**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 7 de dezembro de 2016:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Carlos Henrique Barriquello, Dr. (UFSM)**

(Presidente/Orientador)

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 composite 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 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 “embebed” which will be find by the mobile application developed. However, this system can be used for several other applications, like to find other objects in several environments, example: bags in a airport. This applications are directly related with the concept of the Internet of Things, which says that objects o four day to day 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 na 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** 8](#_Toc468061792)

[1.1. MOTIVAÇÃO 9](#_Toc468061793)

[1.2. OBJETIVOS 9](#_Toc468061794)

[1.2.1. Objetivo geral 9](#_Toc468061795)

[1.2.2. Objetivos específicos 10](#_Toc468061796)

[**2.** **Internet das Coisas** 10](#_Toc468061797)

[**3.** **A tecnologia Wi-Fi** 12](#_Toc468061798)

[4. **Técnicas de Localização em Radiofrequência** 13](#_Toc468061799)

[4.1.1. ÂNGULO DE CHEGADA 14](#_Toc468061800)

[4.1.2. INDICADOR DE FORÇA DO SINAL RECEBIDO 14](#_Toc468061801)

[4.1.3. TEMPO DE CHEGADA 14](#_Toc468061802)

[4.1.4. DIFERENÇA DO TEMPO DE CHEGADA 15](#_Toc468061803)

[**5.** **O Sistema GPS** 15](#_Toc468061804)

[5.1. HISTÓRIA 15](#_Toc468061805)

[5.2. PRINCÍPIOS DE FUNCIONAMENTO 16](#_Toc468061806)

[**6.** **O Módulo ESP8266** 17](#_Toc468061807)

[6.1. Características de Hardware 19](#_Toc468061808)

[6.2. Características de Operação 19](#_Toc468061809)

[6.3. O Módulo ESP – 12E 20](#_Toc468061810)

[**7.** **Descrição do Sistema** 20](#_Toc468061811)

[**8.** **Desenvolvimento Aplicação Android** 22](#_Toc468061812)

[8.1. API do Google Maps 23](#_Toc468061813)

[8.2. API do Firebase 23](#_Toc468061814)

[**9.** **MODELAGEM** 25](#_Toc468061815)

[**9.1.** DIAGRAMA DE CASO DE USO 25](#_Toc468061816)

[**9.2.** DIAGRAMA DE CLASSES 26](#_Toc468061817)

[**9.3.** DIAGRAMA DE SEQUÊNCIA 28](#_Toc468061818)

[**10.** **TELAS DO APLICATIVO** 29](#_Toc468061819)

[10.1. Usuário 30](#_Toc468061820)

[10.2. Cliente 31](#_Toc468061821)

[**11.** **RESULTADOS OBTIDOS** 34](#_Toc468061822)

[11.1. TESTE DE ALCANCE 34](#_Toc468061823)

[11.2. TESTE DE BATERIA 38](#_Toc468061824)

[**12.** **GPS X Aplicação Android** 38](#_Toc468061825)

[**CONCLUSÕES** 40](#_Toc468061826)

[**REFERÊNCIAS** 41](#_Toc468061827)

[APÊNDICES 43](#_Toc468061828)

[Apêndice A – Código ESP8266 43](#_Toc468061829)

[Apêndice B – Código Java tela de busca 44](#_Toc468061830)

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 *Acces 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.

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

O sistema desenvolvido nesse projeto é uma aplicação da Internet das Coisas (*Internet of Things*), por isso, no capítulo 2 é feito uma apresentação sobre o que é, perspectivas e aplicações.

A tecnologia Wi-Fi, um dos recursos utilizados nesse projeto, é apresentada no capítulo 3, onde é mostrado o que motivou seu surgimento e como evoluiu até se tornar a Wi-Fi que conhecemos hoje.

As técnicas de localização em radiofrequência são vistas no capítulo 4, das quais uma é utilizada no sistema GPS e outra é utilizada nesse projeto para determinar a posição no espaço.

O GPS, sistema de localização mais utilizado atualmente, é apresentado no capítulo 5.

No capítulo 6, é apresentado o módulo ESP8266, módulo Wi-Fi utilizado no sistema de localização implementado.

No capítulo 7, é feita uma descrição do funcionamento do sistema desenvolvido nesse projeto.

No capítulo 8, é apresentado como o aplicativo foi implementado, com uma descrição das APIs (*Application Programming Interface*) utilizadas e suas funcionalidades.

No capítulo 9 é apresentado como o aplicativo foi modelado, são apresentados alguns diagramas como os de casos de uso, diagrama de classes e diagrama de sequência das principais funcionalidades.

No capítulo 10 são vistas algumas das telas do aplicativo desenvolvido com descrição de suas funcionalidades.

No capítulo 11 são mostrados alguns experimentos realizados para testar a performance do sistema desenvolvido, um teste de alcance (distância calculada pelo aplicativo) e um teste de duração de bateria para o módulo ESP8266.

* 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 (MAC, *Media Acces 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. OBJETIVOS
     1. Objetivo geral

O objetivo geral deste projeto é implementar o protótipo de um sistema de localização com o módulo Wi-Fi, a parte funcional do sistema será executada por um aplicativo Android, que detectará a presença do sinal emitido por esse módulo e estimará a distância com base na potência do sinal recebido.

