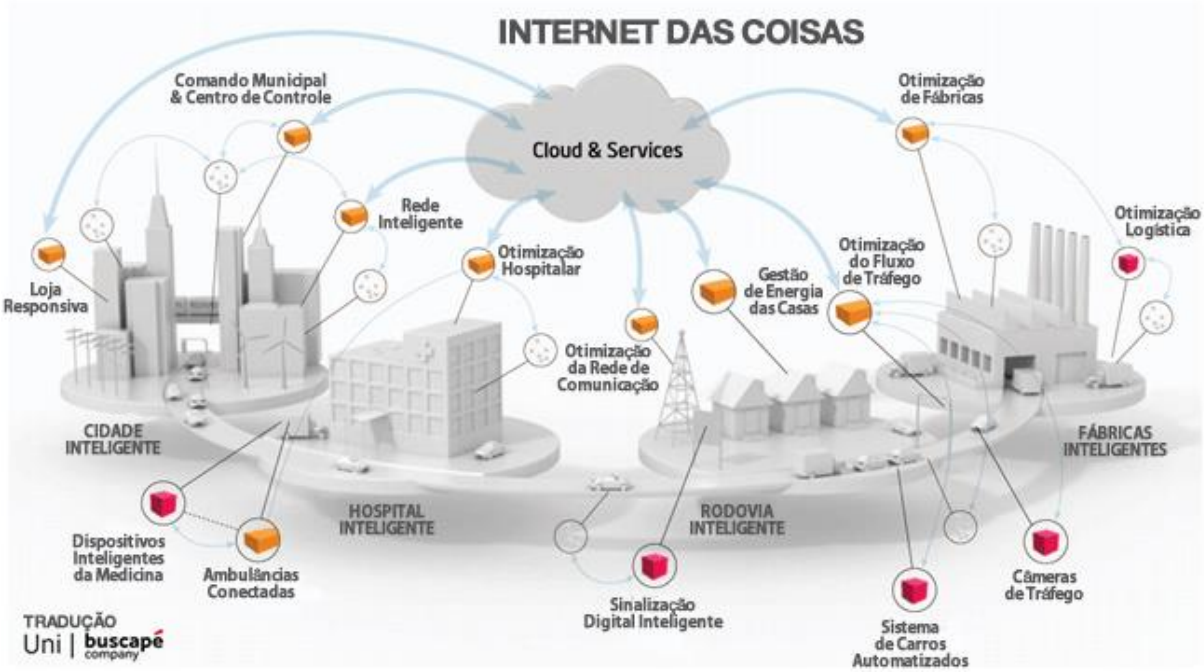
Figura 2.1.2 – Número de coisas conectadas



(Fonte: Infográfico SAS).

Algumas das aplicações da Internet das Coisas podem ser observadas na imagem da figura 2.1.3, esses são apenas alguns exemplos existem inúmeras possibilidades de aplicações nessa área que como visto antes só tende a expandir.

Figura 2.1.3 – Aplicações Internet das Coisas



(Fonte: Ferreira).

2.1.2. PADRÕES DE COMUNICAÇÃO SEM FIO PARA IOT

A Internet das Coisas possibilita inúmeras funcionalidades através da interação de dispositivos. Dentre os padrões de comunicação para essa funcionalidade destacam-se: RFID (NFC), Bluetooth Classic e Bluetooth *Low Energy*, para comunicação a curta distância; *Zig-Bee* e *Z-Wave*, para comunicações em malha; 3G/4G/LTE geralmente utilizados em dispositivos que operam em locais remotos e, O padrão Wi-Fi, geralmente para elevadas transferências de dados. O quadro 2.1.4 apresenta um comparativo entre as tecnologias de comunicação sem fio citadas.

Quadro 2.1.4 – Comparativo tecnologias de comunicação para IoT.

Tecnologia	Topologia	Taxa de transmissão	Alcance	Potência de pico
RFID (NFC)	Ponto a ponto	Até 424 Kb/s	Até 20 cm	> 50 mW
Zigbee	Malha, estrela ou árvore	Até 250 Kb/s	Até 100 m	> 100 mW
Z-Wave	Malha	Até 100 Kb/s	Até 40 m	> 100 mW
3G/4G	Árvore	100 Mbit/s (banda por canal = 20 MHz, modulação 64-QAM) [8]	Até dezenas de quilômetros (depende da antena)	Depende da antena
Wi-Fi	Endereço	Até 600 Mbit/s	30 m (típico)	> 600 mW
Bluetooth Classic	Ponto a ponto	Até 3 Mbit/s	100 m (teórico) e 10 m (típico)	< 150 mW
Bluetooth Low Energy	Ponto a ponto	Até 1 Mbit/s	100 m (teórico) e 10 m (típico)	< 75 mW

Fonte: (Solysion).

2.2. O MÓDULO ESP8266

2.2.1. A TECNOLOGIA WI-FI

Hoje em dia o Wi-Fi é uma tecnologia amplamente difundida, estando presente em praticamente todo lugar, como nossas residências, trabalho e lugares de muito movimento, se tornando assim um dos principais meios de acesso à internet. O módulo ESP8266, utilizado nesse projeto utiliza esse padrão para se comunicar com outros dispositivos.

De acordo com Tanenbaum (2003), essa tecnologia teve início com o surgimento dos *notebooks*, pois muitas pessoas queriam seu dispositivo se conectasse automaticamente à internet no momento em que entrassem no escritório. Muitos grupos começaram a trabalhar para alcançar esse objetivo e, a solução encontrada foi equipar os escritórios e os notebooks com transmissores e receptores de rádio de ondas curtas para permitir a comunicação entre eles. Porém, surgiu um problema de compatibilidade pois, um transmissor de uma marca não conseguia se comunicar com um receptor de outra marca então, a indústria decidiu que adotar um padrão era uma boa ideia. Coube ao time do IEEE (*Institute of Eletrical and Eletronics Engineers*), que padronizou as LANs (*Local Area Network)* com fio, elaborar um padrão de LANs sem fio, esse padrão recebeu o nome de 802.11

Em sua primeira versão, apresentada em 1997, a velocidade de comunicação era de 1 Mbps ou 2 Mbps. Como essa era uma velocidade muito lenta, começou-se a trabalhar em busca de padrões mais rápidos. Houve uma divisão dentro do comitê, resultando na publicação de dois novos padrões em 1999. O padrão 802.11a utiliza uma faixa de frequência mais larga e funciona em velocidades de 54 Mbps. O padrão 802.11b utiliza a mesma faixa de frequências que 802.11, mas emprega uma técnica de modulação diferente para alcançar 11 Mbps. O comitê apresentou outra variante, o 802.11g, que utiliza a técnica de modulação do 802.11a, mas emprega a faixa de frequências do 802.11b e atinge 54 Mbps. Padrões mais recentes ainda foram criados pelo IEEE, o 802.11n e o 8002.11ac.

Publicado em 2009, O padrão 802.11n garante um aumento de até seis vezes em relação aos padrões 802.11g e 802.11a, adicionando 40 MHz à camada física e

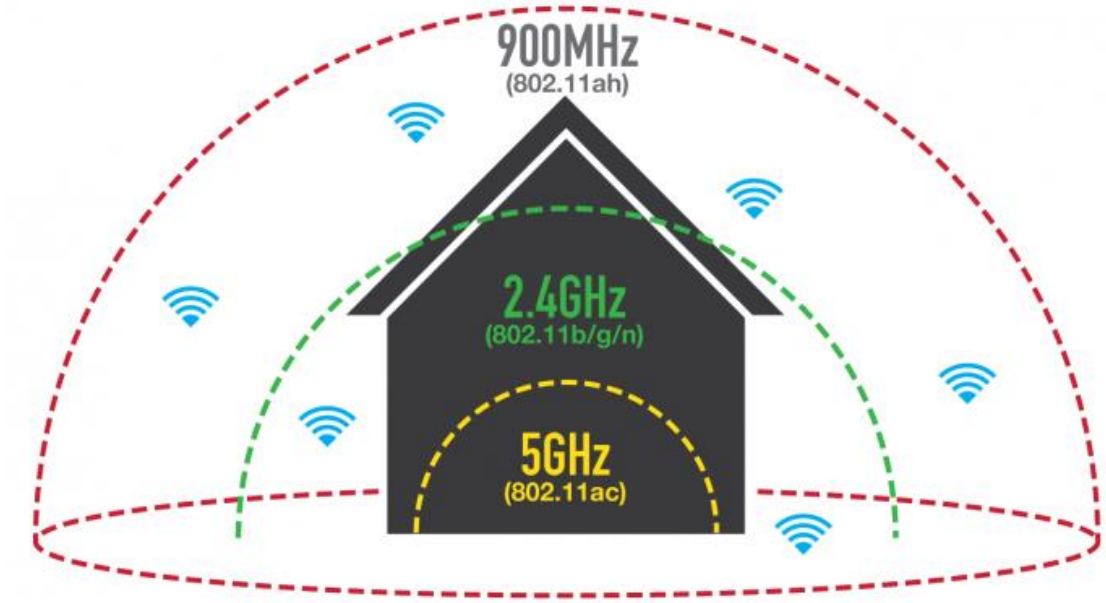
podendo operar nas bandas de 2.4 GHz (802.11b/g) e 5 GHz (802.11ª). Uma das novidades foi a introdução da tecnologia MIMO (*Multiple-Input Multiple-Output*), uma técnica de processamento de sinais para transmitir múltiplos fluxos de dados através de várias antenas. Outra melhoria foi feita no intervalo de Guarda (GI), um pequeno intervalo entre as transmissões cujo objetivo é impedir que as transmissões interfiram umas com as outras, tornando-as imunes a atrasos e eco (Martins, Costa et al, 2013).

Publicada em 2014, a 5ª geração do Wi-Fi, o padrão 802.11ac é uma evolução do 802.11n, fornecendo taxas de até 7 Gbps na banda de 5 GHz, mas de 10 vezes da velocidade que foi previamente padronizada. De acordo com o IEEE:

> “A especificação IEEE 802.11ac adiciona largura de banda de canal de 80 MHz e 160 MHz com canais de 160 MHz contíguos (próximos, vizinhos) e não contíguos para atribuição de canal flexível. Ele adiciona modulação de ordem superior na forma de modulação de amplitude em quadratura de 256 (QAM), proporcionando uma melhoria adicional de 33 por cento na taxa de dados. Uma duplicação adicional da taxa de dados é conseguida aumentando o número máximo de fluxos espaciais para oito.
> A emenda do IEEE 802.11ac introduz uma nova tecnologia revolucionária para suportar múltiplas transmissões de *downlink* simultâneas, referidas como "*multi-input multi-input, multiple-output*" (MU MIMO). Usando a tecnologia de antena inteligente, o MU MIMO permite um uso mais eficiente do espectro, maior capacidade do sistema e latência reduzida, suportando até quatro transmissões simultâneas de usuários[...]” (IEEE, 2014, tradução nossa).

A Wi-Fi Aliance, entidade responsável principalmente pelo licenciamento de produtos baseados na tecnologia, anunciou recentemente o Wi-Fi *HaLow, padrão 802.11ah.* De acordo com Prado (2016), essa tecnologia foi criada especialmente para adaptar as novas aplicações à Internet das Coisas. O Wi-Fi HaLow opera na banda de 900 MHz, o uso dessa banda, além de quase dobrar o alcance da conexão, oferece um Wi-Fi mais robusto, com melhor penetração em paredes e outras barreiras que atrapalham a conexão. Além disso, como a frequência da rede é menor, o padrão consome menos energia. Esse padrão foi projetado para pequenas transferências de informações e não para acessar páginas ou *streaming* de vídeo. A imagem da figura 2.2.1 ilustra uma comparação da frequência de operação e o alcance de cada padrão.

Figura 2.2.1 – Infográfico tecnologia Wi-Fi.



Fonte: (Prado).

A frequência de 900 MHz, utilizada pelo Wi-Fi HaLow, é a frequência utilizada pelo CDMA (*Code Division Multiple Access*, ou Acesso Múltiplo por Divisão de Código), que está caindo em desuso podendo ser, portanto, utilizada por essa nova tecnologia.

2.2.2. APRESENTANDO O MÓDULO ESP8266

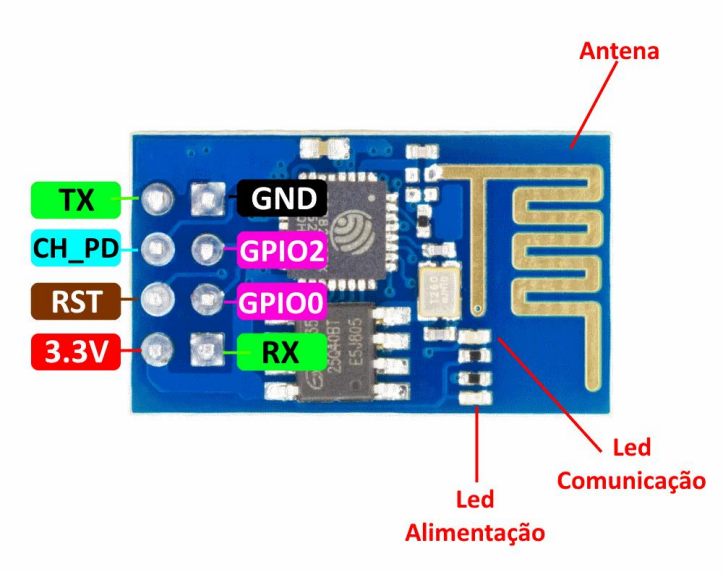
Tendo em vista as principais características das aplicações da Internet das Coisas que são conectividade e mobilidade, o módulo Wi-Fi ESP8266, da empresa Espressif, vem ganhado destaque, devido ao seu tamanho, recursos, facilidade de uso e preço acessível (menos de USD 3,00 em sites internacionais). O ESP8266 é um SOC (*System On Chip*) com protocolo TCP/IP integrado.

