CAIO GALLO BRUNIALTI

SOLUÇÃO DE RASTREAMENTO E MONITORAMENTO DE TRANSPORTE PUBLICO

LIMEIRA
2019

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

CAIO GALLO BRUNIALTI

SOLUÇÃO DE RASTREAMENTO E MONITORAMENTO DE TRANSPORTE PÚBLICO

Monografia apresentada à Faculdade/Instituto da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Engenheiro de Telecomunicações

Supervisor/*Orientador*: Prof. Dr. Rangel Arthur

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA MONOGRAFIA DEFENDIDA PELO ALUNO CAIO GALLO BRUNIALTI, E ORIENTADO PELO PROF. DR. RANGEL ARTHUR

LIMEIRA
2019

ABSTRACT

This project consists of developing a solution for public road transport users (eg. buses and vans), and owners of these fleets. The project makes use of the Arduino Mega microcontroller board, and aims to provide a simple, low-cost operating solution for tracking, stocking and route display of the vehicle equipped with the solution in an application intended for the transport user. From the performed tests, was possible to confirm the efficiency of the proposed solution and its accuracy in performing its tasks, besides, was also possible to bring new improvements to its functioning, which can increase its attractiveness.

RESUMO

Este projeto consiste em desenvolver uma solução destinada aos usuários de transporte público rodoviário (Ex: ônibus e vans), e aos proprietários destas frotas. O projeto faz uso da placa de desenvolvimento Arduino Mega, e tem como objetivo fornecer uma solução simples e de baixo custo operacional de rastreamento, cálculo de lotação e exibição de rota do veículo equipado com a solução em um aplicativo destinado ao usuário do transporte. A partir dos testes realizados, pode-se confirmar a eficiência da solução proposta e sua precisão na execução de suas funções, além de se levantar melhorias pertinentes para a melhoria do funcionamento da mesma, o que pode aumentar sua atratividade.

SUMÁRIO

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas, que de uma forma ou de outra, tornaram possível a realização deste trabalho.

Ao meu pai, Sr. Oscar Antonio, que durante todo o meu curso sempre me motivou nos momentos onde pensei em desistir, e que sempre me motivou a me superar.

Ao meu irmão Murilo, que sempre me motivou durante os estudos, e que por diversas vezes se mostrou curioso em relação ao que estava aprendendo, bem como ter tecido criticas pertinentes aos meus estudos.

Ao meu primo Renato, que durante o tempo onde trabalhou no metro da cidade de São Paulo, me inspirou na escolha de um tema para a realização deste trabalho.

Por fim, ao meu orientador Rangel Arthur, que se mostrou receptivo em me orientar para a realização deste projeto, bem como em despertar meu interesse em automação através da disciplina de Automação e Controle ministrada por ele em 2017.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

A: símbolo de Ampére (unidade de corrente elétrica).

A-GPS: abreviação de Assisted Global Positioning System (ou em português, Sistema de Posicionamento Global Assistido).

CPU: abreviação de Central Processing Unit (ou em português, unidade central de processamento).

EDGE: abreviação de Enhanced Data GSM Enverinment.

HSDPA: abreviação de High Speed Downlink Packet Access.

IoT: abreviação de internet of things (ou em português, internet das coisas).

IDE: abreviação de Integrated Development Environment (ou em português, ambiente de desenvolvimento integrado).

IEEE: abreviação de Institute of Electrical and Electronic Engineers (ou em português, instituto de engenheiros eletricistas e eletrônicos).

GND: abreviação de ground (ou em português, terra).

GPS: abreviação de Global Positioning System (ou em português, Sistema de Posicionamento Global).

GPRS: abreviação de General Packet Radio Service.

GSM: abreviação de Global System for Mobile Communications.

mA: abreviação de miliampere.

RAM: abreviação de random access memory (ou em português, memória de acesso aleatório).

RISC: abreviação de Reduced Instruction Set Computer (ou em português, computador com um conjunto reduzido de instruções).

ROM: abreviação de read-only memory (ou em português, memória somente de leitura).

SMA: abreviação de Sub Miniature Version A (ou em português, sub-minhatura versão A).

SMS: abreviação de Short Message Service.

V: símbolo de Volt (unidade de tensão elétrica).

VCC: abreviação de voltagem em corrente continua.

1. Introdução

Meio de transporte surgido em 1895 na Alemanha, o primeiro ônibus movido à combustão teve seu inicio no Brasil no ano de 1908 na cidade do Rio de Janeiro. Até esta data, o principal meio de transporte de massas nos grandes centros era o bonde, porém, pelo fato do ônibus apresentar diversas vantagens em relação a esse, como uma maior velocidade de deslocamento, uma maior facilidade de acesso a mais localidades (visto que não necessita de instalação de trilhos), e maior flexibilidade de horários, fez com que o ônibus se tornasse preferível em relação ao bonde, e o substituísse gradualmente [1].

De acordo com dados da ANTP (Associação Nacional de Transportes Públicos) [2], com informações coletadas das cidades de São Paulo, Rio de Janeiro, Belo Horizonte, Recife, Porto Alegre, Salvador, Curitiba e Goiânia, foi possível elaborar a Tabela 1, com informações referentes ao numero médio de passageiros de ônibus por mês em cada ano no período de 1994 até 2010, bem como a frota destes veículos disponíveis nas cidades analisadas.

Tabela 1: dados referentes ao numero de passageiros por mês, bem como a frota nas cidades de São Paulo, Rio de Janeiro, Belo Horizonte, Recife, Porto Alegre, Salvador, Curitiba e Goiânia. [2]

Ano	Passageiros (milhões) / mês	Frota	Ano	Passageiros (milhões) / mês	Frota
1994	442,7	28163	2003	308,5	28493
1995	473,7	30937	2004	306,2	30309
1996	459,5	31871	2005	308,6	30036
1997	451,6	31534	2006	315,8	30186
1998	416,9	32625	2007	341,7	29272
1999	367,8	32857	2008	336,3	30122
2000	348,8	32063	2009	331,4	32673
2001	337,4	30794	2010	329,1	32524
2002	351,9	31416			

Pela Tabela 1, é possível notar uma queda do numero de usuários de ônibus a partir do ano de 1996. Isto tem ocorrido por diversos fatores: aumento da frota de automóveis no período analisado, adoção por parte da população a meios de transportes alternativos (como bicicletas, por exemplo), e mais recentemente, pelo advento dos aplicativos de transporte (ao exemplo do Uber).

A migração dos usuários do ônibus para os outros meio citados acima ocorreu por diversos fatores, entre os quais, a alta lotação deste transporte no ambiente urbano, aos recorrentes atrasos enfrentados pelos usuários, a péssima qualidade de conservação de alguns veículos e a falta ou numero insuficiente de linhas que atendam regiões periféricas e distantes destas cidades. [3]

A queda do número de passageiros tem causado diversos transtornos para as empresas de ônibus que operam nestes locais (que apresentaram queda no seu faturamento), bem como para os municípios afetados (que passaram a enfrentar maiores problemas com o excesso de automóveis, bem como uma necessidade de mais subsídios para as empresas de ônibus).