* + 1. Objetivos específicos

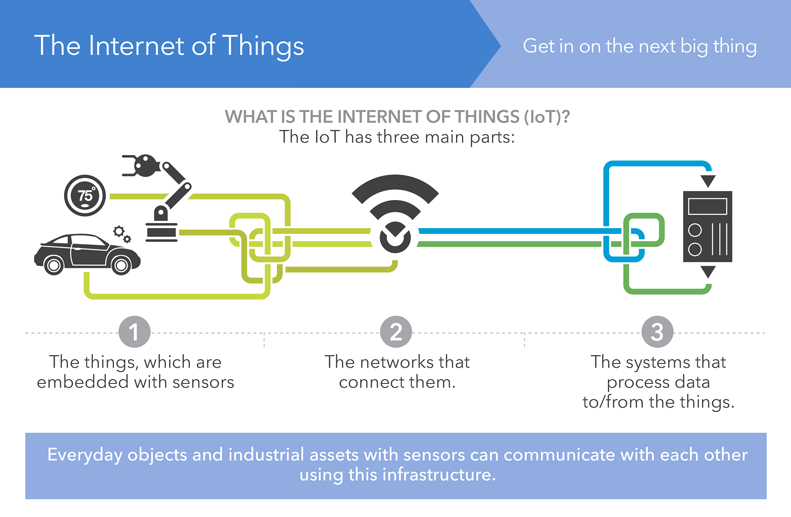
1. Programar o módulo configurando-o como *Access Point,* criando assim uma rede Wi-Fi a qual será detectada pelo aplicativo de celular através do SSID (*Service Set IDentifier*), que é uma *string* de até 32 caracteres que identifica cada rede sem fio (nome da rede Wi-Fi).
2. Implementar a aplicação *Android* para detectar o sinal do módulo e calcular a distância através da potência do sinal recebido.
3. Verificar a eficiência do sistema com testes de precisão e estimativa de consumo de bateria pelo módulo Wi-Fi.
4. **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, 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 – A internet das Coisas



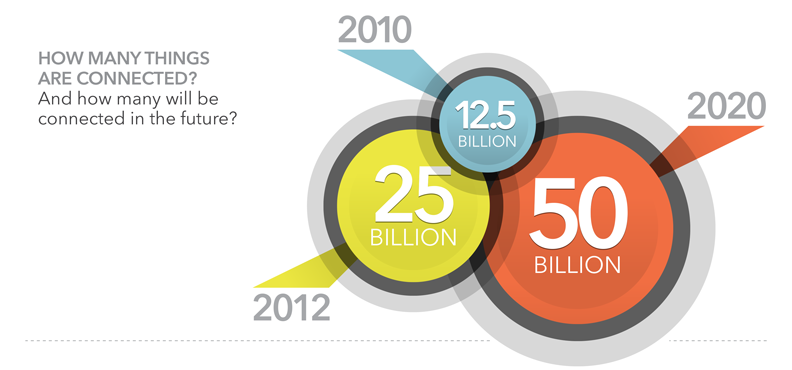
(Fonte: Infográfico SAS).

De acordo com 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 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 IoT.

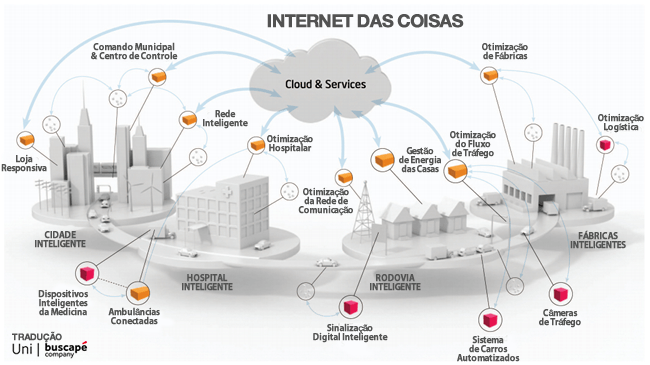
Figura 2.2 – Número de coisas conectadas



(Fonte: Infográfico SAS).

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

Figura 2.3 – Aplicações Internet das Coisas



(Fonte: Ferreira).

1. **A tecnologia Wi-Fi**

Hoje em dia o Wi-Fi (*Wireless Fidelety*) é um dos padrões mais difundidos de acesso à internet, estando presente em praticamente todo lugar e sendo compatível com praticamente qualquer dispositivo moderno que forneça suporte a esse serviço.

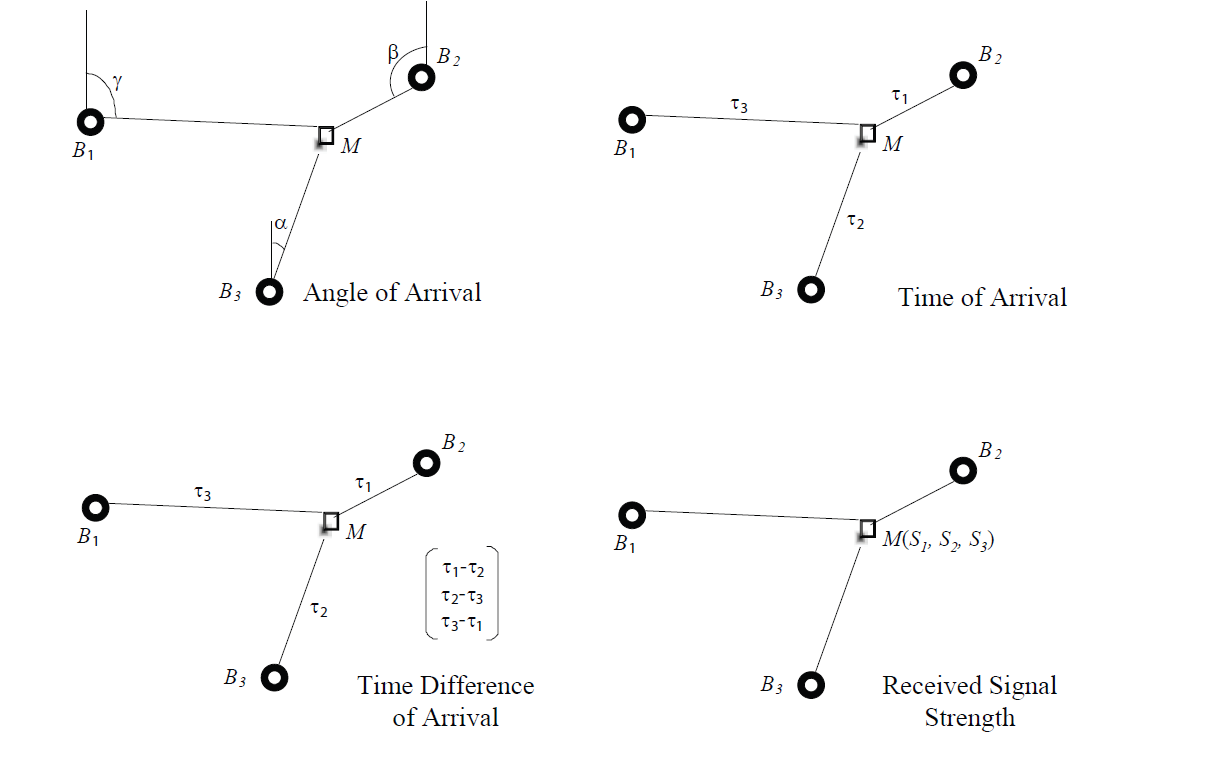
De acordo com Tanenbaum (2003), esse padrão teve início na mesma época do surgimento dos *notebooks*, muitas pessoas queriam que seu dispositivo móvel se conectasse automaticamente à internet assim que entrassem no escritório. Muitos grupos começaram a trabalhar para alcançar esse objetivo e, a solução 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 com fio, elaborar um padrão de LANs sem fio. O padrão recebeu o nome de 802.11, também conhecido popularmente por Wi-Fi.