Esse módulo pode ser utilizado em uma ampla gama de aplicações, exatamente pelo fato de possuir conectividade Wi-Fi, grande poder de processamento e tamanho reduzido tornando-se assim um dispositivo ideal para desenvolver produtos relacionados à Internet das Coisas. Possíveis aplicações para esse módulo são: Tomadas inteligentes; Automação residencial; Monitoramento remoto; Segurança doméstica, comercial e industrial; Redes de sensores; Controle industrial sem fio;

Monitores de bebês e crianças; Eletrônica vestível; Dispositivos para localização Wi-Fi; Tags de identificação para segurança; Câmeras IP; Robótica; E muito mais.

O ESP8266 foi criado em diversas variantes por sua fabricante, oficialmente são 12, numeradas de ESP-01 até ESP-12, a principal diferença está no que diz respeito ao número de IOs e tamanho do chip. A imagem da figura 2.2.2 apresenta o módulo ESP-01 e após e feita uma descrição dos pinos.

Figura 2.2.2 – Módulo ESP-01



(Fonte: Thomsem).

O quadro a seguir apresenta a descrição dos pinos do módulo da figura 2.2.2.

Quadro 3 – Descrição pinos módulo ESP8266

(Continua)

Vcc	Tensão de alimentação 3.3 V, consome até 300mA;
GND	Sinal de Terra GND;
Tx	Sinal do Tx do módulo a ser conectado no Rx do microcontrolador (sinal em 3.3 V);
Rx	Sinal do Rx do módulo, a ser conectado no Tx do microcontrolador (sinal em 3.3 V!);
RST	Sinal de Reset/Restart acionado em nível baixo (GND);
CH_PD	Sinal de habilitação do chip (chip enable), usado na gravação de firmware ou atualização. Deve ser mantido em nível ALTO para operação normal;
GPIOO	Pode ser controlado pelo firmware, e **deve ser colocado em nível baixo (GND) para modo**

(Continuação)

	***update*, ou em nível alto para operação normal;**
GPIOO2	I/O que pode ser controlada pelo firmware;
LED	Quando está ligado, fica aceso em cor vermelha, e aciona a cor Azul para indicar atividade. Pisca uma vez para indicar momento de boot.

(Fonte: Thomsem).

2.2.3. CARACTERÍSTICAS DE HARDWARE

Esse módulo possui as seguintes características de hardware:

- Wi-Fi integrado em frequência de 2.4 GHz com suporte a WPA e WPA2;
- Conectores GPIO (General Purpose Input Output), barramentos I2C, SPI, UART, entrada ADC, saída PWM e sensor interno de temperatura;
- CPU que opera em 80 MHz, com possibilidade de operar em 160 MHz;
- Arquitetura RISC de 32 bits; 32 KBytes de RAM de instruções;
- 96 KBytes de RAM de dados; 64 Kbytes de ROM para boot;
- Memória Flash SPI Winbond W25Q40BVNIG de 512 KBytes;
- Núcleo baseado no IP (Intellectual Processor) Diamand Standard LX3 da Tensilica, esse núcleo é baseado em um IP Xtensa da Cadence, que foi modificado a critérios da Espressif. Existem módulos de diferentes tamanhos e fabricantes.

2.2.4. CARACTERÍSTICAS DE OPERAÇÃO

- Conexão às redes padrão 802.11 B/G/N, porém com velocidade limitada a velocidade da porta serial;
- Alcance aproximado 91 metros;
- Tensão de operação: 3.3 VDC;
- Comunicação serial: pinos TX e Rx
- Modos de operação: Cliente, *Access Point*, Cliente + *Access Point*;
- Modos de segurança wireless: OPEN/ WEP/WPA_PSK/WPA_WPA2_PSK;

- Suporta comunicação TCP e UDP, com até 5 conexões simultâneas;
- Pode operar em faixas de temperatura de - 40ºC a 125ºC;
- Energia de consumo em modo *sleep* menor que 10 uA;
- Tempo para sair de *sleep* e transmitir pacotes menor que 2ms;
- Potência de *standby* menor que 1.0mW;

2.2.5. O MÓDULO ESP – 12E

Por padrão os módulos são fabricados com o firmware AT, que consiste em uma série de comandos para o uso do módulo em si. Para o desenvolvimento desse trabalho foi utilizado o módulo ESP8266-12E, que além das características já citadas possui alimentação micro USB 5V com regulador de tensão 3.3V e vem com o firmware NodeMCU Lua, esse firmware permite programar o dispositivo em linguagem Lua, transformando-o em um microcontrolador com Wi-Fi integrado. A imagem da figura 2.2.3 mostra o módulo EPSP-12E.

Figura 2.2.3 – Módulo ESP-12E

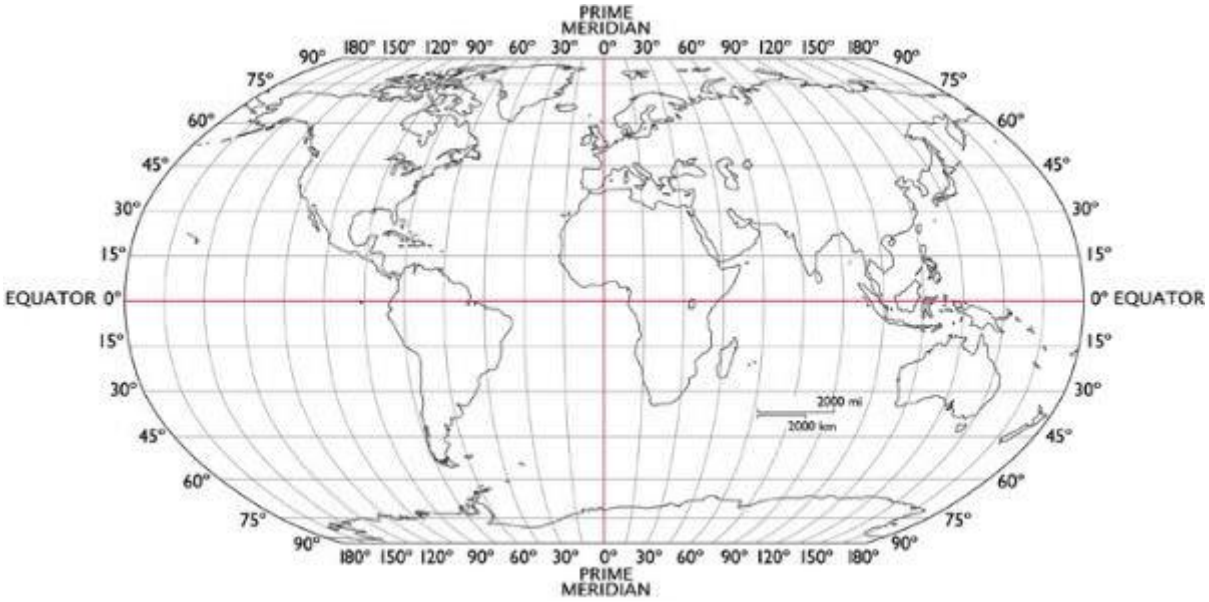
(Fonte: IOT MINILAB).

2.2.6. LOCALIZAÇÃO, INTERNET DAS COISAS E WI-FI

O protótipo desenvolvido nesse projeto se propõe a localizar um objeto, o qual estará de alguma forma com o módulo Wi-Fi “embarcado”. Existem diversas maneiras de se localizar um objeto no espaço, seja por uma posição relativa ou uma posição

absoluta. Uma maneira de se obter uma posição relativa por exemplo, seria através do uso de uma bússola, localizando assim a direção dos pontos cardeais tendo como base o Norte magnético ou através de um endereço que contêm informações de rua, cidade, CEP, estado e país, ou seja, em relação a uma referência. A localização absoluta leva em consideração informações precisas de onde objeto se encontra no espaço. A posição na Terra é referenciada ao Equador e ao meridiano de Greenwich e baseia-se em três denominações: latitude, longitude e altitude. A latitude é o ângulo entre um ponto e a linha do Equador, variado de 0º a 90º para Norte ou Sul. A longitude, também medida em graus, é a distância entre um ponto e o meridiano de Greenwich, podendo ir de 0º a 180º para Leste ou Oeste. A imagem da figura 2.2.4 ilustra os conceitos de latitude e longitude discutidos anteriormente.

Figura 2.2.4 - Latitude e Longitude com referência da Linha do Equador e Meridiano de Greenwich (*Prime Meridian*).



(Fonte: Google Imagens).

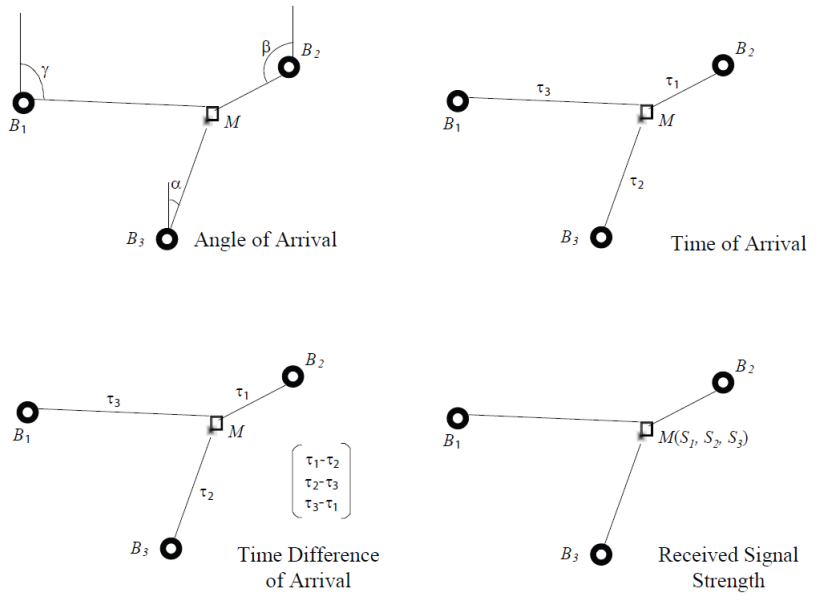
Desta forma, o aplicativo desenvolvido para o sistema *Android*, explora recursos de comunicação e mobilidade presentes hoje nos smartphones, através da Internet das Coisas, para determinar a posição relativa do módulo, e do objeto a ele associado, em relação a posição absoluta do GPS do *smartphone*, localizando assim de forma quase que absoluta o objeto a ser encontrado. Caso o usuário do sistema esteja procurando um objeto, ele deverá estar a uma distância máxima de

aproximadamente 100 metros para que o sinal do módulo seja detectado pelo *smartphone* e a distância seja calculada. Esse valor de alcance é determinado pelo padrão Wi-Fi utilizado pelo módulo.

2.3. TÉCNICAS DE LOCALIZAÇÃO EM RADIOFREQUÊNCIA

As técnicas de localização em radiofrequência podem ser divididas em grupos: Ângulo de Chegada (*Angle of Arrival - AOA*), Indicador de Força do Sinal Recebido (*Received Signal Strength Indicator - RSSI*), Tempo de Chegada (*Time of Arrival - TOA)* e Diferença do Tempo de Chegada (*Time Difference Of Arrival - TDOA*). A imagem da figura 2.3.1 ilustra cada um desses conceitos.

Figura 2.3.1 – Diferentes técnicas de localizar a fonte RF.



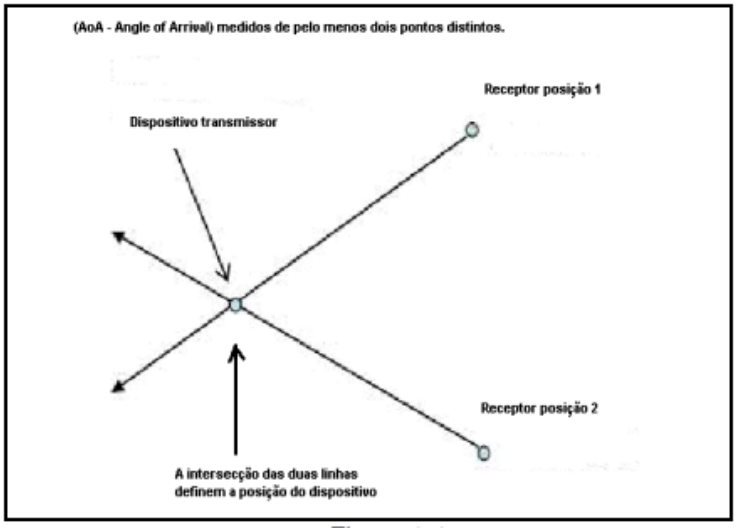
(Fonte: Pereira apud Guoqiang).

2.3.1. ANGLE OF ARRIVAL

A técnica Angle of Arrival (HIGHTOWER; BORRIELO, 2001; RUSSEL, 2003 apud FAGUNDES, 2008), como o nome sugere, utiliza o ângulo de chegada do sinal de radiofrequência, são necessárias duas medições entre o transmissor e o receptor. Em cada medição é calculado o ângulo em que o sinal é mais forte. Com os valores dos ângulos calculados e sabendo o tamanho das arestas adjacentes a ambos, define-se o triângulo. A interseção das linhas determina a localização do dispositivo

transmissor. Essa técnica é boa em ambientes abertos, porém em ambientes fechados o sinal sofre com um problema de multipercurso, o que acaba comprometendo o resultado. Uma representação desta técnica é mostrada na imagem da figura 2.3.2.

Figura 2.3.2 - Representação da técnica AoA.



(Fonte: Fagundes).

2.3.2. TIME OF ARRIVAL

Esta técnica pode ser dividida em duas categorias, uma das quais mede o tempo de propagação num único sentido (do emissor para o receptor) e a outra mede o tempo que um sinal leva para percorrer uma certa distância nos dois sentidos (round-trip time-RTT) (Pereira, 2011).

2.3.2.1. Medição em um único sentido

Este método determina localização do transmissor medindo a diferença entre o instante ao qual o sinal é transmitido e o instante de chegada ao receptor. Conhecida a diferença entre estes dois instantes e a velocidade de propagação no meio, obtém-se na distância entre o emissor e o receptor. Para utilizar essa técnica é necessário

que exista uma sincronização de temporização (clock) entre a estação emissora e receptora, qualquer diferença entre os relógios traduz-se num erro de medição. Esta técnica, por exemplo, é utilizada no GPS.