Este trabalho busca apresentar uma solução (ou um atenuante) para duas das principais queixas apresentadas pelos usuários: lotação excessiva e atraso dos veículos.

Este projeto visa desenvolver uma solução que permita ao passageiro localizar o(s) ônibus de uma determinada linha de interesse, consultar seus pontos de parada, bem como sua lotação, através de um aplicativo disponibilizado gratuitamente.

Embora já existam soluções parecidas na questão de rastreamento de veículos de transporte publico [4-10], este projeto inova em focar em uma solução de baixo custo (com custo de implementação inferior a R$800,00 por veiculo, e custo de operação inferior a R$40,00 por mês, com o hardware baseado no uso de placas de desenvolvimento da família Arduino), e que não se limite a apenas rastrear o veiculo, mas sim em fornecer sua localização ao usuário, e também fornecer sua lotação. Esta solução também é interessante para os proprietários destas frotas, visto que com os dados obtidos com o seu uso, é possível alocar mais veículos para as rotas que se mostrarem com maior ocupação dos veículos, bem como maior atraso.

Este trabalho está organizado da seguinte forma: inicialmente, realiza-se uma revisão teórica acerca dos tópicos pertinentes para a compreensão dos componentes utilizados, bem como nos

seus princípios de funcionamento. Em seguida é realizada uma abordagem prática, onde são mostrados ao leitor os matérias e equipamentos utilizados para a realização deste projeto, bem como sua metodologia. Seguida desta, são mostrados os testes realizados para comprovar a eficiência da solução, bem como para listar suas limitações e possíveis melhorias.

Por fim, são mostradas as conclusões, que possuem como objetivo reforçar o objetivo do trabalho, enfatizar a importância da técnica proposta, e comentar sobre as dificuldades encontradas. Também é comentado sobre a possibilidade de patente ou comercialização da solução proposta, e listado os possíveis trabalhos futuros como continuidade deste trabalho.

2. Revisão Teórica

2.1 GPS

GPS é a abreviação de *Global Positioning System*, ou Sistema de Posicionamento Global, é um sistema de radio navegação baseado em satélites, operado pelo governo dos Estados Unidos da América. Na atual data (Setembro de 2019) há 31 satélites em órbita dedicados a esta função, a uma altitude média de 20.350 Km de altitude em relação a superfície da Terra cada um.

Este sistema de radio navegação funciona da seguinte maneira: cada satélite transmite seu sinal em *broadcasting* com as informações de sua localização, bem como a hora exata em que este sinal foi transmitido (de fato, estes satélites possuem relógios atômicos em seu interior, sendo usados também para o sincronismo de relógios). Ao chegar no receptor, este calcula a diferença entre momento em que o sinal foi enviado até ele ter sido recebido (a velocidade de propagação do sinal é conhecida pelo receptor), e com isso calcula sua distancia até o satélite. Para calcular sua localização exata, é necessário o receptor estar recebendo o sinal proveniente de ao menos 4 satélites diferentes. A partir dai, o receptor consegue calcular sua localização tridimensional (latitude, longitude e altitude) através de geometria.

Este sistema de navegação foi declarado operacional em 1995, e atualmente está presente nas mais diversas aplicações, sejam elas de cunho civil, ou militares [11]. Devido a sua grande difusão e facilidade de uso, foi este o sistema de radio navegação utilizado para a realização deste trabalho. Na Figura 1 é ilustrado o satélite GPS-2F 2 (Navstar 63) de fabricação estadunidense, lançado no ano de 2011 com vida útil de 12 anos [12-13], que é um dos satélites responsáveis pelo funcionamento do Sistema de Posicionamento Global (GPS).

Figura 1: satélite GPS-2F 2 (Navstar 63). [13]

2.2 Lei de Hooke

Lei criada em homenagem a Robert Hooke, quem primeiro a demonstrou em 1660, esta lei relaciona a força relacionada para se deformar uma mola com sua deformação. De maneira sucinta, esta lei diz que a força necessária para se deformar uma mola é diretamente proporcional à deformação sofrida por esta. Matematicamente, esta lei é descrita da forma como está na Equação I:

$$F = -Kx \qquad \text{(Equação I)}$$

Onde:

F = força necessária para se deformar a mola;

K = constante elástica da mola (N/m);

x = deformação total sofrida pela mola.

O sinal negativo representa que força aplicada para se deformar a mola é contrario a força imposta pela mesma para retornar ao seu estado sem deformidade. [14]

Embora esta lei seja elementar, é de extrema importância para a compreensão deste trabalho, uma vez que nele ira ser considerado que cada uma das molas presentes na suspenção do veiculo equipado com a solução apresentará este efeito, ou seja, elas serão comprimidas com uma taxa diretamente proporcional a carga suportada por elas, e, que quanto menor a distancia do veiculo com o solo, maior a carga levada pelo veiculo.

2.3 Microcontroladores

Entende-se como microcontrolador todo chip que possua ROM, RAM e CPU, assim como qualquer processador, porém, com menor capacidade de processamento, sendo destinados a tarefas especificas. Este tipo de chip teve suas origens na década de 1970, com o primeiro dispositivo equipado com um chip microcontrolador sendo a calculadora TMS 1802, criado pela *Texas Instruments*, equipada com um chip TMS1000 (ilustrado na Figura 2). [15]

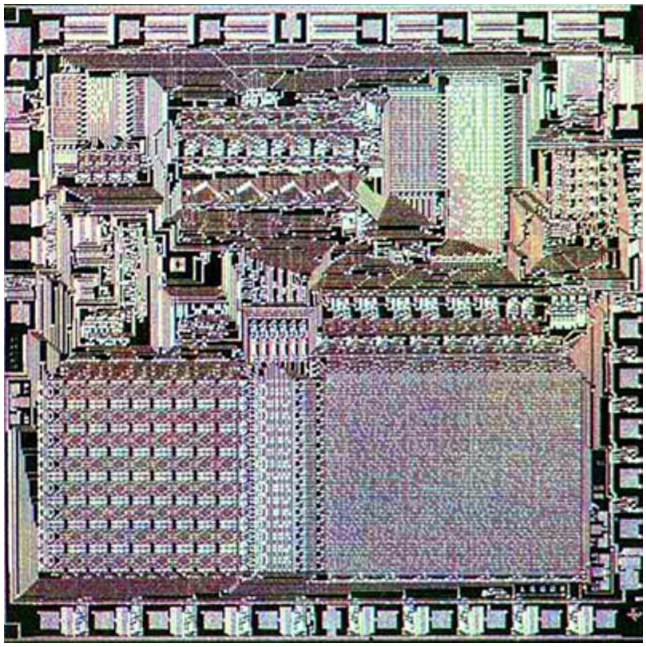

Figura 2: chip TMS1000 sem o seu encapsulamento. [15]

Inicialmente, concebidos utilizando arquitetura 4 bits, foram criados com o intuito de equipar pequenas aplicações, como por exemplo, calculadoras, brinquedos e até fornos de micro-ondas. Atualmente, com avanço de eletrônica, e o surgimento dos microcontroladores de 32 e 64 bits, este tipo de chip é utilizado em uma infinidade de aplicações, incluindo aplicações em automóveis, em automação fabril, etc.