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 ainda 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.

1. **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*).

Figura 4.1 – Diferentes técnicas de localizar a fonte RF.

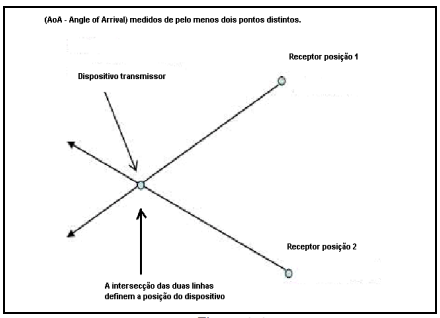


(Fonte: Pereira apud Guoqiang).

* + 1. ÂNGULO DE CHEGADA

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 no mínimo duas medições entre o transmissor e o receptor para localização no plano 2D, e 3 medições para localização em 3 dimensões (triangulação). 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.

Figura 4.1.1 - Representação da técnica AoA.



(Fonte: ).

* + 1. INDICADOR DE FORÇA DO SINAL RECEBIDO

A técnica RSSI (*Received Signal Strength Indicator)* utiliza a potência do sinal recebido para estimar a distância entre a estação e o ponto de acesso.

* + 1. TEMPO DE CHEGADA

O cálculo da posição de um dispositivo móvel é feito através da medição do tempo de propagação de um sinal desde o emissor até o receptor. A velocidade de propagação é conhecida, logo infere-se a distância entre emissor e receptor.

A interseção das circunferências indica a possível posição do dispositivo móvel.

Existe a necessidade de um sincronismo rigoroso entre estações base e móvel, bem como alterações a nível de hardware.

* + 1. DIFERENÇA DO TEMPO DE CHEGADA

1. **O Sistema GPS**

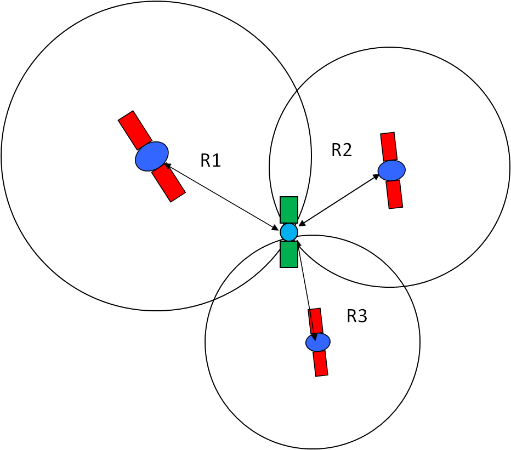
Em termos de sistemas de localização, a tecnologia mais difundida atualmente é a do GPS, estando presente em diversas aplicações e áreas em nosso cotidiano como: navegação, agricultura, segurança, localização, etc.

* 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.

Figura 5.1.1 – Sinais de rádio GPS

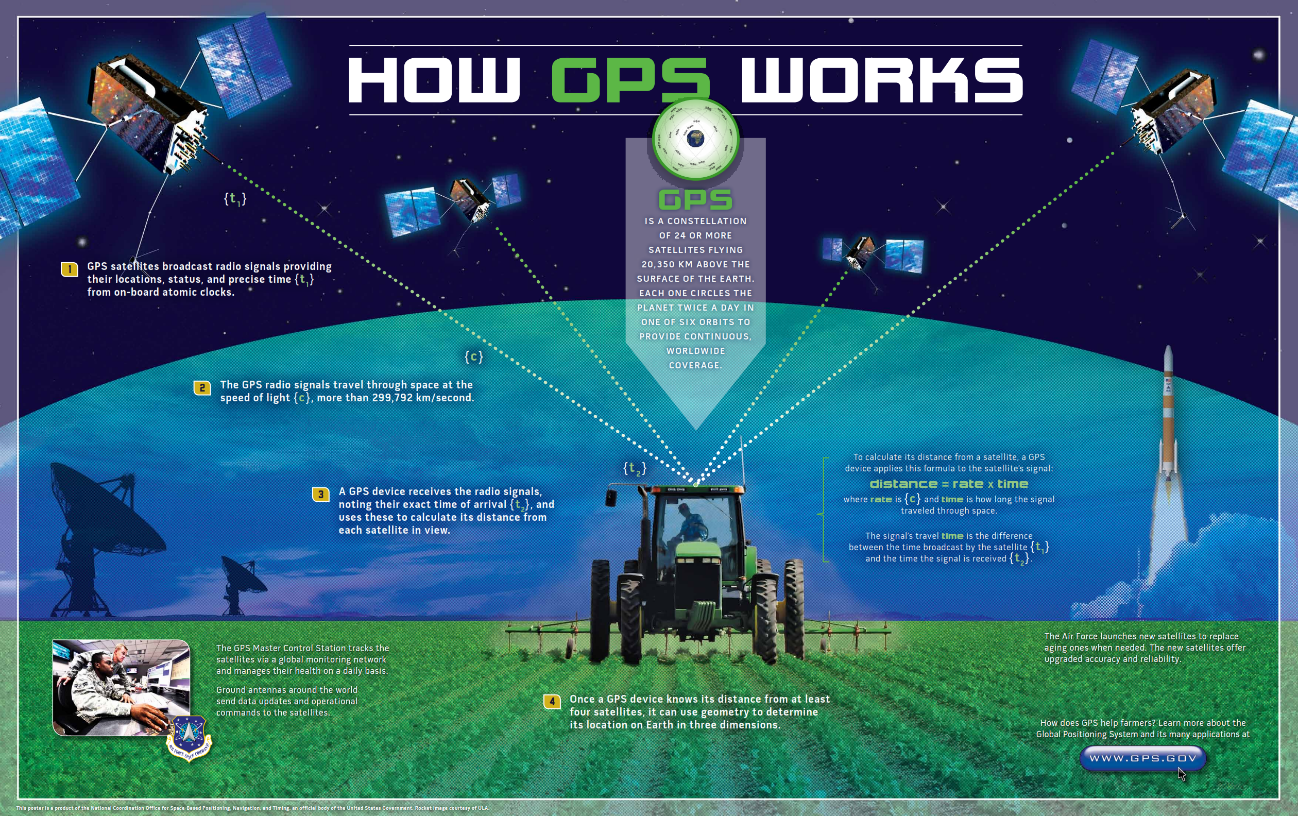


(Fonte: NASA).