2.3.2. RTT

Esta técnica mede o intervalo de tempo de ida e volta para que um sinal percorrera uma determinada distância. Ela requer a utilização de dois sensores, um em cada extremo do elo de comunicação. Quando um sensor envia um sinal, ativa ao mesmo tempo um contador que mede o valor do RTT. Dado ao fato de a medição de tempo é local, não há necessidade de sincronismo. A principal fonte de erro tem a ver com o tempo de processamento do sinal no outro nó da comunicação, que é interpretado também como tempo de trânsito. No entanto, este parâmetro pode ser medido à priori, durante a fase de calibração do sistema ou na primeira vez que os sensores se comuniquem entre si.

2.3.3. TIME DIFERENCE OF ARRIVAL

De acordo com Pereira (2011), esta técnica mede a diferença entre os instantes tempo de chegada do sinal em duas antenas no receptor. Se duas antenas i e j estiverem nas posições X_i e Xj respectivamente e o emissor estiver na posição X_t a diferença temporal (Δtij) referente às duas antenas será dada pela seguinte expressão

$$\Delta_{ij} = t_i - t_j = \frac{1}{C} * \left(\|X_t - X_i\| - \|X_t - X_j\| \right) \qquad (2.3.3)$$

Onde ti e tj representam os instantes o tempo em que o sinal chega às antenas i e j, respectivamente, e C representa a velocidade de propagação no meio.

2.3.4. RECEIVED SIGNAL STRENGTH INDICATOR

A técnica de RSSI (Received Signal Strength Indicator) (Reghlin, 2007) é um método que estima a distância entre dois sensores baseado na potência do sinal recebido no sensor receptor. O sensor transmissor envia um sinal com uma

determinada potência que vai se reduzindo à medida que o sinal se propaga. Quanto maior a distância do sensor receptor, menor resulta a potência do sinal ao atingir este sensor.

2.4. O SISTEMA GPS

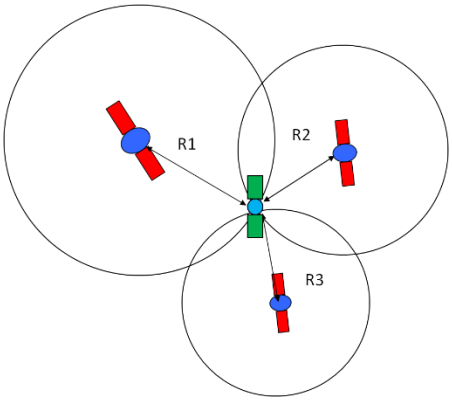
Em termos de sistemas de localização, a tecnologia mais difundida atualmente é o GPS, estando presente em diversas aplicações e áreas em nosso cotidiano como: navegação, agricultura, segurança, localização, etc. O sistema desenvolvido nesse trabalho tem a função de obter a localização relativa do módulo em relação a posição absoluta do módulo GPS do smartphone que detectou o sinal desse módulo.

2.4.1. HISTÓRIA

De acordo com a Administração Nacional da Aeronáutica e Espaço (2012), popularmente conhecido por NASA (*National Aeronautics and Space Administration),* o GPS teve início como um experimento conduzido pela marinha dos EUA na metade dos anos 60, utilizado para localizar submarinos americanos que transportavam mísseis nucleares e, tinha como princípio de funcionamento localização via satélite através de deslocamentos em seu sinal de rádio conhecido como “Efeito Doppler”. Assim, com seis satélites orbitando os polos, submarinos eram capazes de observar as mudanças de satélite em Doppler e marcar a localização de um submarino em questão de minutos. Mais tarde, no início dos anos 70, esse sistema foi adotado e aprimorado pelo DoD (*Departament of Defense)* dos EUA, o DoD decidiu utilizar satélites para dar suporte a seu sistema de navegação proposto. O DoD seguiu em frente e lançou o *Navigation System with Timing and Ranging* (NAVSTAR) em 1978. O sistema com 24 satélites se tornou completamente operacional em 1993.

Atualmente, o sistema GPS é controlado pelo governo norte-americano e é operado pelas Forças Aéreas americanas. Duas camadas de serviço são fornecidas: O *Standard Positioning Service* (SPS), disponível para uso civil e o *Precise Positioning Service* (PPS), de uso restrito das forças armadas norte-americanas, agências federais americanas e forças armadas e governamentais selecionadas. Enquanto o SPS usa o código CA em uma frequência L1, o PPS usa o código P em ambas frequências L1 e L2, garantindo maior precisão. A figura 2.4.1 representa a detecção dos sinais de rádio pelo módulo GPS.

Figura 2.4.1 – Sinais de rádio GPS

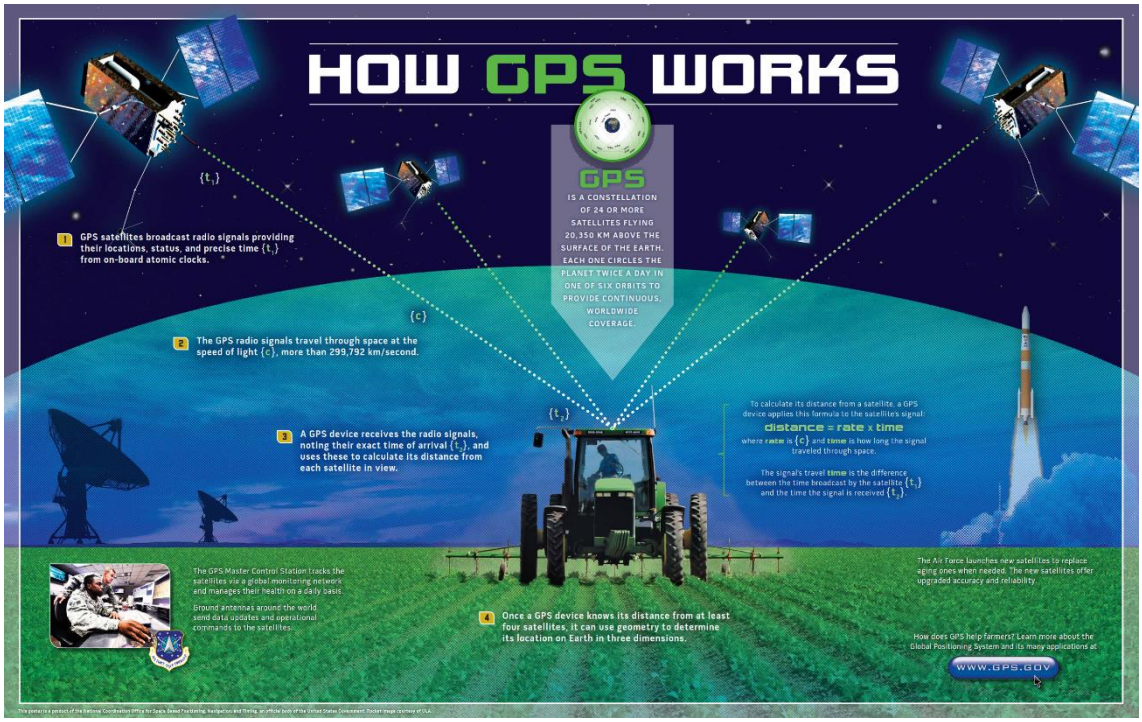


(Fonte: NASA).

2.5. PRINCÍPIOS DE FUNCIONAMENTO

O GPS utiliza a técnica TOA para determinar a posição no espaço. A imagem da figura 2.4.2 apresenta um esquema detalhado de como um dispositivo GPS utiliza essa técnica para determinar sua localização.

Figura 2.4.2 – Como o GPS funciona



(Fonte: GPS.GOV).

Satélites do sistema transmitem sinais de rádio informando, suas localizações, status e tempo preciso obtidos de relógios atômicos onboard. Os sinais de rádio GPS viajam no espaço na velocidade da luz, aproximadamente 3 x 10^8 m/s. O dispositivo GPS recebe os sinais de rádio, salvando seus exatos tempos de chegada e usa esses tempos para calcular a distância de cada satélite em sua linha de visada. Para calclular a distância de um satélite, um dispositivo GPS utiliza a seguinte fórmula:

$$Distância = velocidade * tempo \qquad (6.2)$$

Onde, a velocidade é a velocidade da luz e o tempo é dado pela diferença entre o tempo de transmissão do sinal pelo satélite e o tempo em que o sinal é recebido (TOA). Uma vez que o GPS conhece a distância de pelo menos três satélites, ele utiliza um sistema de triangulação para determinar sua localização na terra em três dimensões. A figura 2.4.3 ilustra a triangulação feita pelo sistema GPS.

Figura 2.4.3 – Triangulação sistema GPS

(Fonte: Martins).

3. METODOLOGIA

3.1. DESCRIÇÃO DO SISTEMA

Atualmente, pesquisas mostram que as pessoas utilizam muito mais o smartphone do que PC para acessar a internet, isso se deve principalmente à mobilidade, permitindo que uma pessoa com smartphone esteja conectada à rede de praticamente qualquer lugar do planeta. Tendo isso em vista, o sistema proposto foi desenvolvido como uma aplicação para smartphones com *Android*, visto que o sistema para dispositivos móveis da Google detém a maior fatia de mercado, o que não impede que no futuro essa solução também possa ser implementada para outros sistemas tais como o iOS, ou permitir que as informações de localização possam ser acessadas em um navegador por meio de um computador. O aplicativo desenvolvido também tira proveito de outros recursos que qualquer smartphone possui hoje em dia, tais como GPS, módulo Wi-Fi, funções como mostrar redes Wi-Fi disponíveis, mandar e receber informações através de redes móveis, etc.

O código implementado para que o módulo opere como Access Point ficou com 891 bytes, ocupando uma pequena parcela dos 64 Kbytes de memória ROM do módulo, o que possibilita adicionar outras operações ao módulo, como leitura de sensores, comunicação para enviar e receber informações, etc.

A função de localização foi implementada tendo como base a técnica de localização RSSI, assim, a distância entre emissor (ESP8266) e receptor (smartphone) é estimada através da potência do sinal recebido.

A distância entre emissor e receptor é calculada por meio da fórmula de Friis, que relaciona potência recebida com potência transmitida e é dada pela equação a seguir:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi R}\right)^2 \qquad (3.1)$$

Onde:

Pr = Potência recebida;

Pt = Potência transmitida;

Gt = Ganho antena transmissor;

Gr = Ganho antena receptor;

λ = Comprimento de onda;

R = distância entre as antenas.

A Pr é dada pela potência do sinal recebido pelo smartphone; Pt é dada pela potência de transmissão do ESP8266 (20 dBm = 100 mW); os ganhos Gt e Gr foram definidos como 1; λ foi obtido para a frequência de operação da Wi-Fi (2.4 GHz) e R é o valor da distância a ser calculado pelo aplicativo.

O diagrama de interação da figura a seguir mostra como funciona a função de localização do módulo.

Figura 3.1.1 – Diagrama de interação entre o ESP8266 e aplicativo.

(Fonte: Elaborada pelo autor).

Não é possível estimar a posição exata do módulo, mas sim a posição do GPS do smartphone e a distância aproximada em que o animal se encontra. Assim, o aplicativo irá mostrar ao dono a marcação de uma circunferência no mapa, com centro nas coordenadas do GPS que identificou o módulo e raio igual a distância estimada.

Logo, caso o dono de um animal de estimação deseje saber em que local seu animal foi avistado pela última vez, basta selecionar a opção no aplicativo que busca a última informação salva no servidor na nuvem.

3.2. DESENVOLVIMENTO DA APLICAÇÃO ANDROID

O aplicativo do sistema de localização foi implementado utilizando o ambiente de desenvolvimento integrado (IDE – *Integrated Development Enviroment*) *Android* Studio, cujo software, documentação e suporte é fornecido pela Google. Dentre os recursos fornecidos pelo *Android* Studio, destacam-se: editor, emulador com vários recursos, compilador, ferramenta para debug e integração com diversas APIs. A imagem a seguir apresenta a interface do usuário do *Android* Studio.

Foram utilizadas duas APIs no desenvolvimento do aplicativo: a do *Google Maps* e a do *Firebase*. É necessário realizar uma série de passos para integrá-las ao *Android* Studio (ambiente de desenvolvimento de aplicações *Android*) e utilizá-las na aplicação.

3.2.1. API DO GOOGLE MAPS

O *Google Maps* é um serviço de visualização de mapas e imagens de satélite fornecido pela Google. A API para *Android* permite adicionar mapas baseados em dados da Google à aplicação.

As APIs do *Google Maps* estão disponíveis para *Android*, iOS, navegadores e via serviços web HTTP. As APIs são gratuitas para uma variedade de casos de uso, possuindo um plano padrão gratuito para aplicações externas gratuitas publicamente disponíveis para dispositivos móveis com cobranças por aumentos acima dos limites de uso e, o plano empresarial com contratos anuais para implementações empresariais e tem como vantagens suporte técnico 24 horas; Acordo de Nível de Serviço (ANS) e recursos avançados de implementação da API e sem publicidade.

A API do *Google Maps* fornece os seguintes recursos para adicionar à aplicação: mapas interativos; visualização de imagens de satélite e do *Street View* e, a adição de marcadores, janelas de informação e polilinhas personalizadas. A imagem da figura 3.2.1 apresenta os principais recursos disponíveis através da API do Google Maps.

Figura 3.2.1 – Recursos disponíveis na API do *Google Maps*

(Fonte: Documentação API Google Maps).

Para adicionar o Google *Maps* a aplicação, é necessário obter uma chave de acesso, para acessar os servidores do *Google Maps*, os procedimentos para obter uma chave de acesso e adicioná-la ao código no Androis Studio estão disponíveis na documentação do Google Maps.

3.2.2. API DO FIREBASE

O *Firebase* fornece um conjunto de ferramentas para desenvolver aplicações de alta qualidade, de forma rápida e fácil.

A maioria dos recursos do *Firebase* é gratuito em qualquer escala, todos os recursos pagos têm uma camada gratuita com dois planos pagos para quando o aplicativo começar a expandir. A medida que a popularidade da aplicação for crescendo não é necessário se preocupar em expandir o código do servidor ou provisionar mais capacidade, esse serviço é fornecido pelo *Firebase*.

A imagem a seguir mostra todos os serviços fornecidos pelo *Firebase*, cada recurso funciona de forma independente, cabe ao desenvolvedor decidir quais recursos integrar ou não à aplicação.

Figura 3.2.2 – Recursos API Firebase

(Fonte: Documentação API Firebase).