O chip microcontrolador que equipa a placa de desenvolvimento usada neste projeto é o Atmel ATmega2560, que possui como característica, entre outros itens, arquitetura RISC de 8 bits, memória FLASH de 256KB, 8KB de SRAM interna, 2 contadores/timers de 8 bits, 4 contadores/timers de 16 bits, RTC com oscilador próprio e 4 canais PWM de 8 bits. [16]

Decidiu-se por optar o uso de uma placa de desenvolvimento com este microcontrolador devido a seu excelente desempenho e resolução suficiente para a aplicação, e pelo seu baixo custo de aquisição, além da abundante quantidade de periféricos e sensores disponíveis para a placa de desenvolvimento (Arduino Mega).

2.3 Ultrassom

Entende-se como ultrassom, toda frequência de áudio superior a 20 KHz (como comparação, a audição humana, na maioria dos casos, consegue detectar sons com essa frequência máxima). Os estudos com essa faixa de frequência sonora se iniciaram na década de 1880 com os físicos

franceses Pierre Curie e Paul-Jacques Curie, que descobriram o efeito piezoelétrico em certos cristais, porém somente na primeira guerra mundial, seu uso para ecolocalização teve inicio. [17]

A ecolocalização é a capacidade de se determinar distancias através da emissão de ondas sonoras na faixa do ultrassom. Neste mecanismo, as ondas emitidas se propagam no ambiente, refletem em um objeto, e retornam até sua fonte (seja esta um transceptor eletrônico, ou até mesmo um animal que faz uso deste mecanismo, como é o caso do morcego) [18]. Como a velocidade de propagação da onda sonora no meio é conhecida pelo emissor, multiplica-se a velocidade da onda pelo tempo entre sua emissão e recepção, divide-se o resultado por 2, e assim se obtém a distancia da fonte até o objeto.

Neste projeto, escolheu-se pela ecolocalização como ferramenta para calcular a distancia entre o veiculo e o solo devido a sua facilidade de uso, baixo custo dos sensores equipados com esta tecnologia, boa precisão, bem como sua abundancia de sensores para a placa de desenvolvimento utilizada (Arduino Mega).

3 Materiais e Metodologia

3.1 Materiais

Para a realização deste trabalho, foram usados os seguintes materiais:

- Adaptador RP-SMA para SMA;
- Antena para GPS com conector SMA;
- Cabo uFL para RP-SMA;
- Chip da operadora Vivo;
- Fonte chaveada 12V/3A;
- Inversor 12V para 110V automotivo;
- Placa de desenvolvimento Arduino Mega;
- Sensor ultrassônico HC-SR04;
- Shield TinySine 3G/GPRS/GSM/GPS;
- Materiais diversos, entre os quais, fios, fita isolante, caixa plástica de montagem e cabo USB para computador;
- Veiculo de uso rodoviário com as especificações mínimas necessárias.

Para se realizar a programação da placa Arduino Mega, utilizou-se a IDE do Arduino. A plataforma utilizada para o recebimento dos dados enviados via rede móvel foi a dweet.io. Por fim, para a programação do aplicativo do usuário, utilizou-se a plataforma Freeboard.io (que recebia as informações do dweet.io, e com isso já plotava as coordenadas do GPS em uma mapa, além de criar espaços destinados a exibição da altitude do veiculo, bem como sua velocidade e lotação) em conjunto com MIT AppInventor, sendo este ultimo responsável pela criação do aplicativo do usuário.

PLACA DE DESENVOLVIMENTO ARDUINO MEGA

O projeto Arduino, de origem italiana, nada mais é do que uma plataforma de hardware *open-source* baseada em hardwares e softwares de uso simplificado. A ideia por trás da concepção deste projeto foi criar uma plataforma que fosse de fácil manuseio por iniciantes, mas que oferecesse flexibilidade e recursos para usuários avançados. Outras características desta plataforma incluem o baixo custo das placas de desenvolvimento, o suporte a diversos sistemas operacionais (o ambiente de desenvolvimento do Arduino esta disponível para Windows, Macintosh e diversas distribuições de Linux), hardware *open-source* que possibilita ao usuário sua reprodução e modificação sem problemas relacionados a licenças, e que facilita a criação de novos periféricos por terceiros que podem acrescentar novas funcionalidades e possibilidades as placas originais, etc. [19]

Todos estes fatores foram decisivos na escolha de uma placa de desenvolvimento Arduino para este trabalho. A escolha da placa Arduino Mega em detrimento a outros modelos (em especial ao Arduino Uno e ao Arduino Leonardo) ocorreu por diversos fatores, entre os quais:

- Maior memória SRAM disponível (enquanto o Arduino Uno possui 2KB de SRAM, e o Arduino Leonardo 2,5KB de SRAM, o Arduino Mega possui 8KB de memória SRAM, o que permite maior tranquilidade no desenvolvimento de programa utilizado pela solução);
- Maior memória FLASH para armazenamento de programa disponível (enquanto o Arduino Uno e o Arduino Leonardo possuem 32KB de memória FLASH, o Arduino Mega possui 256KB de memória FLASH, o que também permite maior tranquilidade no desenvolvimento de programa utilizado pela solução);
- Maior quantidade de entradas disponíveis e saídas disponíveis (enquanto o Arduino Uno possui 14 pinos de entrada e saída digitais, e o Arduino Leonardo possui 20 pinos de entrada e saída digitais, o Arduino Mega, possui 54 pinos de entrada e saída digital).

As características do hardware do Arduino Mega, além das já citadas, incluem: um microcontrolador ATmega2560, tensão de operação de 5V, corrente máxima de 20 mA por pino digital, clock de 16 MHz, 16 pinos de entrada analógica, tensão de entrada por fonte externa recomenda de 7-12V e 15 pinos digitais que fornecem saída PWM. Na Figura 3 é mostrada a vista superior desta placa de desenvolvimento, já na Figura 4 é mostrada a sua vista inferior. [20][21-22]