* 1. PRINCÍPIOS DE FUNCIONAMENTO

O GPS utiliza a técnica TOA para determinar a posição no espaço. A imagem da Figura explica como um dispositivo GPS utiliza essa técnica para determinar sua localização.

Figura 5.1.2 – 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 300 Km/h. 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:

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.

Figura 5.1.3 – Triangulação sistema GPS



(Fonte: Martins).

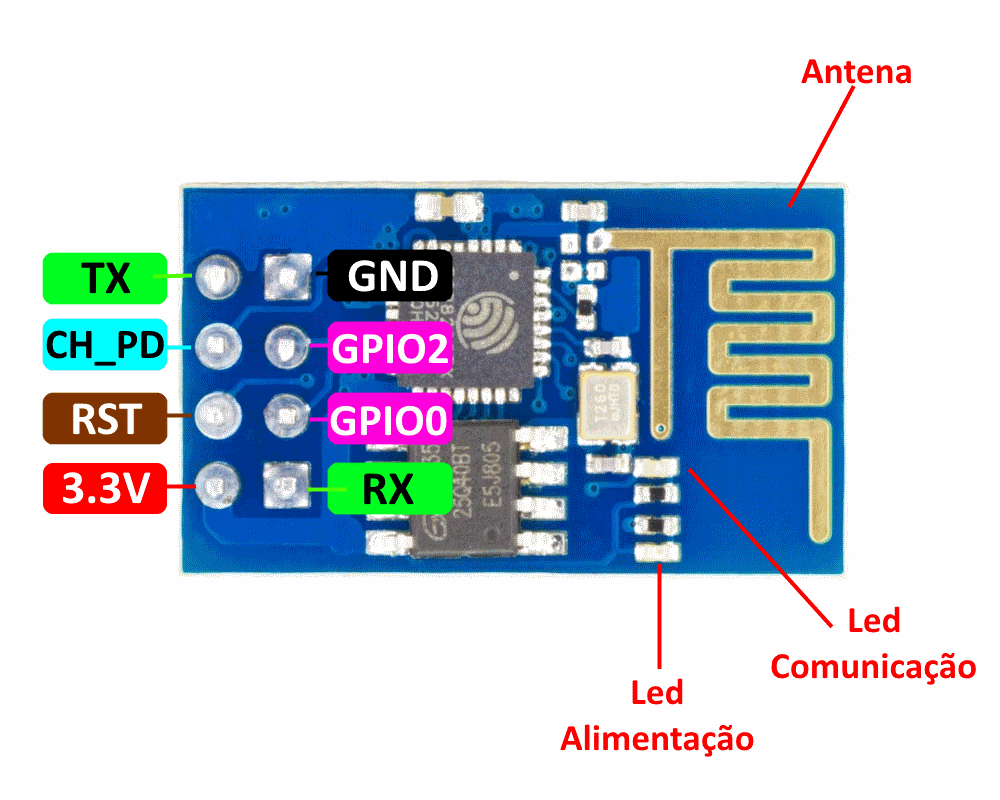
1. **O Módulo ESP8266**

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, um poder de processamento e tamanho reduzido. 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 a seguir mostra o ESP-01 e após e feita uma descrição dos pinos.

Figura 6 – Módulo ESP-01



(Fonte: Thomsem).

O quadro a seguir apresenta a descrição dos pinos do módulo da figura 6 .

Quadro 6 – Descrição pinos módulo ESP8266

|  |  |
| --- | --- |
| 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 *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).

* 1. 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.

* 1. 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, *Acces Point*, Cliente + *Acces 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;

* 1. O Módulo ESP – 12E

Por padrão os módulos saem de fábrica 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 a seguir mostra o módulo EPSP-12E.

Figura 6.3 – Módulo ESP-12E



(Fonte: IOT MINILAB).

O código implementado para que o módulo opere como *Acces 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.

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.

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:

(7)

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 7 – 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.

1. **Desenvolvimento 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, mas é tudo muito bem explicado na documentação de cada uma.

* 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.

Figura 8.1 – 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.

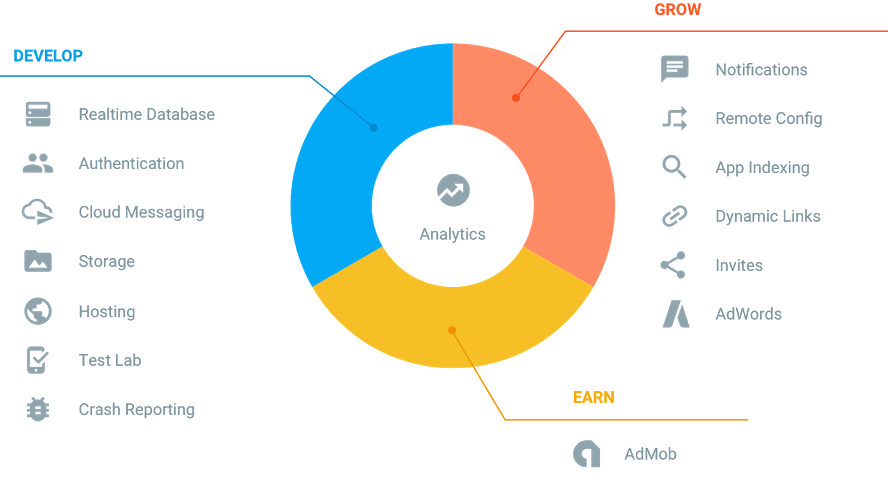
* 1. API do Firebase

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 8.2 – Recursos API Firebase



(Fonte: Documentação API Firebase).