Dentre os recursos mostrados na figura 3.2.2, no aplicativo desenvolvido foi utilizado o Realtime Database, que é um banco de dados hospedado na nuvem. Os dados transmitidos são armazenados em JSON (*JavaScript Object Notation*) e sincronizados em tempo real com todos os clientes conectados e permanecem disponíveis quando o aplicativo é desconectado. Assim, todos os clientes compartilham uma instância de Realtime Database e automaticamente recebem atualizações com os dados mais recentes.

O banco de dados no servidor do *Firebase* tem a estrutura de uma árvore JSON, hospedada na nuvem. Diferentemente de um banco de dados SQL, não há tabelas nem registros. Quando dados são adicionados à árvore JSON eles se tornam um nó na estrutura JSON existente.

Outros recursos que podem ser adicionados futuramente na aplicação através dessa API são o *Firebase* Authentication, para autenticação do usuário, o qual possibilita criar uma tela de *login* utilizando serviços como uma conta Google ou Facebook; Realtime Messaging que possibilita a troca de mensagens entre usuários da aplicação; *Storage* para armazenamento de arquivos como imagens e, o *AdWords* para obter ganhos extras através de publicidades no aplicativo.

3.3. MODELAGEM

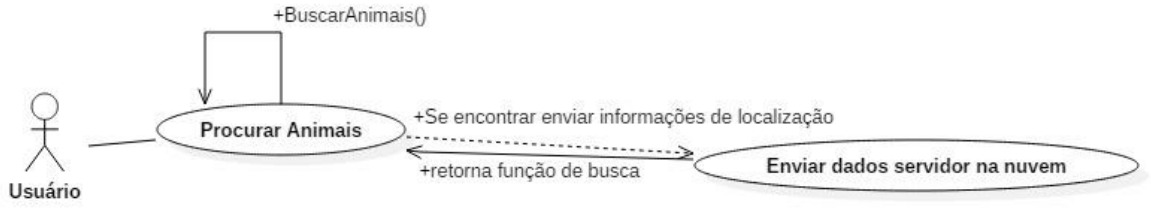
As principais funcionalidades do aplicativo desenvolvido são demonstradas através de diagramas utilizando a linguagem UML (*Unified Modeling Language*). A UML é uma linguagem gráfica para visualização, construção e documentação de artefatos de sistemas complexos de software (Booch, 2006), geralmente utilizando orientação a objetos.

3.3.1. DIAGRAMA DE CASO DE USO

Os casos de uso descrevem a funcionalidade do sistema percebida por atores externos. Um ator interage com o sistema podendo ser um usuário, dispositivo ou outro sistema (Furlan, 1998).

No protótipo do aplicativo do sistema de localização para encontrar animais, através da opção de busca, o usuário pode procurar os animais que estejam por perto e o sistema determina as informações de localização e envia para o servidor na nuvem. Nos diagramas a seguir, são feitas as descrições do caso de uso, explicando o fluxo principal da aplicação para um melhor entendimento da ordem dos eventos.

Figura 3.1.1 – Diagrama caso de uso função de busca



(Fonte: Elaborada pelo autor).

No quadro 3.1.1 é feita a descrição do caso de uso da imagem da figura 3.1.1.

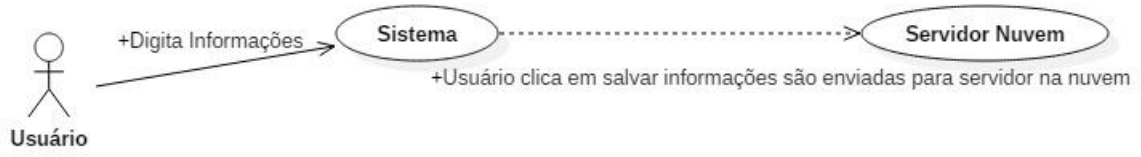
Quadro 3.1.1 – Descrição do caso de uso função de busca.

Descrição	Este caso de uso permite que o usuário procure animais. Ao encontrar, o aplicativo envia as informações de localização determinadas pelo aplicativo para o servidor na nuvem.
Caso de Uso	Buscar Animais
Ator	Usuário
Pré-condição	Usuário selecionar função de busca na tela principal da aplicação.
Pós-condição	Após enviar informações para o servidor, a aplicação retorna para a função de busca até que e continua buscando até que a aplicação seja encerrada.
Fluxo Principal	1. Usuário acessa seleciona a opção de buscar animais na tela inicial do aplicativo; 2. Aplicação busca por animais e, caso entre, envia informações de localização para o servidor.
Fluxo Alternativo	As informações de localização não são enviadas ao servidor pois o smartphone não possui acesso à internet.
Caso de Uso Complementar	Nenhum.

(Fonte: Elaborada pelo autor).

O diagrama da figura 3.1.2 apresenta o caso de uso da função de cadastro do usuário.

Figura 3.1.2 – Diagrama caso de uso função de busca



(Fonte: Elaborada pelo autor).

No quadro 3.1.2 é feita a descrição do caso de uso do diagrama da figura 3.1.2.

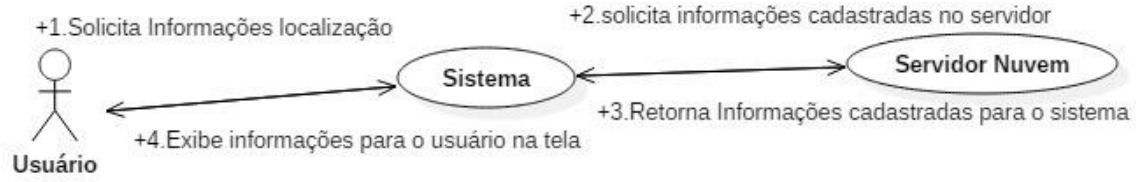
Quadro 3.1.2 – Descrição do caso de uso função de cadastro.

Descrição	Este caso de uso permite que o usuário cadastre suas informações no sistema através do aplicativo. Essas informações serão salvas no banco de dados do servidor na nuvem.
Caso de Uso	Cadastro de informações no sistema
Ator	Usuário
Pré-condição	Usuário selecionar função de novo cadastro no aplicativo.
Pós-condição	Após enviar fornecer as informações ao sistema, esses dados são transmitidos e salvos no servidor na nuvem.
Fluxo Principal	1. Usuário acessa seleciona a opção de realizar cadastro no aplicativo; 2. Usuário preenche os campos solicitados e seleciona a opção salvar; 3. Aplicação busca envia informações fornecidas para o banco de dados do servidor na nuvem.
Fluxo Alternativo	As informações fornecidas pelo usuário não são enviadas ao servidor pois o smartphone não possui acesso à internet.
Caso de Uso Complementar	Editar cadastro.

(Fonte: Elaborada pelo autor).

O diagrama da figura 3.1.3 apresenta o caso de uso da função de busca das informações cadastradas no servidor na nuvem.

Figura 3.1.3 – Diagrama caso de uso busca informações no servidor na nuvem



(Fonte: Elaborada pelo autor).

No quadro 3.1.3 é feita a descrição do caso de uso do diagrama da figura 3.1.3.

Quadro 3.1.3 – Descrição do caso de uso função de cadastro.

(Continua)

Descrição	Este caso de uso permite que o usuário visualize no aplicativo as informações salvas no servidor na nuvem.

(Continuação)

Caso de Uso	Busca de informações salvas no banco de dados do servidor na nuvem.
Ator	Usuário
Pré-condição	Usuário selecionar função para mostrar as informações de localização no aplicativo.
Pós-condição	Informações salvas no servidor são exibidas na tela do aplicativo.
Fluxo Principal	1. Usuário solicita no aplicativo as informações de localização salvas no servidor na nuvem; 2. O aplicativo solicita leitura dos dados salvos no servidor; 3. Servidor envia ao aplicativo as informações solicitadas; Aplicação exibe ao usuário as informações solicitadas.
Fluxo Alternativo	As informações solicitadas pelo usuário não podem ser exibidas pois o smartphone não possui acesso à internet.
Caso de Uso Complementar	Exibir localização em uma mapa do Google Maps.

(Fonte: Elaborada pelo autor).

3.3.2. DIAGRAMA DE CLASSES

Um diagrama de classes apresenta as classes utilizadas para o desenvolvimento do sistema, com seus atributos e métodos e a relação entre as classes. A imagem da figura mostra a classe utilizada no desenvolvimento do protótipo da aplicação.

Figura 3.3.2 – Classe Coleira.

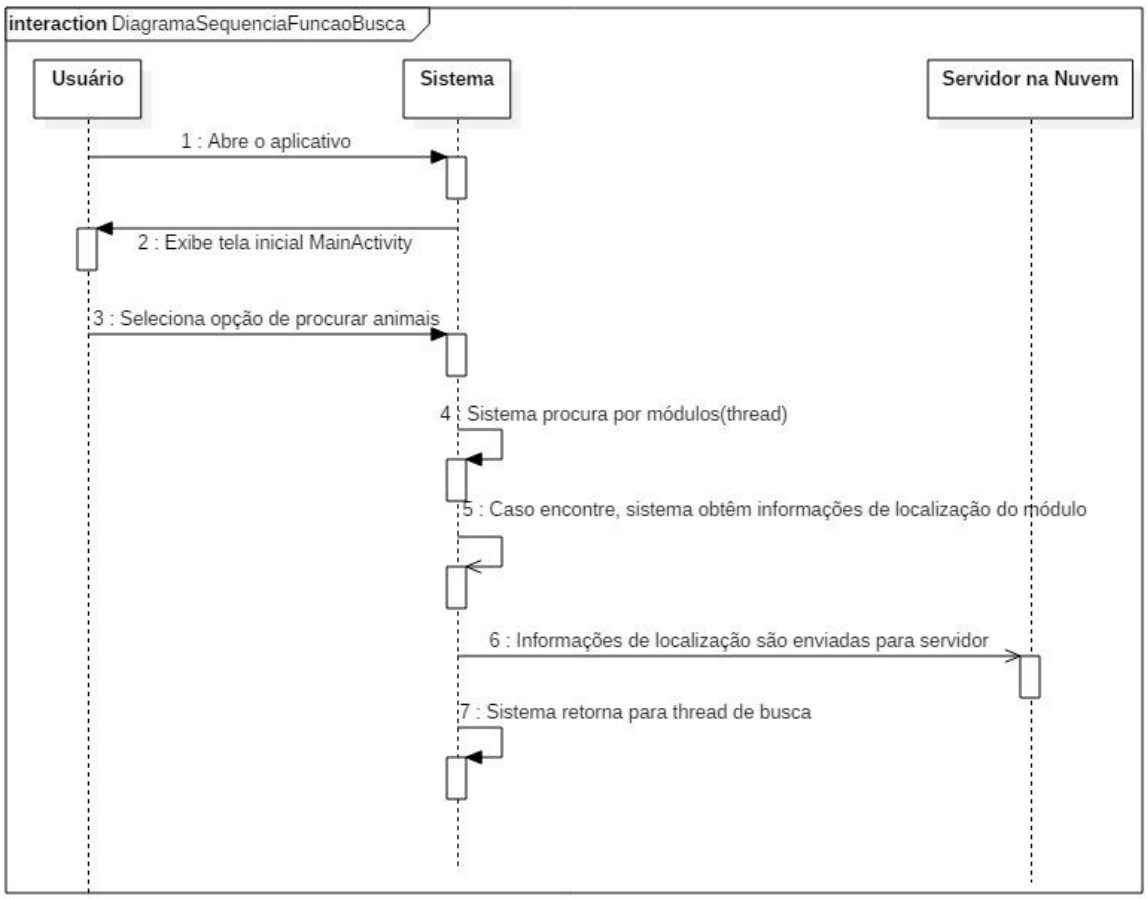
(Fonte: Elaborada pelo autor).

3.3.3 DIAGRAMA DE SEQUÊNCIA

Um diagrama de sequência expõe o aspecto do modelo que enfatiza o comportamento dos objetos em um sistema, incluindo suas operações, interações, colaborações e histórias de estado em sequência temporal de mensagens e representação explícita de ativação de operações.

O diagrama de sequência da figura 3.3.3 ilustra como o usuário utiliza o aplicativo para buscar os módulos que estão próximos ao *smartphone*.

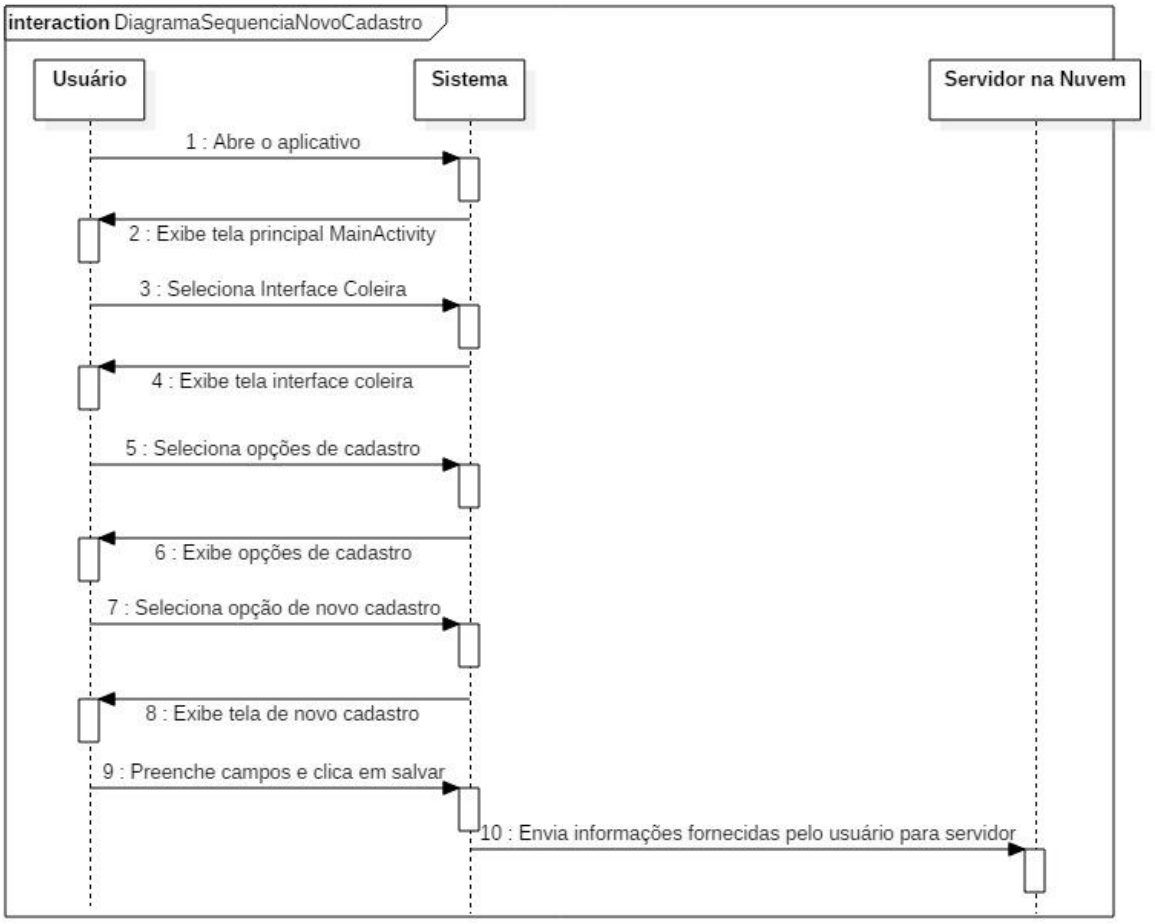
Figura 3.3.3 – Diagrama sequência função de busca.