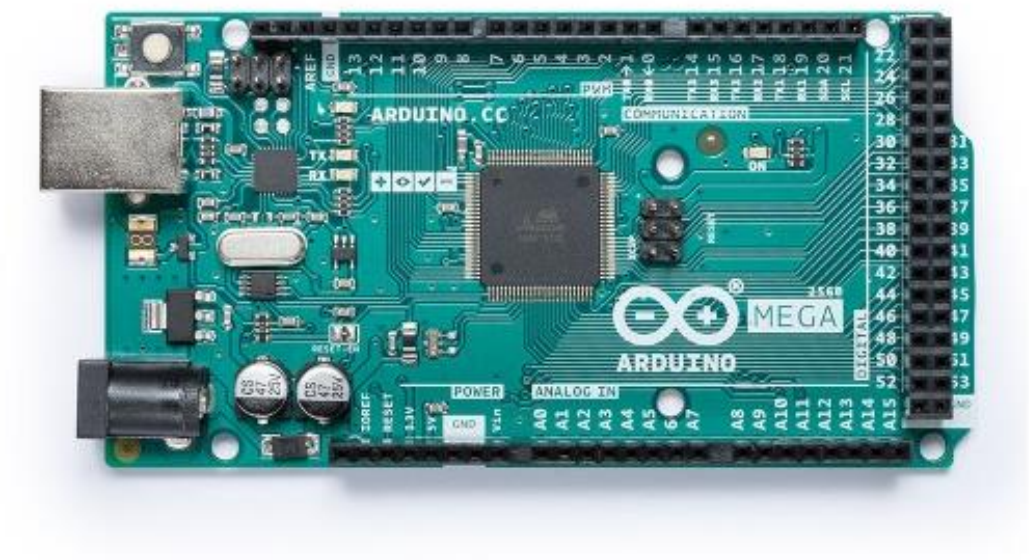

Figura 3: vista superior de uma placa de desenvolvimento Arduino Mega. [20]

Figura 4: vista inferior de uma placa de desenvolvimento Arduino Mega. [20]

SENSOR ULTRASSONICO HC-SR04

Este sensor é destinado para a medida de distancias numa faixa de 2 centímetros até 4 metros, com precisão de 3 milímetros. Ele faz uso de uma frequência sonora de 40 KHz (faixa localizada no ultrassom) para realizar as medições: inicialmente, este sensor emite um sinal sonoro de duração de 10 microssegundos, e caso haja algum objeto na sua distancia de operação, este sinal sonoro ira se refletir no objeto e retornar até o sensor, que com base na diferença de tempo entre este sinal transmitido e recebido, bem como na sua potencia recebida, consegue calcular a distancia do objeto em questão. Este sensor opera com uma tensão de 5V (padrão TTL) com uma corrente de 15 mA, e possui um total de 4 pinos [23]:

- VCC
- GND
- TRIG
- ECHO

O pino VCC é conectado a uma fonte de 5V, sendo responsável pela alimentação do sensor, assim como o pino GND, que dever ser ligado ao terra da fonte de alimentação, ou ao terra da placa que faz uso deste sensor. O pino TRIG tem a função de regular a saídas dos pulsos de som, e o pino ECHO envia os dados pertinentes sobre o seu retorno (como o tempo que o som levou para se propagar a partir do sensor e retornar a ele, bem com sua potencia na recepção) para o microcontrolador. A Figura 5 apresenta as vistas superior e inferior deste sensor.

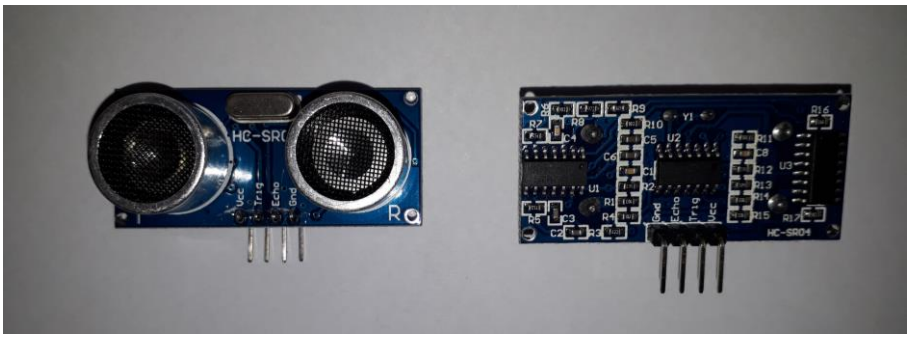

Figura 5: vistas superior e inferior do sensor HC-SR04.

SHIELD TINYSINE 3G/GPRS/GSM/GPS SIM5320E

Esta *shield* foi responsável por fornecer o serviço de GPS, bem como o serviço de rede móvel para o envio de dados da solução. De fabricação chinesa pela companhia *TinySine*, esta *shield* é baseada no chip SIM5320E (versão destinada ao mercado europeu, mas que funciona no Brasil) da companhia SIMCom, que entre outras características, possui suporte a tecnologia GSM (850/900/1800/1900 MHz), WCDMA (900/2100 MHz) e HSDPA (900/2100 MHz).

Em relação ao receptor de GPS contido nesta *shield*, este possui uma sensibilidade de rastreamento de -157dBm, uma precisão de 2 metros, um receptor de até 16 canais, de frequência 1575.42 MHz, com atualização da posição a cada 1 segundo. Também há suporte ao A-GPS (ou GPS assistido, onde os dados recebidos via satélite são complementados com os dados recebidos via rede móvel de dados) [24]. Para se fazer uso do GPS, conectou-se uma adaptador com entrada uFL e saída RP-SMA na *shield*, em seguida conectou-se um conversor RP-SMA na saída do adaptador, e por fim, conectou-se uma antena com entrada SMA no outro lado do conversor.

Entre as outras características pertinentes desta *shield*, pode-se citar: necessidade de uma fonte de alimentação externa de 9V com pelo menos 2A que deve ser conectado ao Arduino (a pinagem desta *shield* permite que ela seja encaixada diretamente sobre os Arduinos Uno e Mega), entrada para microfone e saída para fone de ouvido, capacidade de recepção e envio de SMS, capacidade para receber/realizar chamadas de voz, suporte as interfaces USB 2.0, UART, SPI, I2C, PCM, cartão SIM, suporte aos protocolos FTP, FTPS, HTTP, HTTPS, SMTP, POP3, DNS, entre outras características. [24-25]

Esta *shield* foi escolhida devido a sua grande quantidade de recursos, pelo fato dela oferecer suporte a tecnologia GPS e rede móvel numa mesma placa, diminuindo o numero de periféricos que precisariam estar conectados ao Arduino Mega utilizado, e pela facilidade na integração com o Arduino Mega: esta *shield* é encaixada diretamente sobre a placa, deixando o conjunto mais compacto.

Esta *shield* está ilustrada na Figura 6 (vista inferior) e na Figura 7 (vista superior).