Desntre os recursos mostrados na figura 8.2, no aplicativo desenvolvido foi utilizado o Realtime Database, que é um banco de dados hospedado na nuvem. Os dados 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.

1. **MODELAGEM**

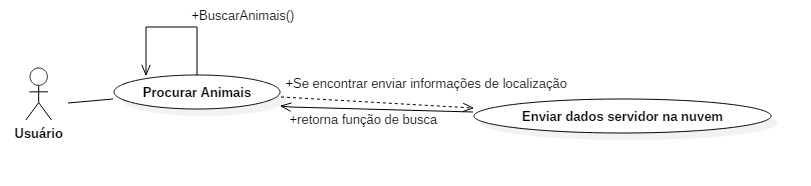
As principais funcionalidades do aplicativo desenvolvido são demonstradas através de linguagem UML (*Unified Modeling Language*). A UML é uma linguagem gráfica para visualização, construção e documentação de artefatos de sistemas complexos de software (Booch, 2006), geralmente utilizando orientação a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

* 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. No diagrama a seguir, é feita a descrição do caso de uso, explicando o fluxo principal da aplicação para um melhor entendimento da ordem dos eventos.

Figura 9.1 – Diagrama caso de uso função de busca

****

(Fonte: Elaborada pelo autor).

No quadro 9.1 é feita a descrição do caso de uso da imagem da figura 9.1.

Quadro 9.1 – Descrição do caso de uso.

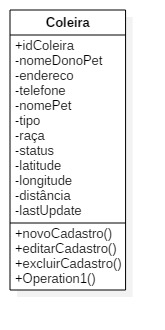
|  |  |
| --- | --- |
| Descrição | Este caso de uso permite que o usuário procure animais. Ao encontrar, o aplicativo envia as informações de localização determinadas pelo aplicativo para o servidor na nuvem. |
| Caso de Uso | Buscar Animais |
| Ator | Usuário |
| Pré-condição | Usuário selecionar função 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).

* 1. 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.

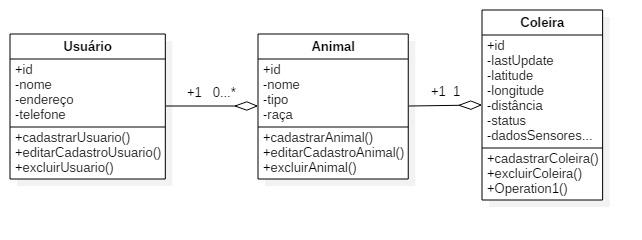
Figura 9.2 – Classe Coleira.

****

(Fonte: Elaborada pelo autor).

Esse modelo de classe única encapsulando todos os atributos não foi a melhor a escolha pois caso um usuário deseje cadastrar mais de um animal com coleira o banco de dados do servidor ficará com informações redundantes e, caso o usuário altere alguma informação, como o endereço por exemplo, será necessário alterar esse dado em todos os cadastros relacionados a esse usuário no banco de dados. Uma possível solução para esse problema é mostrada no diagrama de classes da figura .

Figura 9.2 – Diagrama de classes otimizado.

****

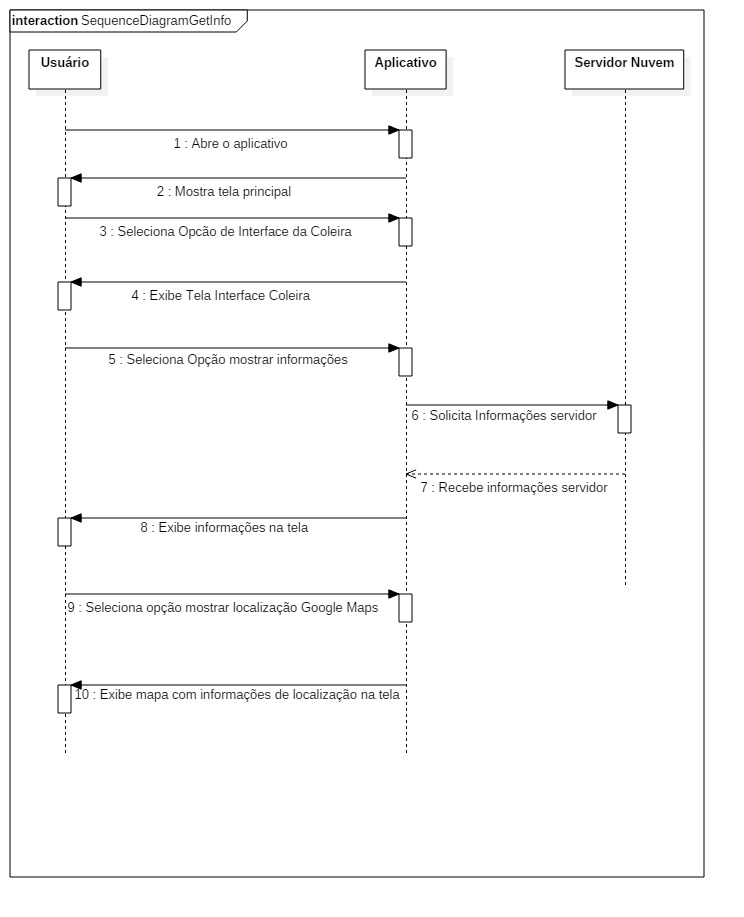
(Fonte: Elaborada pelo autor).

Dessa forma, um usuário pode cadastrar mais de um animal e, caso algum atributo seja alterado no banco de dados, essa informação estará disponível para todos os objetos relacionados a ele. Essa estrutura pode ser utilizada em uma futura versão da aplicação. Métodos de escrita e leitura precisam ser definidos para enviar/receber informações para o servidor do *Firebase Database*, que utiliza a estrutura JSON.

* 1. 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 mostra como o usuário utiliza a aplicação para visualizar as informações de localização do animal cadastrado.

Figura 9.3 – Diagrama sequência mostrar informações de localização da coleira.