(Fonte: Elaborada pelo autor).

O diagrama de sequência da figura 3.3.4 mostra como o usuário utiliza o aplicativo para cadastrar suas informações, do animal e do módulo e quando essas informações são enviadas para o servidor na nuvem.

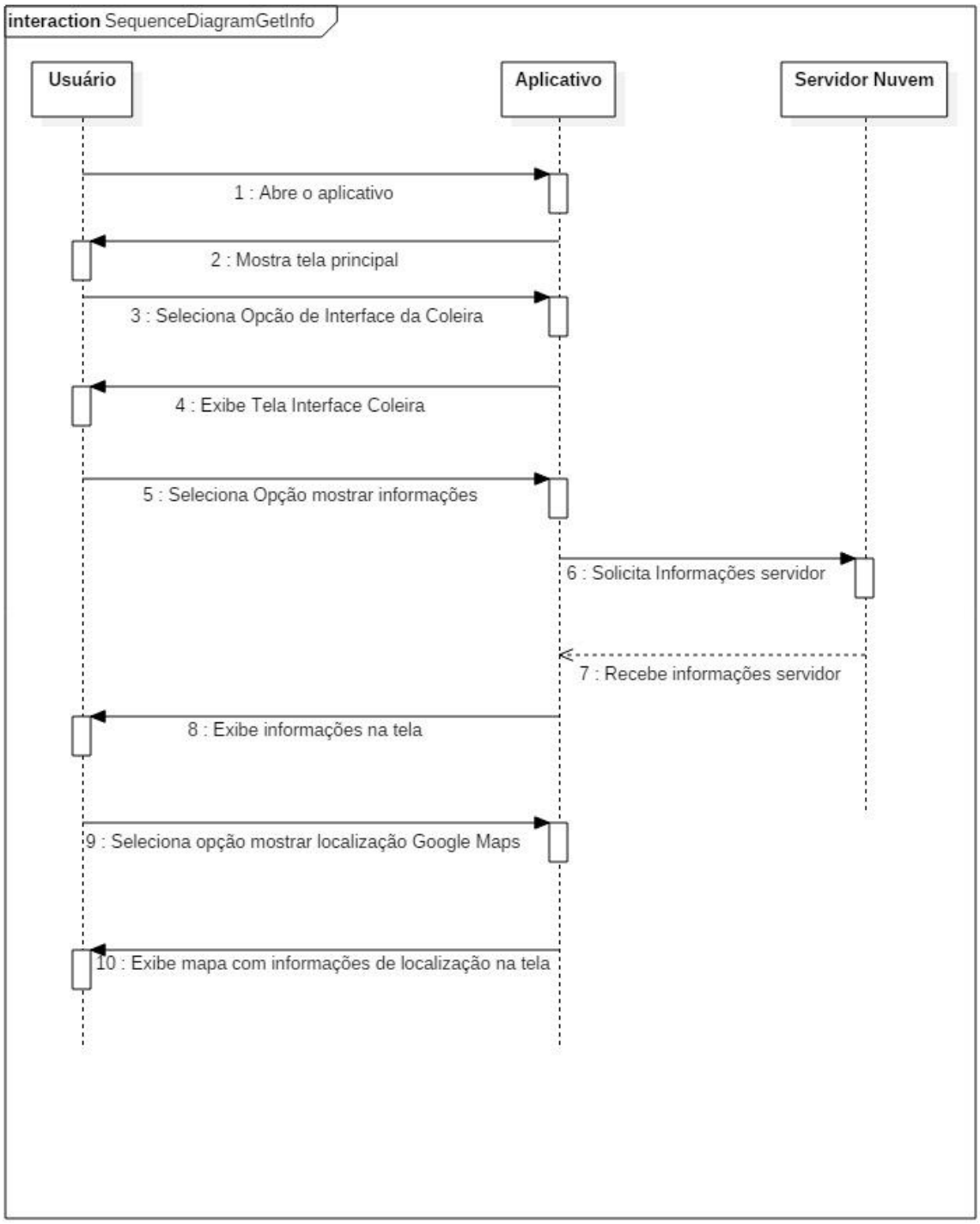
Figura 3.3.4 – Diagrama sequência cadastrar informações no sistema.



(Fonte: Elaborada pelo autor).

O diagrama de sequência da figura 3.3.5 mostra como o usuário utiliza a aplicação para visualizar as informações de localização do animal cadastrado.

Figura 3.3.5 – Diagrama sequência mostrar informações de localização da coleira.

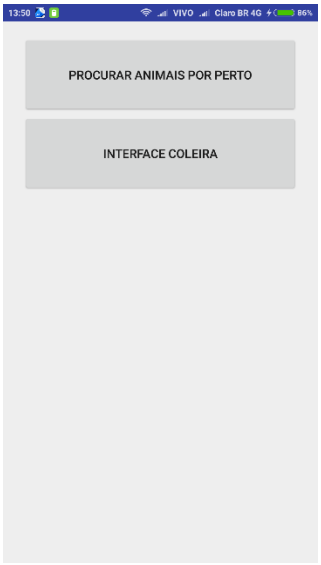


(Fonte: Elaborada pelo autor).

3.4. TELAS DO APLICATIVO

O aplicativo foi implementado sob duas perspectivas: a do cliente, dono de um animal, que adquiriu a coleira; e a do usuário que irá utilizar o aplicativo para localizar os animais. A imagem da figura 3.4.1 apresenta a tela inicial do aplicativo, onde é possível escolher entre a função de busca e a interface do cliente.

Figura 3.4.1 – Tela inicial do aplicativo.



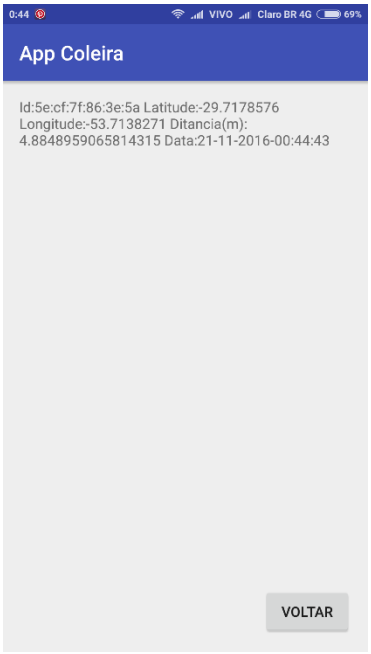
(Fonte: *print screens* do aplicativo no sistema Android).

A seguir, é feita uma descrição mais detalhada de cada caso de uso, com diagramas de interação e imagens das telas do aplicativo.

3.4.1. USUÁRIO

O usuário comum do aplicativo, que também pode ser um cliente, realiza a operação de busca dos animais (módulos) que estão por perto. A imagem da figura 3.4.2 apresenta um *print* da tela de busca do aplicativo.

Figura 3.4.2 – Tela função de busca.

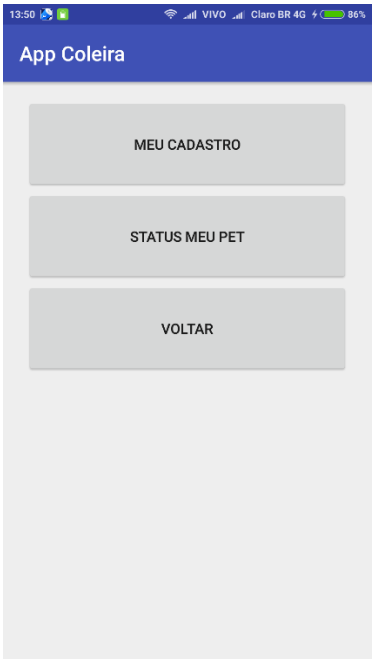


(Fonte: *print screens* do aplicativo no sistema Android).

3.4.2. CLIENTE

Ao entrar na interface do cliente, são exibidas opções de cadastro e de visualização de status do pet. A figura 3.4.3 apresenta a tela de interface do cliente.

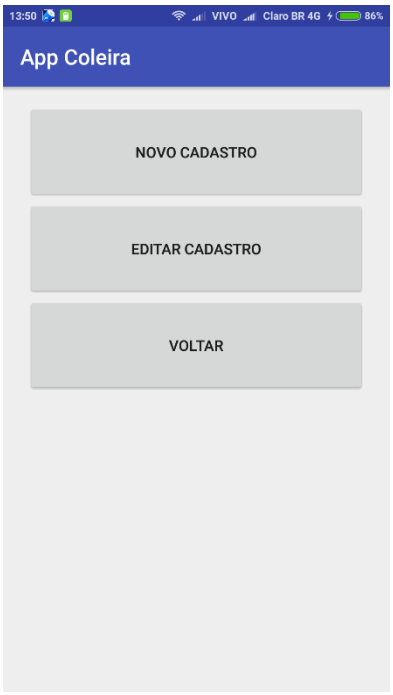
Figura 3.4.3 – Tela interface cliente.



(Fonte: *print screens* do aplicativo no sistema Android).

Ao selecionar a opção Meu Cadastro, o aplicativo seguirá para a uma tela onde é possível criar um novo cadastro, ou editar o cadastro feito anteriormente. A figura 3.4.4 mostra a tela com as opções de cadastro do cliente no aplicativo.

Figura 3.4.4 – Tela interface cadastro cliente.



(Fonte: *print screens* do aplicativo no sistema Android).

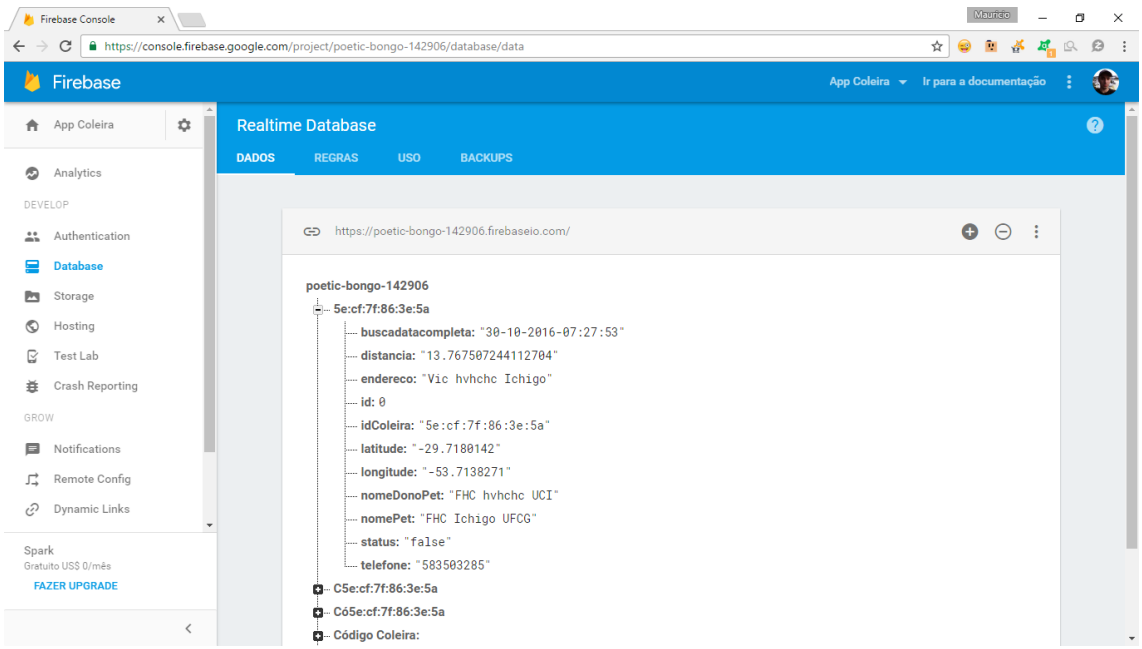
Para utilizar o sistema, primeiro é necessário que o cliente realize um cadastro. Após preencher os campos e clicar em salvar as informações são armazenadas no banco de dados do *Firebase Database*. Foi criado também um banco de dados local, utilizando o recurso do *Android SQLite*. Nesse banco de dados local é armazenado o código da coleira cadastrado pelo usuário para que essa informação esteja disponível quando a aplicação for fazer a leitura das informações no servidor na nuvem, onde cada registro é único e identificado por esse código. A figura 3.4.5 apresenta a tela de cadastro do cliente.

Figura 3.4.5 – Tela novo cadastro

(Fonte: *print screens* do aplicativo no sistema Android).

Na imagem da figura 3.4.6 podemos ver os dados armazenados no servidor *Firebase Database,* na nuvem, na estrutura JSON.