Figura 6: vista inferior da *shield TinySiny* 3G/GSM/GPRS/GPS SIM5320E [26]

Figura 7: vista superior da *shield TinySiny* 3G/GSM/GPRS/GPS SIM5320E [26]

IDE ARDUINO

Ambiente de desenvolvimento disponível de forma gratuita e *open source*, é utilizado para a programação de todas as placas de desenvolvimento Arduino, sejam elas genuínas, ou clones. Este ambiente faz uso das linguagens de programação C/C++, e disponibiliza uma série de opções para o utilizador: possibilita a importação de bibliotecas desenvolvidas por terceiros, possibilita a seleção da porta serial que será utilizada na comunicação entre o computador e a placa de desenvolvimento, permite obter informações sobre a placa utilizada (como seu modelo e numero de série), possibilita acessar o monitor serial onde os dados obtidos pela placa são exibidos, entre outras possibilidades. A Figura 8 ilustra a tela inicial deste ambiente de desenvolvimento. [27-28]

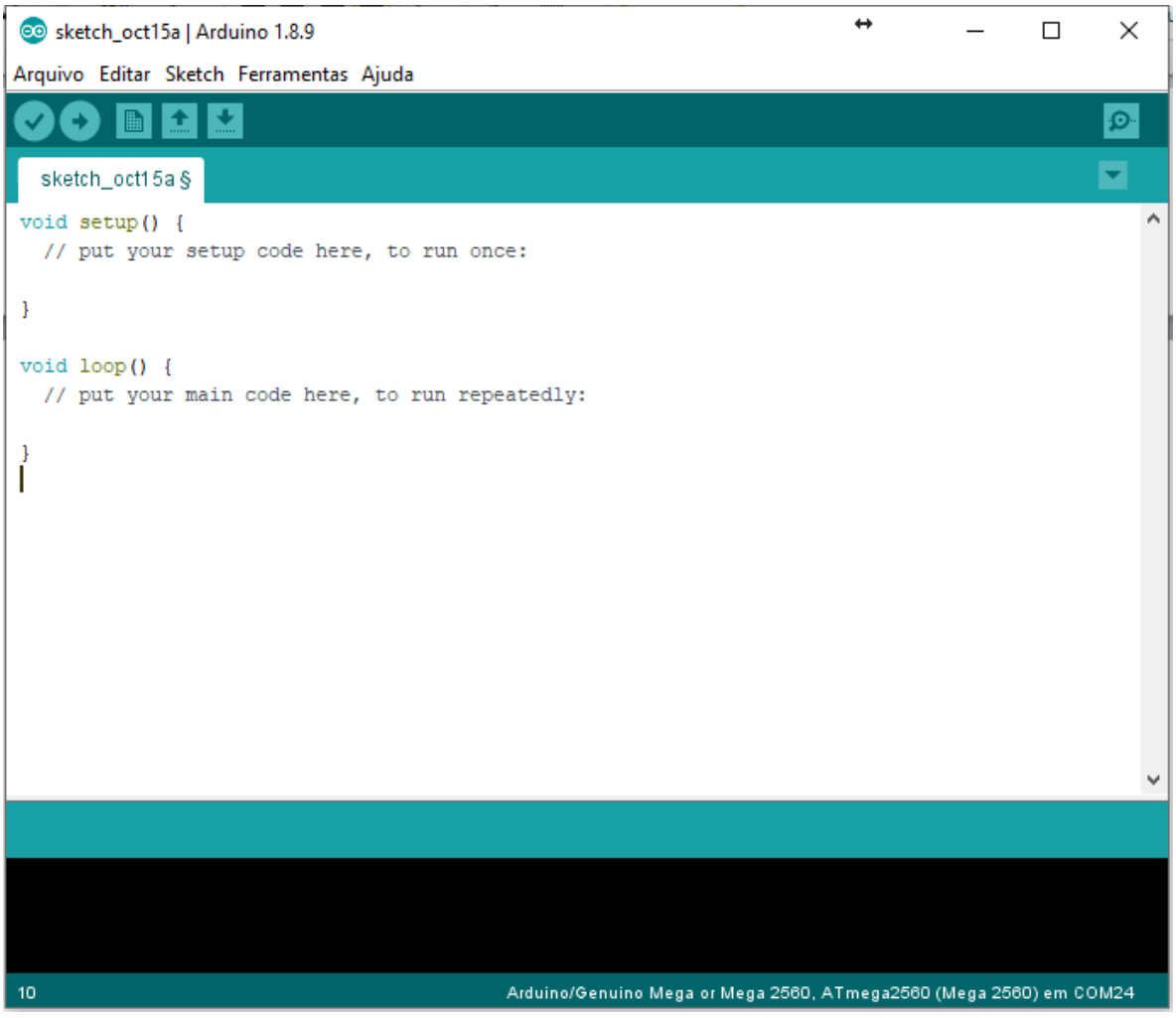

Figura 8: tela inicial do ambiente de desenvolvinento Arduino.

PLATAFORMA DWEET.IO

Plataforma web criada pela companhia Bug Labs inc, tem como objetivo ser uma nuvem destinada exclusivamente ao recebimento de dados enviados por sensores, maquinas, dispositivos e robôs conectados a web, e usados em projetos de IoT. Esta plataforma, usada na opção gratuita, armazena os últimos 5 dados enviados por cada dispositivo num período de até 24 horas (após este período todos os dados são apagados, ou no caso de mais de 5 dados serem enviados em um período inferior a 24 horas, os mais antigos são excluídos para dar espaço aos mais novos). [29]

Pela facilidade na sua utilização, bem como pela pouca quantidade de dados necessários para o seu correto funcionamento. Na Figura 9 são mostrados os dados coletados e enviados pela solução (dados de latitude, longitude, velocidade, altitude e lotação) e exibidos pelo dweet.io.