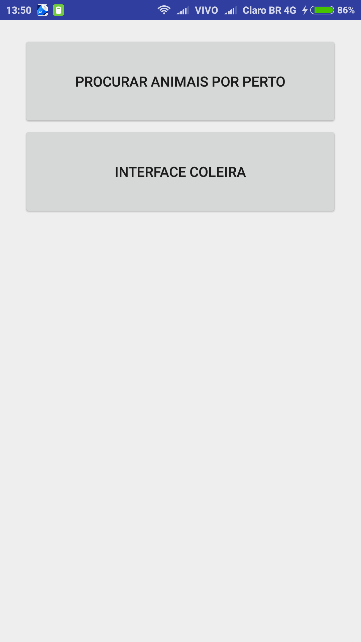
****

(Fonte: Elaborada pelo autor).

1. **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 mostra a tela inicial do aplicativo, onde é possível escolher entre a função de busca e a interface do cliente.

Figura 10 – Tela inicial do aplicativo.



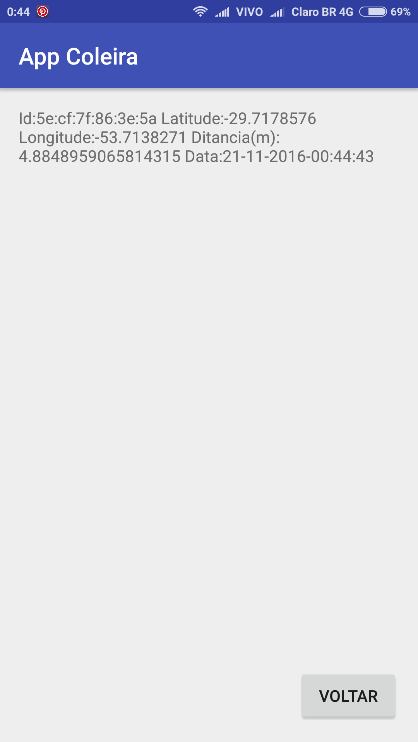
(Fonte: *print screens* do aplicativo no sistema Android).

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

* 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.

Figura 10.1 – Tela função de busca.

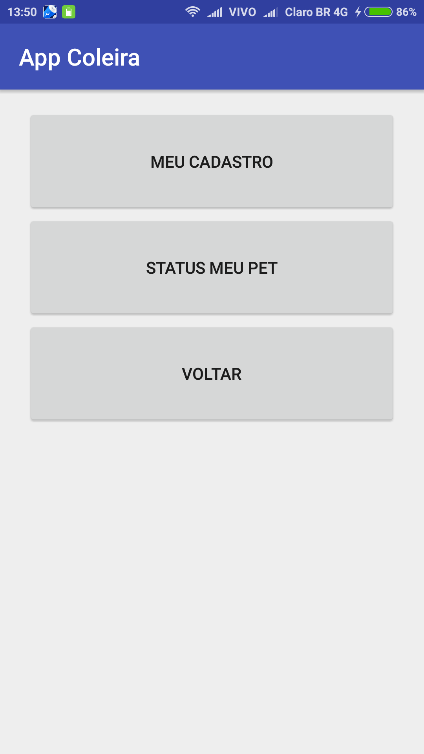


(Fonte: *print screens* do aplicativo no sistema Android).

* 1. Cliente

Ao entrar na interface do cliente, são exibidas opções de cadastro e de visualização de status do pet.

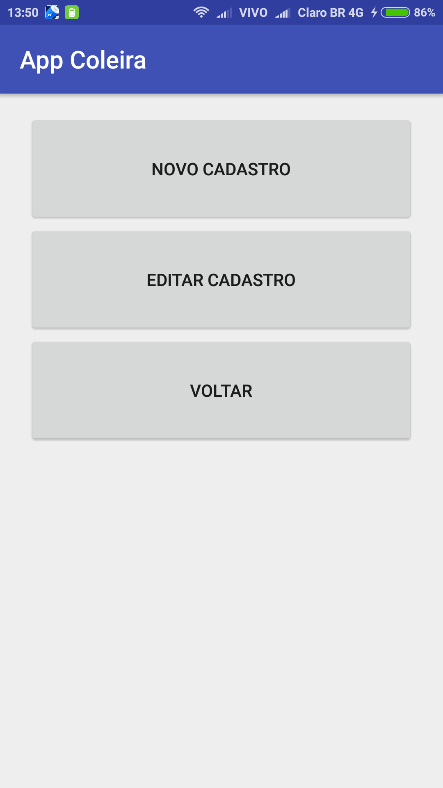
Figura 10.2 – Tela interface cliente.



(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.

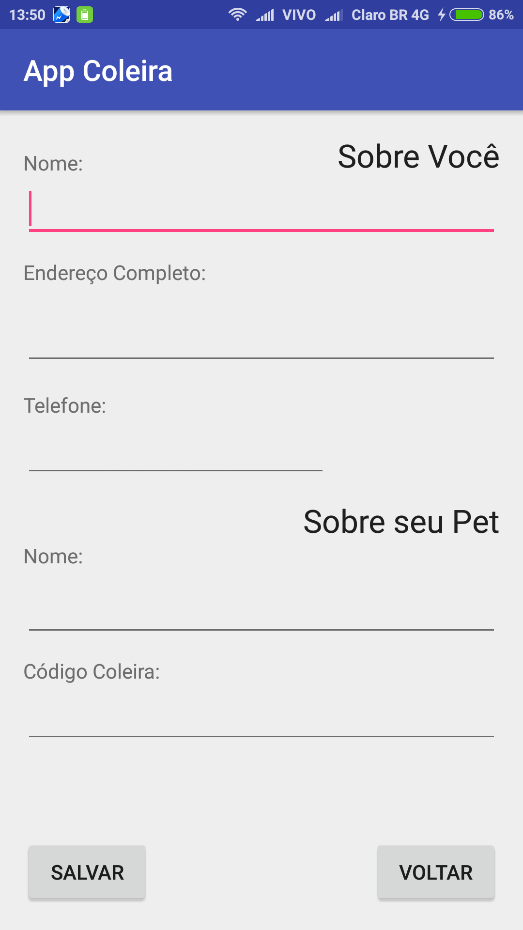
Figura 10.3 – Tela interface cadastro cliente.



(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.