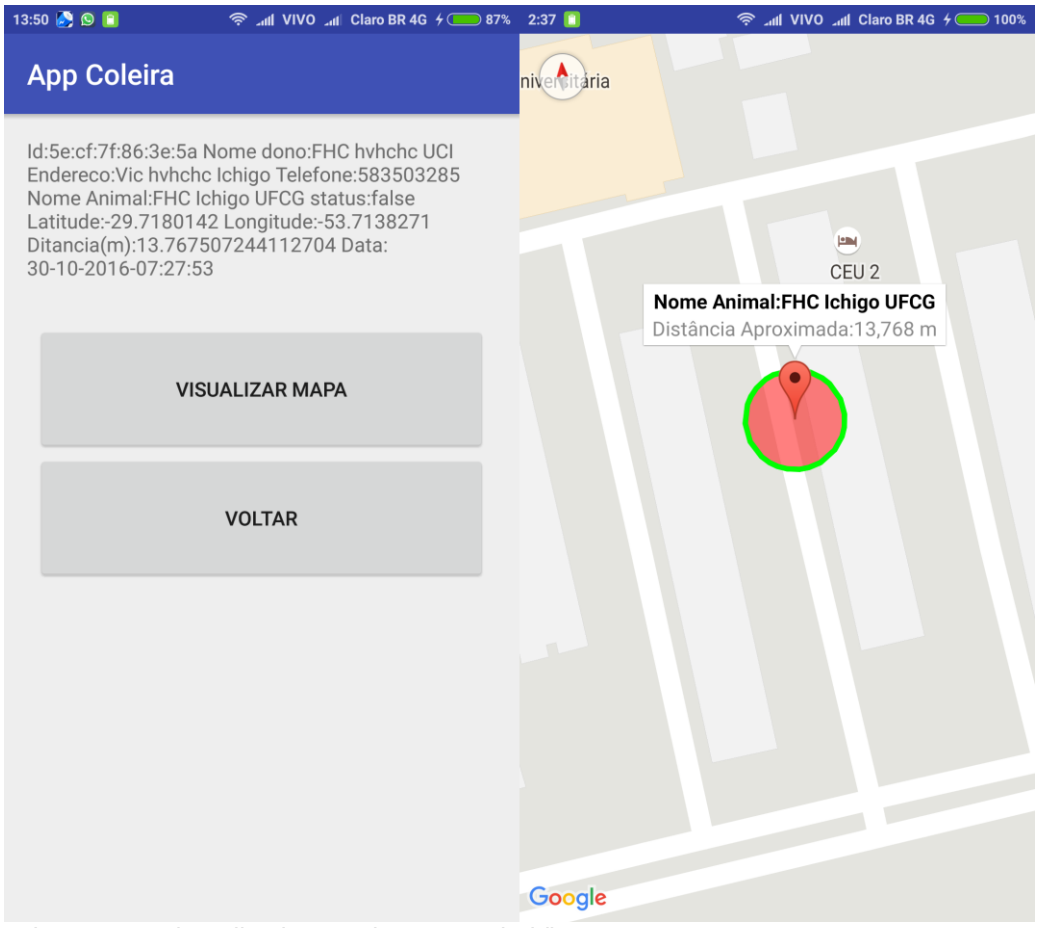
Figura 3.4.6 – Console Firebase Database.



(Fonte: *print screen* do console do Realtime Database no navegador Google Chrome no sistema Windows).

Após realizar o cadastro, o cliente pode visualizar as informações cadastradas e a localização do módulo através da opção de status. A imagem da Figura mostra a tela com as informações do módulo cadastrado obtidas do *Firebase Database* e a visualização no mapa do *Google Maps* da localização do módulo (dentro da área da circunferência, em vermelho). A figura 3.4.7 mostra a tela com as informações de localização buscadas do servidor do *Firebase Database* e a localização no mapa, através do *Google Maps*.

Figura 3.4.7 – Tela informações de localização com visualização no mapa.



(Fonte: *print screens* do aplicativo no sistema Android).

4. RESULTADOS OBTIDOS

4.1. TESTE DE ALCANCE

Foi realizado um teste de alcance para verificar se a distância R entre o módulo e o celular, calculada pelo aplicativo através da equação de Friis (equação 3.1), corresponde ao valor real. As medições foram feitas em um local com poucas redes Wi-Fi, com o módulo no chão e, sem nenhum obstáculo entre o módulo e o celular utilizado para que assim não houvesse nenhuma interferência que pudesse alterar os valores estimados pelo aplicativo. A tabela 4.1 apresenta os valores medidos para cada distância, com o valor médio, variância e desvio padrão calculados.

Tabela 4.1 – Valores medidos experimentalmente.

Dist/Medida	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	$\overline{X}$	*Variâncias (S^2)*	*Desvio Padrão (S)*
1	13,8	13,8	15,4	13,8	13,8	13,8	21,8	19,4	15,4	12,3	15,3	17,2	3,5
2	30,8	21,8	30,8	27,5	27,5	27,5	27,5	34,6	34,6	27,5	29	32	5,2
5	109,4	154,5	194,5	86,9	86,9	77,4	69	86,9	86,9	86,9	103,9	114,8	9,3
10	308,2	488,5	308,2	244,8	345,8	345,8	345,8	308,2	345,8	345,8	338,7	379,7	18,6
15	690	218,2	218,2	194,5	194,5	388	488,5	548,1	615	488,5	404,3	417,5	22,1
30	615	690	774,2	690	774,2	690	774,2	615	615	488,5	672,6	753,7	22,1
50	1227	774,2	1127	1227	974,7	974,7	1227	1544,7	1227	1944,7	1224,8	1360,7	44,1
60	1376,8	1733,2	1733,2	1733,2	1733,2	1544,7	1093,6	1544,7	1376,8	1733,2	1560,3	1754	41,6
80	2448,2	2182	3082,2	3082,2	2747	2747	1733,2	2448,2	2448,2	1944,7	2486,3	2766,8	44,1
100	3082,2	3082,2	2747	2747	3082,2	3082,2	3082,2	3082,2	4393,7	3082,2	3146,3	3503	55,5

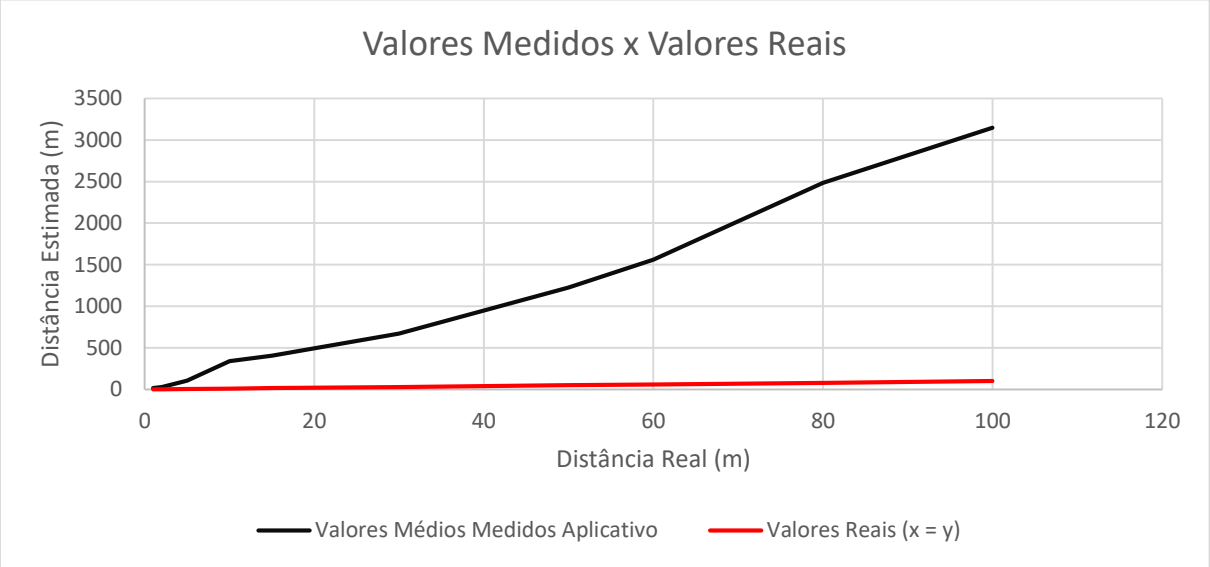
(Fonte: Elaborada pelo autor).

A primeira coluna da tabela 4.1 apresenta as distâncias reais em que as distâncias forma estimadas pelo aplicativo, as colunas numeradas de 1 a 10 apresentam o número de medições feitas. As três últimas colunas apresentam a média a variância e o desvio padrão dos valores de cada medida. Como se pode observar na tabela, à medida que a distância aumenta os valores tendem a variar e

se desviar mais dos valores reais, o que aumenta o erro e diminui a precisão do sistema.

O gráfico da figura 4.1.1 apresenta os valores médios calculados pelo aplicativo em comparação aos valores reais esperados.

Figura 4.1.1 – Valores Medidos X Valores Reais.



(Fonte: Elaborada pelo autor).

Como pode-se notar houve uma grande diferença entre a distância estimada pelo aplicativo e o valor real esperado. Como os valores encontrados são praticamente lineares, podem ser aproximados por uma reta, assim, é possível calcular um fator de calibração para a equação para que os valores fiquem mais próximos do esperado. O fator de calibração é dado pelo coeficiente angular da reta aproximada pelos valores medidos experimentalmente. A equação da reta é dada por:

$$y = ax + b \qquad (4.1)$$

O coeficiente angular m de uma reta é calculado através da equação:

$$a = m = \tan\alpha = \frac{\Delta y}{\Delta x} \qquad (4.2)$$

Linearizando a reta dos valores medidos experimentalmente pelos pontos inicial e final, (1, 15,327) e (100, 3146,626), calcula-se o coeficiente angular da reta que passa por esses pontos, que também é o fator de calibração o qual deve ser aplicado à equação de Friis para que o valor da distância calculada pelo aplicativo seja aproximadamente igual ao valor real.

$$m = \frac{3146{,}275 - 15{,}327}{100 - 1} = 31{,}626 \qquad (4.3)$$

$$y - y_0 = m\,(x - x_0) \qquad (4.4)$$

Calculando a função da reta aproximada pelos valores medidos através da equação, no ponto (1, 15,327):

$$y = 31{,}626x - 16{,}299 \qquad (4.5)$$

Dividindo a equação pelo fator de calibração m = 31,626:

$$y = x - 0{,}515 \qquad (4.6)$$

Assim, para que o valor calculado pelo aplicativo seja aproximadamente igual ao da reta y = x deve-se fazer o seguinte ajuste a equação de Friis: multiplicar o valor calculado por 1/(31,626) e, somar 0,515 ao valor final. A tabela 4.2 apresenta o valor médio medido pelo aplicativo e o valor médio ajustado, pode-se perceber que os valores ficaram muito próximos dos valores reais.

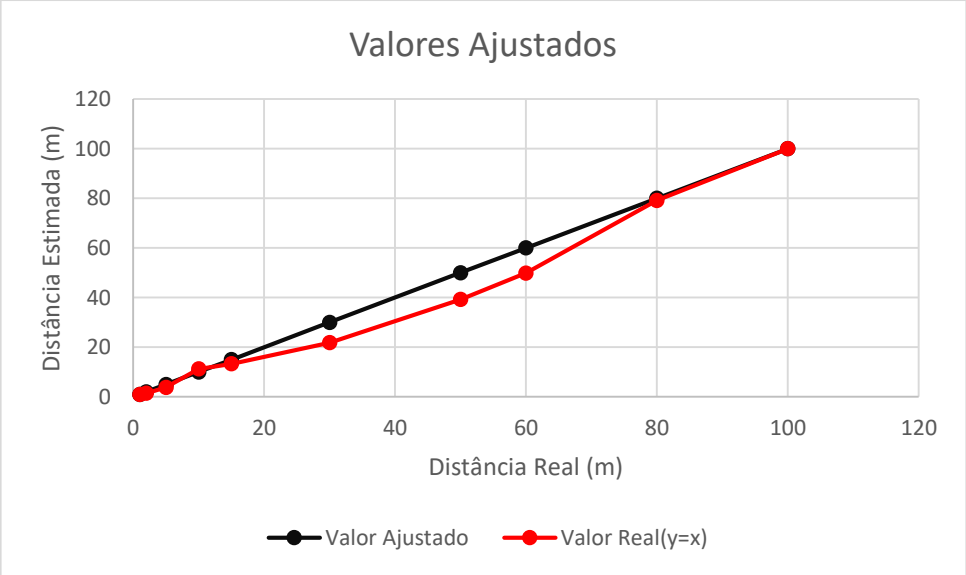
Tabela 4.2 – Valores médios medidos e ajustados.

Distância (m)	*Valor medido médio (m)*	*Valor médio ajustado*
1	15,3	1,0
2	29,0	1,4
5	103,9	3,8
10	338,7	11,2
15	404,3	13,3
30	672,6	21,8
50	1224,8	39,2
60	1560,3	49,9
80	2486,3	79,1
100	3146,3	100,0

(Fonte: Elaborada pelo autor).

O gráfico da figura 4.1.2 mostra os dados médios após serem calibrados em comparação aos valores esperados.

Figura 4.1.2 – Valores após calibração calculados.



(Fonte: Elaborada pelo autor).

Os valores de ajuste calculados nas equações (4.5) e (4.6) foram aplicados à função de cálculo da distância no código fonte do aplicativo e novas medidas foram feitas para verificar na prática a melhora da precisão do sistema. As distâncias estimadas pelo aplicativo a partir dessa atualização são apresentadas na tabela 4.3.