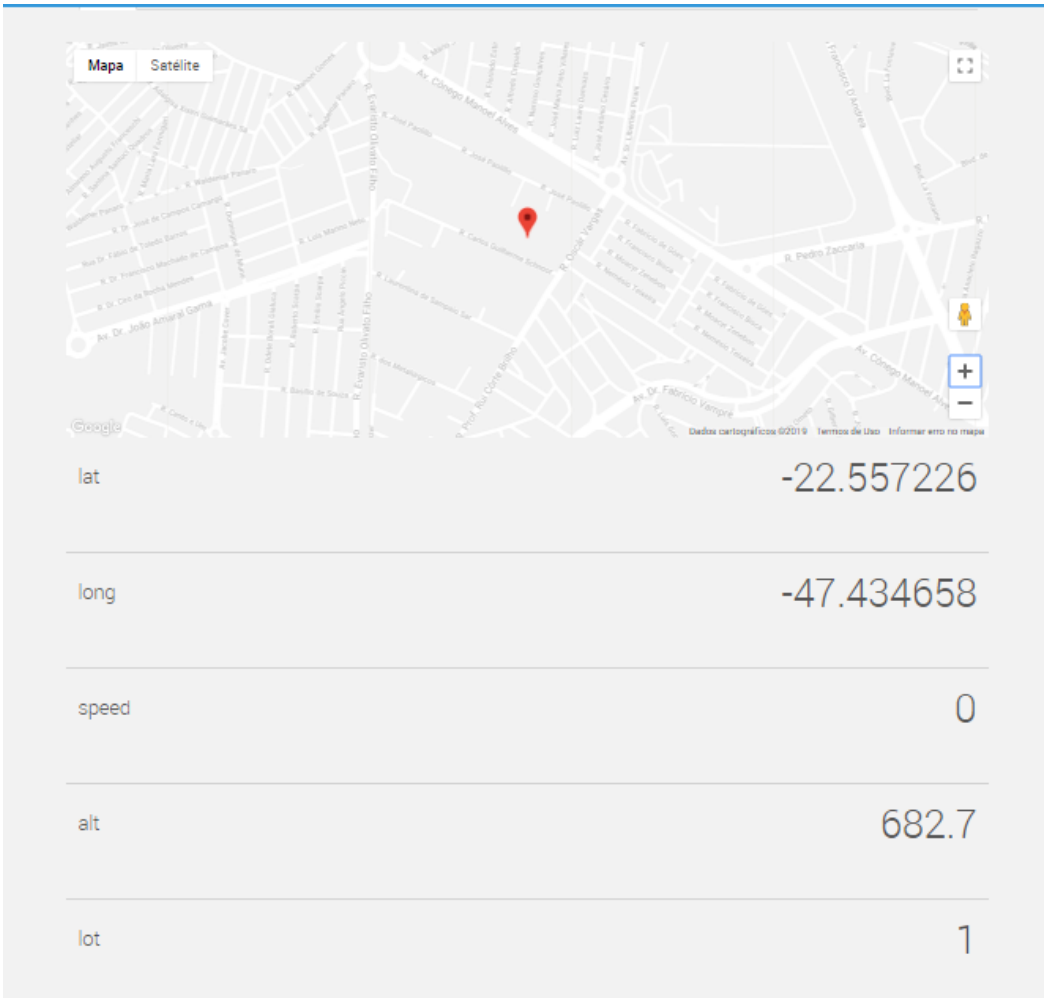

Figura 9: dados provenientes da solução compilados e ilustrados no dweet.io.

PLATAFORMA FREEBOARD.IO

A plataforma freeboard.io é utilizado para o tratamento dos dados recebidos pelo dweet.io (de fato, os dois foram criados pelo mesma companhia), porém, também oferece suporte a outras nuvens. Nesta plataforma, todas as informações são exibidas em um painel, e existem diversos templates já prontos onde o usuário pode tratar da maneira que julgar mais pertinentes os dados coletados por seu equipamento. Existem mapas pré-configurados que podem ser utilizados para plotar a latitude e longitude coletadas, existem escalas (como a utilizada pela lotação e pelo velocímetro neste trabalho, mas também pode ser utilizada para uma infinidade de opções) pré-prontas onde o usuário pode plotar os dados de maneira gráfica e não somente em texto. Também existe a opção de adicionar imagens para este painel, códigos em HTML e textos. [30]

Neste trabalho, os dados referentes a latitude e longitude coletados pelo GPS da shield são convertidos para um ponto no mapa, e os dados referentes a lotação e a velocidade são convertidos para uma escala com suas respectivas unidades de medida, e a altitude é exibida numa caixa especifica, como ilustrado na Figura 10.

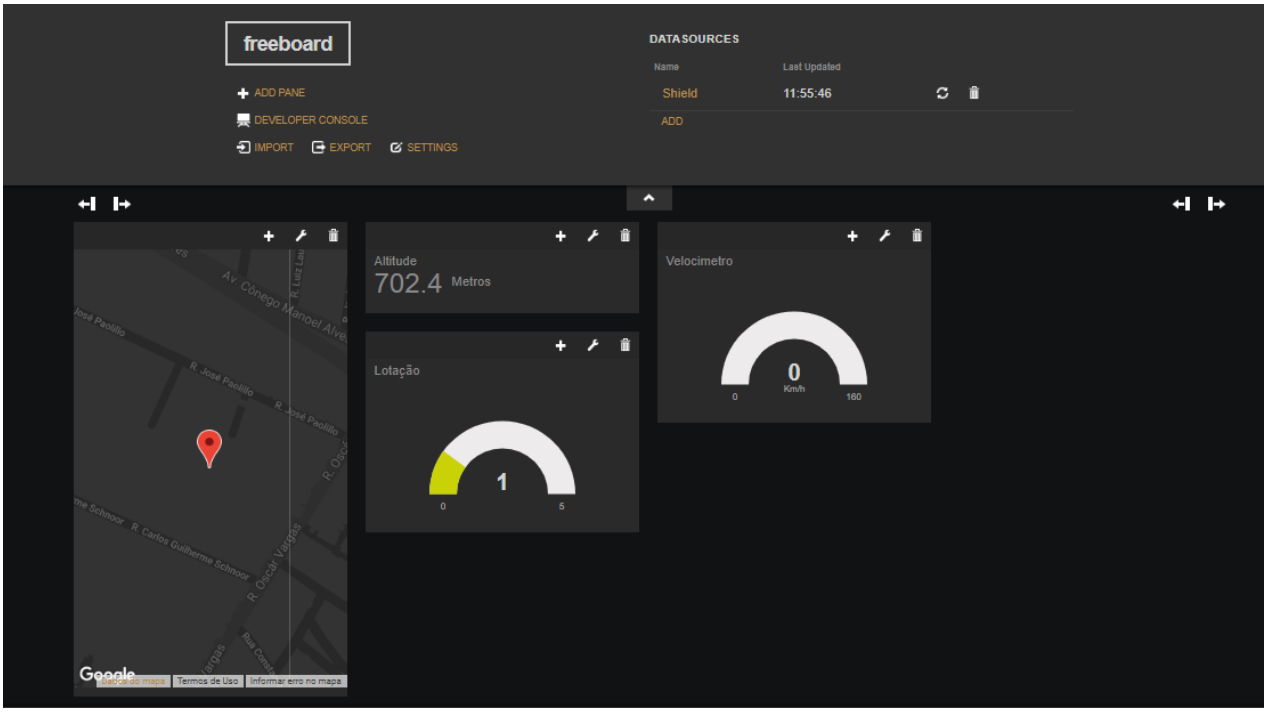

Figura 10: vista do painel de controle onde são recebidas e processadas as informações de localização, velocidade, altitude e lotaçao do veiculo.

MIT APP INVENTOR

O MIT App Inventor é um ambiente de programação simplificado, desenvolvido e mantido pelo Instituto de Tecnologia de Massachusetts (ou em inglês, Massachussets Institute of Technology - MIT), e concebido para a construção de aplicativos para smartphones e tablets. [31]

Este ambiente foi criado com o objetivo de popularizar e democratizar o desenvolvimento de software para pessoas com pouco conhecimento em programação. A sua programação é baseada no uso de blocos, onde o usuário modela o seu aplicativo, sendo ofertada uma ampla de recursos para o utilizador: é possível adicionar imagens ao aplicativo, criar botões, listas, tabelas, adicionar paginas web, mapas, arquivos de áudio, customizar o layout, adicionar o uso de sensores (como acelerômetros e GPS), entre outras diversas opções.

Escolheu-se o uso deste ambiente de programação para a criação do aplicativo do usuário devido a todas as suas qualidades já citadas: facilidade de uso, grande quantidade de recursos disponíveis, além do fato de ser disponível de maneira gratuita, e contar com uma grande comunidade utilizadora ao redor do mundo.

Tomando como exemplo o aplicativo desenvolvido para a solução proposta neste trabalho, na Figura 11 é mostrada a tela inicial do aplicativo do usuário, onde há o botão para o mesmo acessar a lista de linhas de transporte disponíveis no aplicativo, enquanto que na Figura 12 é mostrado os blocos que modelam o funcionamento do botão quando ele é acionado (neste caso, abrir a pagina do aplicativo onde estão todas as listas disponíveis).