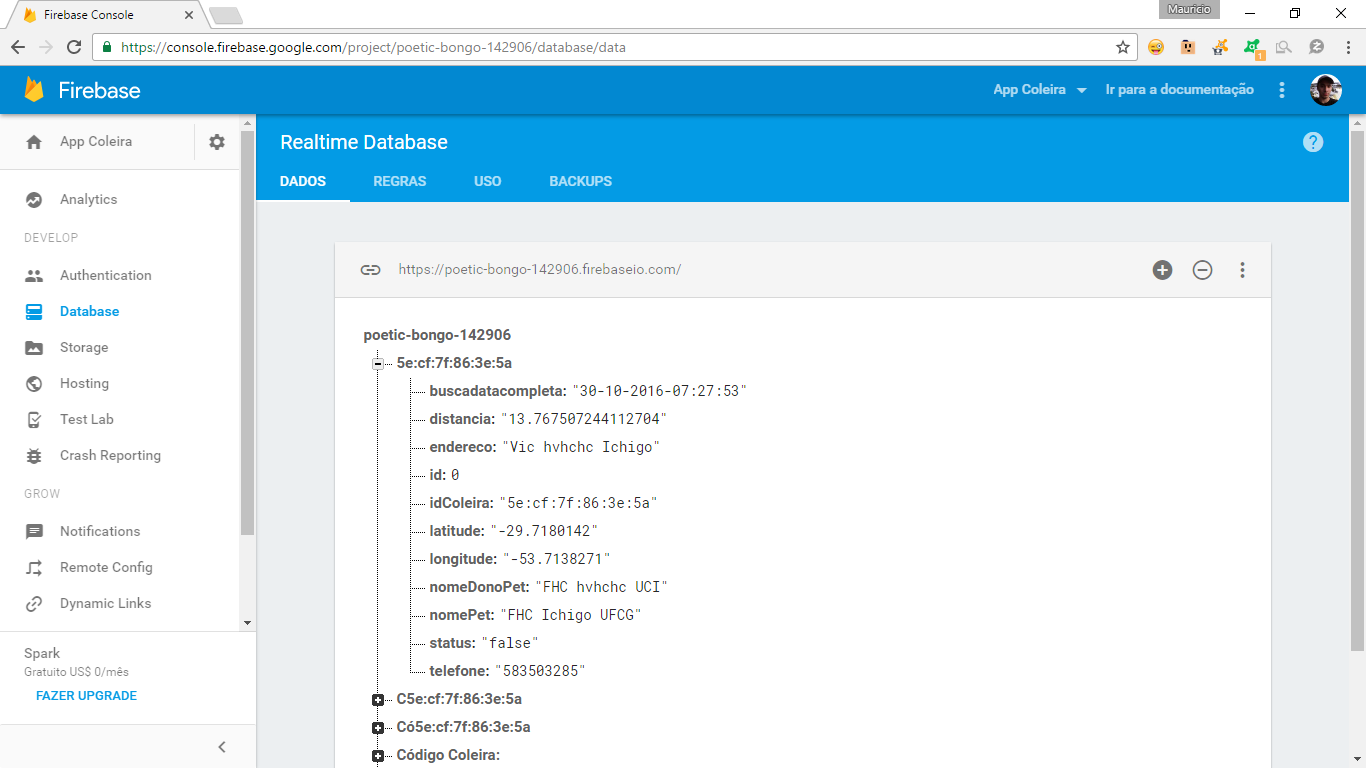
Figura 10.4 – Tela novo cadastro



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

Na imagem da Figura podemos ver os dados armazenados no Firebase Database na estrutura JSON.

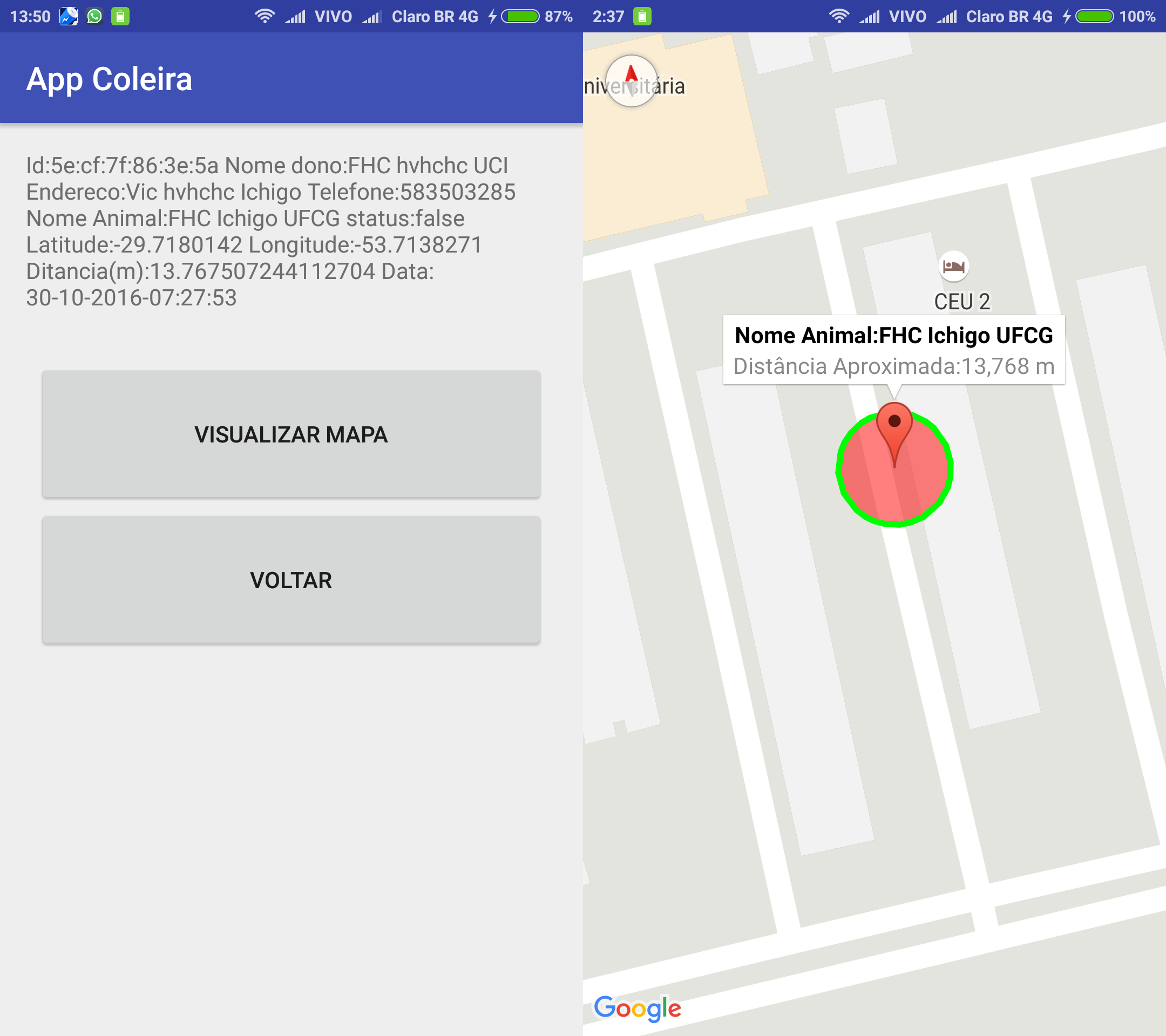
Figura 10.5 – Console Firebase Database.