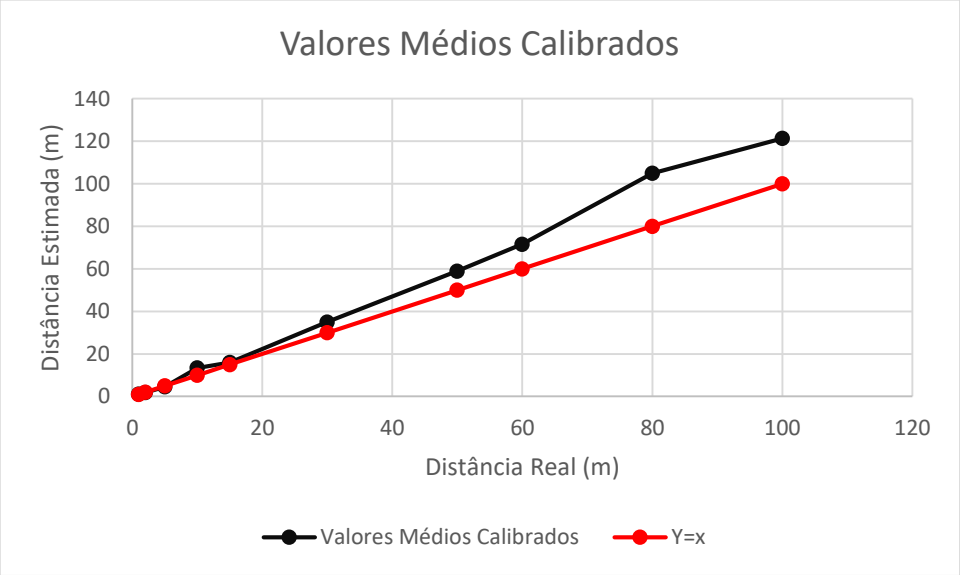
Tabela 4.3 – Valores medidos pelo aplicativo após ajuste da equação.

Distância/ Medida	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	$\overline{X}$	*Variâncias (S^2)*	*Desvio Padrão (S)*
1	1,1	1,0	2,1	1,1	0,9	1,1	0,9	0,9	0,9	1,9	1,2	1,3	1,4
2	1,6	1,6	1,5	2,1	1,4	1,6	3,0	1,7	1,5	2,7	1,9	2,1	1,6
5	4,9	4,0	5,4	4,9	3,3	4,9	3,3	3,9	5,4	5,4	4,5	5,0	2,3
10	16,0	12,9	14,3	14,3	14,3	12,8	12,8	10,3	12,8	14,3	13,5	14,7	3,8
15	17,8	16,0	14,3	15,3	14,3	14,3	16,0	17,8	16,0	17,8	16,0	17,5	4,2
30	31,3	44,0	28,0	39,3	35,1	39,3	39,3	31,3	31,3	31,3	35,0	39,3	5,6
50	49,4	55,3	69,5	62,0	44,0	62,0	49,4	55,3	55,3	87,4	59,0	66,6	9,3
60	87,4	87,4	77,9	77,9	49,4	69,5	62,0	62,0	55,3	87,4	71,6	77,8	9,3
80	*98,0*	*138,2*	*87,4*	*109,9*	*109,9*	*123,2*	*109,9*	*98,0*	*87,4*	*87,4*	*104,9*	*117,3*	*9,3*
100	98,0	109,9	109,9	138,2	138,2	123,2	138,2	138,2	109,9	109,9	121,3	137,4	10,5

(Fonte: Elaborada pelo autor).

O gráfico da figura 4.1.3 mostra a média dos valores obtidos, a partir das novas medições, em comparação aos valores esperados, percebe-se que houve uma melhora nas distâncias estimadas, porém, há uma diminuição da precisão a medida que a distância aumenta.

Figura 4.1.3 – Valores médios medidos após ajuste da equação.



(Fonte: Elaborada pelo autor).

Alguns fatores devem ser levados em consideração para que haja uma melhora na precisão. Como as medidas foram feitas em um único celular, os ganhos das antenas do celular podem variar de um dispositivo para outro, esses valores foram definidos como 1 na equação que calcula da distância. Uma solução para esse problema seria fazer uma calibração da potência recebida em relação a distância em tempo de execução de *software*.

4.2. TESTE DE BATERIA

Um teste de bateria foi realizado para se ter uma estimativa da energia consumida pelo módulo ESP8266. Para realizar este teste foi utilizado uma célula de bateria semelhante à da figura 4.2.1, a qual possui uma capacidade de 2600 mAH.

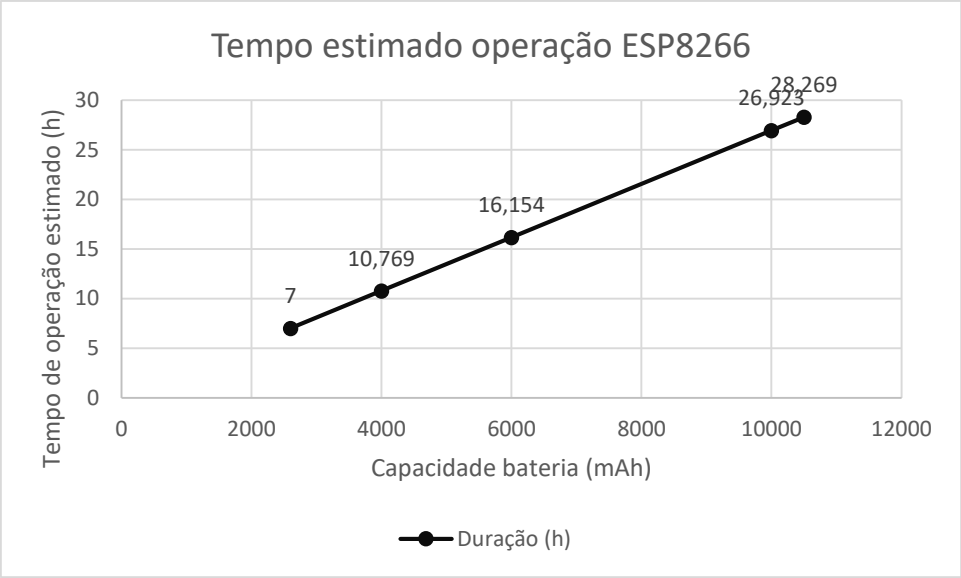
Figura 4.2.1 – Lipstick Power Bank

(Fonte: Google Imagens).

O teste foi feito com a bateria completamente carregada assim, ela foi conectada a alimentação do módulo ESP8266, configurado com *Access Point*, e foi medido o tempo de operação que a carga acabasse. O tempo para que a bateria descarregar foi de 7 horas, o que se mostrou um tempo razoável, porém o ideal seria que o módulo operasse por um tempo maior. Uma alternativa para aumentar esse tempo de operação é programar o módulo para operar em modo de baixo consumo

de energia ou programar uma função *sleep*, a qual permite que o módulo fique ativo por um período de tempo e “dormindo” até que seja acordado novamente de forma que esse ciclo sempre se repita. O gráfico da figura 4.2.2 apresenta o tempo estimado de operação do módulo para diferentes capacidades de baterias vendidas hoje em dia, o único problema é o tamanho dessas baterias para certas aplicações, quanto maior a capacidade maior a bateria.

Figura 4.2.2 – Estimativa de tempo de operação módulo ESP8266



(Fonte: Elaborada pelo autor)

5. CONCLUSÕES

Este trabalho apresenta o desenvolvimento do protótipo de um sistema de localização que explora os conceitos de conectividade e mobilidade proposto pela Internet das Coisas.

A metodologia utilizada foi implementar uma aplicação *Android* capaz de detectar o módulo através do SSID da rede Wi-Fi. O aplicativo calcula ainda a distância relativa do módulo em relação a posição absoluta do GPS do *smartphone* com base na potência do sinal recebido, apresentando no *Google Maps* a latitude e longitude do GPS e uma circunferência de raio calculado que mostra onde o módulo pode estar localizado. Deve-se levar em conta que a distância mínima para que o celular detecte a presença do módulo é de aproximadamente 100 metros, essa distância é definida pelo alcance dos padrões Wi-Fi 802.11b/g/n utilizados atualmente pelo módulo ESP8266.

Após concluir este projeto, verificou-se que o sistema de localização implementado executou a função de localização do módulo Wi-Fi através do aplicativo *Android*. Os valores de distância estimados pelo aplicativo com base na potência do sinal, calculado através da equação 7, apresentaram uma boa precisão após ser calculado um fator de correção para a equação. O módulo apresentou também um consumo relativamente baixo de energia, sendo possível otimizar a duração da bateria através da programação do módulo por meio de funções para baixo consumo de energia. Assim, esse protótipo pode ser utilizado futuramente na criação de produtos onde a solução seja localizar objetos que estarão associados ao circuito do módulo Wi-Fi cuja posição será estimada pelo aplicativo *Android*.

5.1. GPS X APLICAÇÃO ANDROID

Para as aplicações propostas caso fosse utilizado o sistema GPS para obter as coordenadas do objeto, seria necessário um módulo GPS além de um hardware adicional para envio das coordenadas obtidas outra desvantagem, é consumo elevado de energia por esse sistema. A principal vantagem do GPS é a sua precisão, que possui uma acurácia de aproximadamente 10 m.

Em comparação ao GPS o protótipo do sistema de localização desenvolvido tem como vantagens: menor consumo de bateria, precisão suficiente para as aplicações propostas, possibilidade de integrar sensores ao mesmo módulo com comunicação entre o módulo e o aplicativo para envio de dados lidos por esses sensores, um consumo de bateria pelo módulo relativamente baixo em comparação ao GPS; utilização dos recursos de conectividade dos smartphones pelo aplicativo para enviar/receber informações de forma segura. A principal desvantagem é o alcance limitado definido pelo padrão Wi-Fi (100 a 500 m), apesar disso, o sistema apresentou uma boa precisão uma vez que o aplicativo detectou a presença do módulo.

5.2. TRABALHOS FUTUROS

Como possíveis trabalhos futuros pode-se apontar:

- Uma atualização do aplicativo *Android*, com melhoras no design para uma interface mais amigável e intuitiva.
- Otimização na estrutura do banco de dados do sistema, através da reestruturação das classes do sistema e das funções de leitura e escrita no servidor na nuvem.
- Projeto do circuito do módulo ESP8266, através de uma ferramenta como o Eagle, de forma que o circuito projetado apresentado tenha um tamanho razoável.
- Aplicação do protótipo desse sistema na criação de produtos a serem comercializados, como a coleira para localizar animais por exemplo.
- Fazer uma calibração da distância por aparelho via software, uma vez que diferentes aparelhos podem apresentar diferentes ganhos de antena, o que pode influenciar na precisão do sistema.
- Caso o módulo ESP8266, usado no desenvolvimento desse projeto, utilizasse o Wi-Fi HaLow, o alcance do sinal cobriria o dobro da distância, com menor interferência em ambientes fechados e, ainda consumiria menos energia. Talvez em uma versão futura uma versão futura desse módulo implemente

esse padrão uma vez que a maioria das aplicações em que ele é usado é direcionado à Internet das Coisas.

REFERÊNCIAS

ALMEIDA, Hyggo. **Tudo Conectado**. Computação Brasil. 2015.p. 6-8.

API GOOGLE MAPS. Disponível em <https://developers.google.com/maps/documentation/android-api/?hl=pt-br>. Acesso em 20 out. 2016.

API FIREBASE. Disponível em:< https://firebase.google.com/docs/>. Acesso em 3 out. 2016.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Elsevier Brasil, 2006.

CURVELLO, André. **Apresentando o módulo ESP8266,** 2015. Disponível em < http://www.embarcados.com.br/modulo-esp8266/>. Acesso em 8 ago. 2016.

FAGUNDES, Leonardo Peres. **Técnicas de localização de dispositivos móveis em redes WiFi – TDOA**, 2008. Disponível em <http://www.lume.ufrgs.br/handle/10183/15975>. Acesso em 10 ago. 2016.

FARIA, Carolina. **GPS (Sistema de Posicionamento Global)**. Disponível em < http://www.infoescola.com/cartografia/gps-sistema-de-posicionamento-global/>. Acesso em 2 nov. 2016.

FERREIRA, Thais. **Internet das coisas: o mundo conectado.** Disponível em <http://www.dcomercio.com.br/categoria/tecnologia/internet_das_coisas_o_mundo_conectado>. Acesso em 22 nov. 2016.

FRANÇA, Tiago C. de; PIRES, Paulo F.; PIRMEZ, Luci; DELICATO, Flávia C.; FARIAS, Claudio. **Web das Coisas: Conectando Dispositivos Físicos ao Mundo Digital**. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC).2011. p.1-8. Disponível em <http://www.labnet.nce.ufrj.br/download/minicurso-wot-final.pdf>. Acesso em 12 ago. 2016.

FURLAN, José Davi. **Modelagem de Objetos através da UML**. São Paulo: Makron Books, 1998.

GPS EDUCATIONAL POSTER. Disponível em <http://www.gps.gov/multimedia/poster/>. Acesso em 2 nov. 2016.

IEEE. **NEW IEEE 802.11ac™ SPECIFICATION DRIVEN BY EVOLVING MARKET NEED FOR HIGHER, MULTI-USER THROUGHPUT IN WIRELESS LANS.** Disponível em <http://standards.ieee.org/news/2014/ieee_802_11ac_ballot.html>. Acesso em 29 nov. 2016.

INFOGRÁFICO INTERNET DAS COISAS SAS. Disponível em <http://www.sas.com/pt_br/insights/big-data/internet-das-coisas.html#m=the-internet-of-things-infographic>. Acessado em 12 ago. 2016.

IOT MINILAB. Disponível em <https://iotminilab.blogspot.com.br/p/blog-page.html>. Acesso em 30 ago. 2016.

LATITUDE E LONGITUDE. **Imagem**. Disponível em <https://www.google.com.br/imgres?imgurl=http%3A%2F%2Fwww.isobudgets.com%2Fwp-content%2Fuploads%2F2014%2F03%2Flatitude-longitude.jpg&imgrefurl=http%3A%2F%2Fwww.isobudgets.com%2Fhow-to-calculate-local-gravity%2F&docid=cqgi2H-vXolwVM&tbnid=Aj_xPbUvZjghFM%3A&vet=1&w=600&h=298&bih=638&biw=1366&ved=0ahUKEwiwusS32tjQAhUSPJAKHSz1A0YQMwgxKAAwAA&iact=mrc&uact=8>. Acesso em 30 nov. 2016.

LIPSTICK POWER BANK. **Imagem**. Disponível em <https://www.google.com.br/imgres?imgurl=http%3A%2F%2Fwww.dhresource.com%2Falbu_791551164_00-1.0x0%2Flipstick-power-bank-2600mah-universal-for.jpg&imgrefurl=http%3A%2F%2Fwww.dhgate.com%2Fproduct%2Flipstick-power-bank-2600mah-universal-for%2F163350700.html&docid=iGdoJE1Sva0RvM&tbnid=aeZA42f2AvBNAM%3A&vet=1&w=800&h=800&bih=638&biw=1366&ved=0ahUKEwiB-s75vc_QAhXCE5AKHdVLBG0QMwgtKBAwEA&iact=mrc&uact=8>. Acesso em 29 nov. 2016.