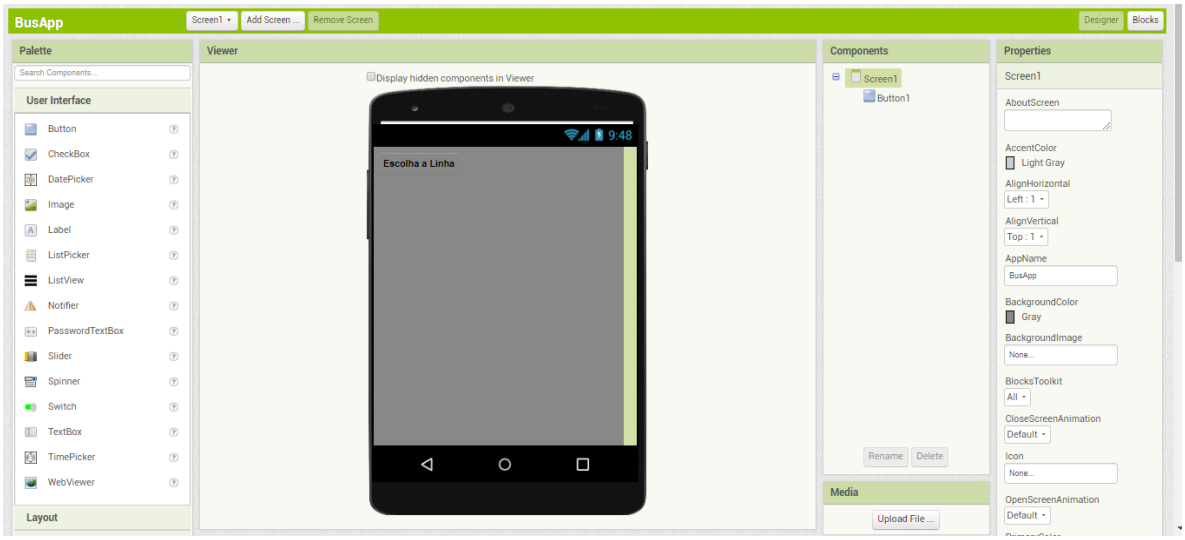

Figura 11: tela inicial do aplicativo do usuario no MIT App Inventor.

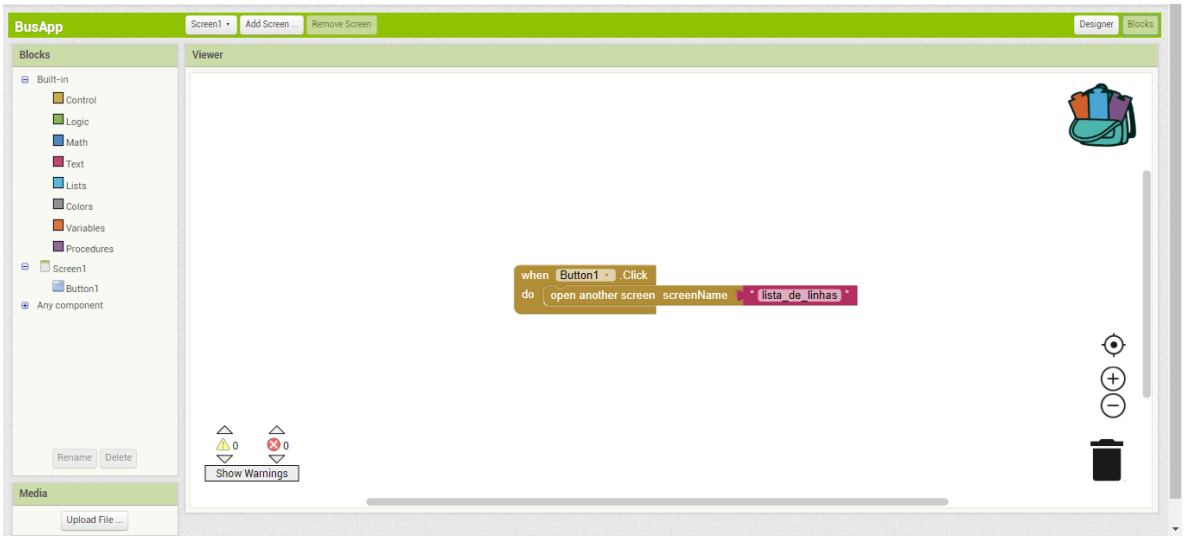

Figura 12: blocos referentes a tela inicial do aplicativo do usuario no MIT App Inventor.

GOOGLE MAPS

O Google Maps é um serviço de mapas, localização e imagens de satélite criado pela Google. Atualmente disponivel para web, iOS e Android, esta ferramenta é de uso gratuito, e oferece ao usuario uma ampla gama de recursos, entre as quais, navegação, inserção de pontos de interesse e comércios, adição de pontos turisticos no mapa, ferramentas de monitoramento de transito, entre outras opções. [32]

Este serviço também ofere ao usuario a opção de criar mapas personalizados para sua utilização, e entre as opções de customização disponiveis, é possivel adicionar marcadores no mapa com descrição, adicionar rotas pré definidas, medir distancias e customizar o estilo do mapa usado. Na figura 13 é ilustrado uma mapa criado com o auxilio desta ferramenta para uma linha de transporte atendida pela solução.

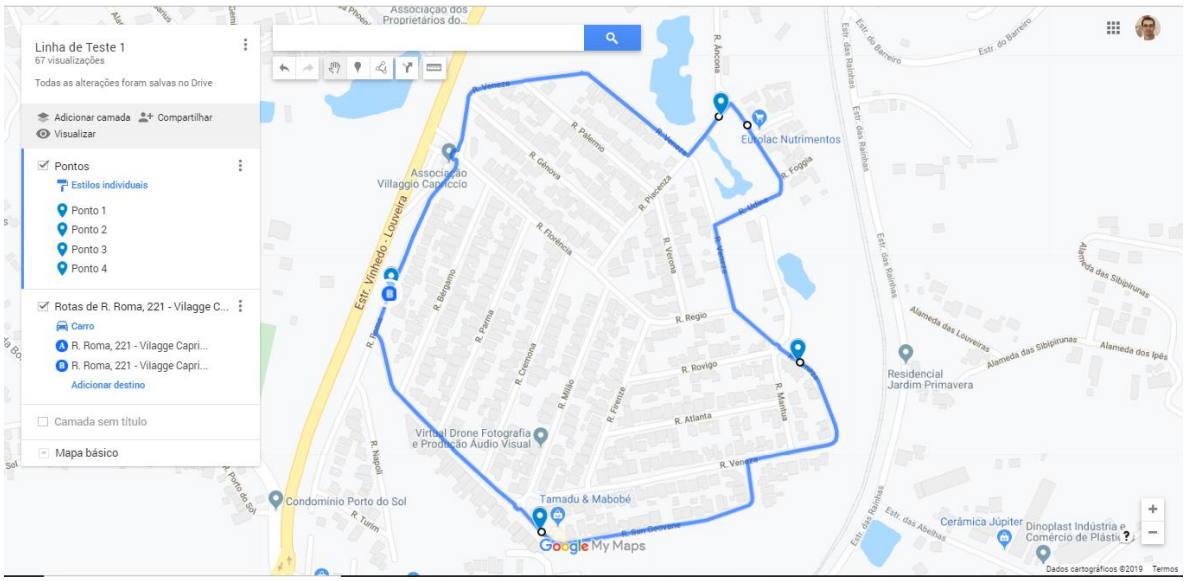

Figura 13: Mapa customizado criado com o uso do Google Maps contendo a rota utilizada em uma linha de teste, bem como seus pontos de parada.