(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).

Figura 10.6 – Tela informações de localização com visualização no mapa.



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

1. **RESULTADOS OBTIDOS**
   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, corresponde ao valor real. As medições foram feitas em um local com poucas redes Wi-Fi, para que não houvesse interferência nos valores estimados pelo aplicativo. A tabela apresenta os valores medidos para cada distância, com o valor médio, variância e desvio padrão calculados.

Tabela 11.1 – Valores medidos experimentalmente.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Distância (m) | Medida 1 | Medida 2 | Medida 3 | Medida 4 | Medida 5 | Medida 6 | Medida 7 | Medida 8 | Medida 9 | Medida 10 | |  | | --- | |  | | Variâncias (S^2) | Desvio Padrão (S) |
| 1 | 13,767 | 13,767 | 15,447 | 13,767 | 13,767 | 13,767 | 21,820 | 19,447 | 15,447 | 12,270 | 15,327 | 17,203 | 3,503 |
| 2 | 30,821 | 21,820 | 30,821 | 27,469 | 27,469 | 27,469 | 27,469 | 34,582 | 34,582 | 27,469 | 28,997 | 32,016 | 5,241 |
| 5 | 109,359 | 154,473 | 194,471 | 86,867 | 86,867 | 77,420 | 69,000 | 86,867 | 86,867 | 86,867 | 103,906 | 114,845 | 9,320 |
| 10 | 308,216 | 488,490 | 308,216 | 244,825 | 345,824 | 345,824 | 345,824 | 308,216 | 345,824 | 345,824 | 338,708 | 379,731 | 18,596 |
| 15 | 690,009 | 218,200 | 218,200 | 194,471 | 194,471 | 388,021 | 488,490 | 548,094 | 614,971 | 488,490 | 404,342 | 417,528 | 22,102 |
| 30 | 614,972 | 690,010 | 774,204 | 690,010 | 774,200 | 690,010 | 774,204 | 614,972 | 614,972 | 488,490 | 672,604 | 753,742 | 22,102 |
| 50 | 1.227,030 | 774,204 | 1.127,030 | 1.227,030 | 974,665 | 974,665 | 1.227,030 | 1.544,739 | 1.227,030 | 1.944,712 | 1.224,814 | 1360,658 | 44,099 |
| 60 | 1.376,750 | 1.733,226 | 1.733,226 | 1.733,226 | 1.733,226 | 1.544,740 | 1.093,592 | 1.544,740 | 1.376,751 | 1.733,226 | 1.560,270 | 1754,025 | 41,632 |
| 80 | 2.448,247 | 2.182,003 | 3.082,161 | 3.082,161 | 2.746,979 | 2.746,979 | 1.733,226 | 2.448,247 | 2.448,247 | 1.944,712 | 2.486,296 | 2766,779 | 44,099 |
| 100 | 3.082,161 | 3.082,161 | 2.746,979 | 2.746,979 | 3.082,161 | 3.082,161 | 3.082,161 | 3.082,161 | 4.393,668 | 3.082,161 | 3.146,275 | 3502,985 | 55,517 |

(Fonte: Elaborada pelo autor).

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

Figura 11.1 – 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, isso se deve a diversos fatores que podem interferir na qualidade do sinal de radiofrequência como reflexão, refração, difração e, erros de medida. Como os valores encontrados são praticamente lineares, podem ser aproximados por uma reta, 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:

(11.1)

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

(11.2)

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 Friiss para que o valor da distância calculada pelo aplicativo seja aproximadamente igual ao valor real.

(11.3)

(11.4)

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

(11.5)

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

(11.6)

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 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 11.2 – Valores médios medidos e ajustados.

|  |  |  |
| --- | --- | --- |
| Distância (m) | Valor medido médio (m) | Valor médio ajustado |
| 1 | 15,3266 | 0,999620249 |
| 2 | 28,9971 | 1,431875356 |
| 5 | 103,9058 | 3,800455005 |
| 10 | 338,7083 | 11,22480522 |
| 15 | 404,3417 | 13,30010403 |
| 30 | 672,6044 | 21,78245083 |
| 50 | 1224,8135 | 39,24305603 |
| 60 | 1560,2703 | 49,85005028 |
| 80 | 2486,2962 | 79,13057579 |
| 100 | 3146,2753 | 99,99882027 |

(Fonte: Elaborada pelo autor).

O gráfico da figura 11.2 mostra os dados médios após serem calibrados em comparação aos valores esperados.

Figura 11.2 – Valores após calibração.

(Fonte: Elaborada pelo autor).

* 1. TESTE DE BATERIA

Um teste de bateria foi realizado para se ter uma estimativa da energia consumida pelo módulo ESP8266.

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.

Em comparação ao GPS o protótipo do sistema de localização desenvolvido tem como vantagens: menor consumo de bateria, uma boa precisão 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.

**CONCLUSÕES**

# 

# **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.

**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.

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.

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 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 – 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\_item\_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);  
  
 *//Necessário gravar daddos como transação  
  
 //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);  
 }  
}

Apêndice C -