MARTINS, Bruno; COSTA, Rodolfo Machado Brandão; QUEIROZ, Rodrigo Leite de. **IEEE 802.11 a,b,g,n**. Disponível em <http://www.gta.ufrj.br/grad/13_1/80211abgn/index.html**>**. Acesso em 29 nov. 2016.

MARTINS, Elaine. **Como funciona o GPS?** Disponível em <http://www.tecmundo.com.br/gps/2562-como-funciona-o-gps-.htm>. Acesso em 2 nov. 2016.

NASA. **Global Positioning System History**, 2012. Disponível em <https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html>. Acesso em 2 nov. 2016.

PEREIRA, Ezequiel da Veiga. **Desenvolvimento de um sistema de localização de fontes de Rádio Frequência para aplicações indoor**. Disponível em <http://paginas.fe.up.pt/~ee01272/DIS/lib/exe/fetch.php?media=mieec.pdf>. Acesso em 29 nov. 2016.

PRADO, Jean. Wi-Fi HaLow é o novo padrão de rede para a internet das coisas. Disponível em< https://tecnoblog.net/190048/wi-fi-halow-internet-das-coisas/>. Acesso em 29 nov. 2016.

REGHELIN, Ricardo. **Um algoritmo descentralizado de localização para redes de sensores sem fio usando calibragem cooperativa e heurísticas.** Disponível em

<https://repositorio.ufsc.br/bitstream/handle/123456789/90148/246134.pdf?sequence=1>. Acesso em 29 nov. 2016.

SOLYSION. **Conheça os padrões de comunicação sem fio para a Internet das Coisas (IOT)**. Disponível em < http://solysion.com.br/eletronica/conheca-os-padroes-de-comunicacao-sem-fio-para-a-internet-das-coisas-iot/>. Acesso em 29 nov. 2016.

SIGNIFICADOS. **Significado de Latitude e Longitude**. Disponível em <https://www.significados.com.br/latitude-e-longitude/>. Acesso em 30 nov. 2016.

SINGER, Tayla. **Tudo conectado: conceitos e representações da Internet das Coisas**. Salvador, 2012. Disponível em <http://www.simsocial2012.ufba.br/modulos/submissao/Upload/44965.pdf>. Acesso em 8 ago. 2016.

TANEMBAUM, A. S. **Redes de Computadores**. 4 ed., Ed. Campus, Rio de Janeiro, 2003.

THOMSEM, Adilson. **Tutorial módulo Wireless ESP8266 com Arduino**, 2015. Disponível em <http://blog.filipeflop.com/wireless/esp8266-arduino-tutorial.html>. Acesso em 5 ago. 2016.

APÊNDICES

Apêndice A – Código Módulo ESP8266

Código salvo na memória do módulo ESP8266, que define o módulo como um Ponto de Acesso. As linhas que começam com dois hifens são comentários.

```
-- Setando modo Wifi para estação e Ponto de Acesso
wifi.setmode(wifi.STATIONAP)

-- Access Point com senha
cfg={}
--definir nome da rede (SSID)
cfg.ssid="testeWiFi"
cfg.pwd="senha12345678"
wifi.ap.config(cfg)


-- Timer de 1 em 1 segundo para aguardar conectar
local cnt = 0
local maxtry = 30
tmr.alarm(0, 1000, 1, function()
if (wifi.ap.getip() == nil) and (cnt <= maxtry) then
print("Configurando Access Point, aguarde...")
cnt = cnt + 1
else
tmr.stop(0);
print("Configuração efetuada com sucesso")
if (cnt <= maxtry) then
print("Access Point\r\nMAC:"..wifi.ap.getmac().."\r\nIP:"..wifi.ap.getip())
else
print("Timeout configurando access point")
end
cnt = nil;
collectgarbage();
end
end)

-- Configurando Servidor Http
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
conn:on("receive",function(conn,payload)
print(payload)
conn:send("<h1> Hello, NodeMCU.</h1>")
end)
```

end)

Apêndice B – Classe Dados

```
package com.example.mauri.appcoleira;


/**
 * Created by mauri on 20/09/2016.
 */
public class DadosColeira{
    private int id;

    private String IdColeira;
    private String NomeDonoPet;
    private String Endereco;
    private String Telefone;

    // Dados pet
    private String NomePet;
    private String status; //false= não perdido, true perdido

    //Dados Busca
    private String latitude;
    private String longitude;
    private String distancia;
    private String buscadatacompleta;            //data e hora em que a
busca foi feita

    public DadosColeira (){
        // Default constructor required for calls to
DataSnapshot.getValue(User.class)
    }

    public DadosColeira(int ID,String idcol,String nd,String end,String
tel,String np, String stat, String lat, String Long, String dist, String
data){ // String data, String hora, GPSTracker GPS ){
        boolean temp = false;
        this.id = ID;
        this.IdColeira = idcol;
        this.NomeDonoPet = nd;
        this.Endereco = end;
        this.Telefone = tel;
        this.NomePet = np;
        this.status = String.valueOf(temp);
        this.latitude = "0";
        this.longitude = "0";
        this.distancia = "0";
        this.buscadatacompleta = "0";
    }



    public DadosColeira(String idcol, String nd, String end, String tel,
String np){ // String data, String hora, GPSTracker GPS ){
        boolean temp = false;
        this.IdColeira = idcol;
        this.NomeDonoPet = nd;
```

```
        this.Endereco = end;
        this.Telefone = tel;
        this.NomePet = np;
        this.status = String.valueOf(temp);
        this.latitude = "0";
        this.longitude = "0";
        this.distancia = "0";
        this.buscadatacompleta = "0";
    }

    public DadosColeira (DadosColeira d){
        this.IdColeira = d.getIdColeira();
        this.NomeDonoPet = d.getNomeDonoPet();
        this.Endereco = d.getEndereco();
        this.Telefone = d.getTelefone();
        this.NomePet = d.getNomePet();
        this.status = d.isStatus();
        this.latitude = d.getLatitude();
        this.longitude = d.getLongitude();
        this.distancia = d.getDistancia();
        this.buscadatacompleta = d.getBuscadatacompleta();
    }

    public DadosColeira(String idcol,String nd,String end,String tel,String
np, String stat, String lat, String Long, String dist, String data){ //
String data, String hora, GPSTracker GPS ){
        this.IdColeira = idcol;
        this.NomeDonoPet = nd;
        this.Endereco = end;
        this.Telefone = tel;
        this.NomePet = np;
        this.status = stat;
        this.latitude = lat;
        this.longitude = Long;
        this.distancia = dist;
        this.buscadatacompleta = data;
    }

    public DadosColeira setDadosColeira (DadosColeira dados){
        DadosColeira dadosalterados = new DadosColeira(dados);

        return dadosalterados;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }


    public String getIdColeira() {
        return IdColeira;
    }

    public String getNomeDonoPet() {
        return NomeDonoPet;
    }
```

```
public String getEndereco() {
    return Endereco;
}

public String getTelefone() {
    return Telefone;
}

public String getNomePet() {
    return NomePet;
}

public String isStatus() {
    return status;
}

public String getLatitude() {
    return latitude;
}

public String getLongitude() {
    return longitude;
}

public String getDistancia() {
    return distancia;
}

public String getBuscadatacompleta() {
    return buscadatacompleta;
}

public void setIdColeira(String idColeira) {
    IdColeira = idColeira;
}

public void setNomeDonoPet(String nomeDonoPet) {
    NomeDonoPet = nomeDonoPet;
}

public void setEndereco(String endereco) {
    Endereco = endereco;
}

public void setTelefone(String telefone) {
    Telefone = telefone;
}

public void setNomePet(String nomePet) {
    NomePet = nomePet;
}

public void setStatus(String status) {
    this.status = status;
}

public void setLatitude(String latitude) {
    this.latitude = latitude;
}

public void setLongitude(String longitude) {
```

```
        this.longitude = longitude;
    }

    public void setDistancia(String distancia) {
        this.distancia = distancia;
    }

    public void setBuscadatacompleta(String buscadatacompleta) {
        this.buscadatacompleta = buscadatacompleta;
    }


}
```

Apêndice C - Código Java tela de busca

```
package com.example.mauri.appcoleira;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
//import android.widget.ArrayAdapter;
//import android.widget.ListView;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.ArrayList;
import java.util.List;
import java.lang.Math;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class BuscarAnimais extends AppCompatActivity {

    private Button botaoVoltar;
    private WifiManager mainWifi;
    private WifiReceiver receiverWifi;
    //private StringBuilder sb = new StringBuilder();
    private TextView textView;
    private final Handler handler = new Handler();
    //private ListView lv;

    // SSID deve ser o mesmo da rede Wi-Fi criada pelo módulo
    public static final String NomeRede = "TESTE_WIFI";
    //CALIBRAÇÃO CÁLCULO DISTÂNCIA
    public static final double ajuste = 1/(31.626); // APÓS SOMAR 0.515

    private DatabaseReference rootRef =
FirebaseDatabase.getInstance().getReference();
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_buscar_animais);

        //lv = (ListView) findViewById(R.id.listView);
        botaoVoltar=(Button)findViewById(R.id.botaovoltar_telabusca);
        textView =(TextView)findViewById(R.id.tv_telabusca);

        mainWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

        receiverWifi = new WifiReceiver();

        if(!mainWifi.isWifiEnabled()){
            mainWifi.setWifiEnabled(true);
        }

        registerReceiver(receiverWifi, new IntentFilter(
                WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));

        doInback();
    }

    private void doInback() {
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                mainWifi = (WifiManager)
getSystemService(Context.WIFI_SERVICE);

                if (receiverWifi==null) {
                    receiverWifi = new WifiReceiver();
                }

                registerReceiver(receiverWifi, new IntentFilter(
                        WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
                mainWifi.startScan();

                doInback();
            }
        }, 10000); // 10 segundos
    }

    protected void OnPause(){
        unregisterReceiver(receiverWifi);
        super.onPause();
    }

    protected void onResume()
    {
        registerReceiver(receiverWifi, new IntentFilter(
                WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
        super.onResume();
    }

    private class WifiReceiver extends BroadcastReceiver {

        public void onReceive (Context c, Intent intent){
            ArrayList<String> connections = new ArrayList<String>();
            //ArrayList<Float> Signal_Strenth = new ArrayList<Float>();
```

```
            //sb = new StringBuilder();
            List<ScanResult> wifiList;
            wifiList = mainWifi.getScanResults();
            String wifis[];
            wifis = new String[wifiList.size()];
            String teste;


            for(int i = 0; i < wifiList.size(); i++){
                connections.add(wifiList.get(i).SSID);

                wifis[i] = ((wifiList.get(i)).toString());
                //Signal Strenth.add(wifiList.get(i).);

                teste = wifiList.get(i).SSID;

                if(teste.equals(NomeRede)){ // codificar nome da rede do
transmissor para identificar um padrão
                    //COLETAR INFORMAÇÕES
                    coletaInfo(wifiList.get(i).BSSID,
wifiList.get(i).level); //variável recebe retorno função, após armazena em
buffer
                }
            }
            //lv.setAdapter(new
ArrayAdapter<String>(getApplicationContext(),android.R.layout.simple list i
tem_1,wifis));
        }
    }

    public void coletaInfo (String  id, int level){
        String texto;
        double latitude = 0, longitude = 0;
        double potrdb = level; //potência recebida em dbm
        double potrW; //Potência recebida em Wats
        double pottW; //Potência transmitida em Wats
        double distancia; // variável que armazena a distância (raio em m)
do celular em relação ao módulo transmissor


        // instancia o service, GPSTracker gps
        GPSTracker gps = new GPSTracker(BuscarAnimais.this);

        if(gps.canGetLocation()){
            latitude = gps.getLatitude();
            longitude = gps.getLongitude();
        }


        //Calculo da distancia utilizando a fórmula de Friiss
        // (Pr/Pt) = Gt*Gr* [(lambda/4*PI*D)]
        //lambda = C/F; C veloc. da luz, F frequência
        //lambda = 0.12248 // comprimento de onda do sinal Wi-Fi
(Frequência = 2.4 GHz)


        int Gt = 1; //Ganho da antena de transmissão
        int Gr = 1; //Ganho da antena de recepção


        //Potência transmissão: 20 dBm = 100 mW = 0.1 W
```

```
        pottW = 0.1;
        // Cálculo da Potência recebida em W
        potrW = (Math.pow(10, (potrdb/10))) * 0.001;
        //Cálculo da distância
        // distancia = (0.12248/(4 * Math.PI)) * Math.sqrt((pottW * Gt *
Gr)/ (potrW)); // VALOR NÃO CALIBRADO
        distancia = ((0.12248/(4 * Math.PI)) * Math.sqrt((pottW * Gt * Gr)/
(potrW))) * ajuste + 0.515; // VALOR CALIBRADO

        //buscar informações de data/hora
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy-
HH:mm:ss");
        Date data = new Date();
        Calendar cal = Calendar.getInstance();
        cal.setTime(data);
        Date data_atual = cal.getTime();

        String data_completa = dateFormat.format(data_atual);

        //Dados coletados: id, latitude, longitude, distância (Armazenar em
uma classe?)
        texto ="Id:" + id.toString() + " Latitude:" +
String.valueOf(latitude) + " Longitude:" + String.valueOf(longitude) + "
Ditancia(m):" + String.valueOf(distancia)
        + " Data:" + data_completa;

        textView.setText(texto);

        //Alteração no banco de dados
        DatabaseReference childref = rootRef.child(id);

        DatabaseReference Lat = childref.child("latitude");
        Lat.setValue(String.valueOf(latitude));

        DatabaseReference Long = childref.child("longitude");
        Long.setValue(String.valueOf(longitude));

        DatabaseReference dist = childref.child("distancia");
        dist.setValue(String.valueOf(distancia));

        DatabaseReference date = childref.child("buscadatacompleta");
        date.setValue(data_completa);


    }

    public void back_tomain (View view){
        Intent mainactivity = new Intent(this, MainActivity.class);
        startActivity(mainactivity);
    }
}
```