Optou-se pelo uso deste serviço de mapas em detrimento a outros devido a sua gratuidade, facilidade de uso, possibilidade de criação de mapas customizados, e por sua grande difusão, o que torna o uso do aplicativo do usuario mais simples para a maioria do publico.

VEICULO DE USO RODOVIARIOS COM AS ESPECIFICAÇÕES MINIMAS NECESSARIAS

Para este projeto, definiu-se como especificações minimas necessarias para o funcionamento correto da solução:

1. Veiculo de uso rodoviario com 2 eixos;
2. Veiculo equipado com bateria de no minimo 12V com capacidade minima de 2,5A para a alimentação do módulo;
3. Veiculo que apresente uma distancia minima do solo de 10 centimetros quando vazio (distancias menores do que está se tornam dificeis a elaboração de uma escala de lotação eficiente devido a grande precisão exigida pelos sensores de distancia);
4. Veiculo que não seja operado em regiões sujeitas a alagamentos (condição necessaria uma vez que os sensores de distancia se localizam na parte de baixo do veiculo e não são tolerantes ao contato com água);
5. Veiculo que opere em região com acesso a rede de comunicação móvel de segunda geração (condição necessaria para o envio dos dados da solução para a internet).

3.2 Metodologia

A metodologia utilizada para a realização deste trabalho se deu em 5 etapas distintas.

Inicialmente, com o auxilio da plataforma IEEE, realizou-se uma revisão bibliográfica sobre as soluções atualmente disponíveis, que realizam a mesma função da solução proposta, e em seguida, foram sugeridas novas funcionalidades em relação a estas.

Já o passo seguinte foi escolher as tecnologias que seriam empregadas para a execução do projeto. Escolheu-se o uso da placa microcontroladora Arduino Mega, pelo fato da mesma ser uma plataforma *open source* e com amplo suporte na atual data, além de possuir especificações de hardware suficientes para a execução da solução. Nesta etapa também foram selecionadas Shields e módulos compatíveis com a placa e necessários para a execução do projeto.

Na terceira etapa, realizou-se o desenvolvimento dos softwares necessários para o funcionamento do hardware, bem como a realização de adaptações necessárias no hardware escolhido devido a limitações de software.

Na quarta etapa, realizou-se o desenvolvimento de todo o sistema de internet necessário para realizar a recepção dos dados coletados pelo hardware, bem como seu envio até o aplicativo do usuário, cujo desenvolvimento ocorreu em conjunto nesta etapa.

Por fim, na quinta e última etapa, realizou-se testes com um veiculo em uma rota pré-selecionada, realizando a verificação do funcionamento da solução bem como sua eficiência.

4 Testes e Resultados

4.1 Metodologia de Testes

Para se atestar e comprovar a eficácia e eficiência da solução proposta, os testes foram divididos em 2 baterias distintas, sendo elas:

1ª Bateria de Testes: nesta etapa, cada componente da solução foi testado individualmente, com o objetivo de se conhecer suas características e atestar sua precisão. Os componentes foram testados na seguinte ordem:

1) Testou-se a precisão dos 4 sensores ultrassônicos utilizados;

2) Testou-se a precisão do GPS utilizado pela *Shield*;

3) Testou-se a precisão do altímetro utilizado pela *Shield*;

4) Testou-se a precisão do velocímetro utilizado pela *Shield*;

5) Calculou-se o tempo médio de cada atualização das informações no aplicativo.

Em cada um destes ensaios, os componentes foram testados 5 vezes, e os resultados obtidos foram tabelados e comparados entre si. Em seguida, se comparou os resultados obtidos em cada item (com exceção do item 5) com a precisão de cada componente, para se assegurar que eles estavam dentro do padrão de qualidade especificado pelo fabricante. O quinto item tem a importância de ser utilizado como um alerta para o usuário do aplicativo, para que este tenha a noção que a localização mostrada do veiculo não é instantânea, ou seja, ocorre um atraso entre a localização ser adquirida, e ela ser exibida no aplicativo.

2ª Bateria de Testes: já nesta etapa, foi realizada a calibragem da escala de lotação, e foi feito o teste dos componentes em conjunto, já na sua forma final da solução. Esta etapa tem como objetivo comprovar o seu funcionamento, bem como sua qualidade, e destacar possíveis falhas e limitações de seu uso. Nesta etapa, foram realizados testes em 2 percursos pré-definidos.

4.2 1ª Bateria de Testes

TESTE DE PRECISÃO DOS SENSORES ULTRASSÔNICOS

Para o teste de precisão dos sensores ultrassônicos, foi adotado o seguinte método de teste: fixou-se os 4 sensores utilizados lado a lado em uma superfície plana e reta (ilustrada na Figura 14), e em seguida, com o auxilio de uma trena, colocou-se esta superfície a uma distancia pré-definida de uma parede (distancia essa igual a 890 mm). Com a superfície fixa, testou-se os sensores de maneira individual e tabelou-se os resultados, que se encontram na Tabela 2.

	Sensor 1	Sensor 2	Sensor 3	Sensor 4
1ª medição	889 mm	892 mm	889 mm	887 mm
2ª medição	892 mm	891 mm	888 mm	890 mm
3ª medição	890 mm	891 mm	891 mm	887 mm
4ª medição	890 mm	889 mm	890 mm	888 mm
5ª medição	891 mm	890 mm	892 mm	891 mm

Tabela 2: dados compilados a partir de teste de precisão dos sensores ultrassônicos.

Figura 14: superfície onde os sensores ultrassônicos foram dispostos para teste.

Considerando o fato de trena utilizada possuir uma precisão de 0,5 mm de acordo com o fabricante, pelos resultados pode-se concluir que os sensores estão dentro da margem de erro de 3 mm fornecido pelo fabricante, sendo aptos a serem utilizados neste projeto.

TESTE DE PRECISÃO DO GPS

Para o teste de precisão do GPS, posicionou-se a shield parada em uma localização conhecida (latitude -23.067320 e longitude -46.946412), e com isso comparou-se os resultados obtidos com a localização de referencia. Os resultados de latitude e longitude obtidos se encontram na Tabela 3, e os pontos plotados no mapa a partir destas medições se encontram na Figura 15.

Medição	Latitude	Longitude
1ª Medição	-23.067314	-46.946476
2ª Medição	-23.067297	-46.946526
3ª Medição	-23.067305	-46.946354
4ª Medição	-23.067282	-46.946445
5ª Medição	-23.067228	-46.946377

Tabela 3: pontos obtidos através do teste do GPS.

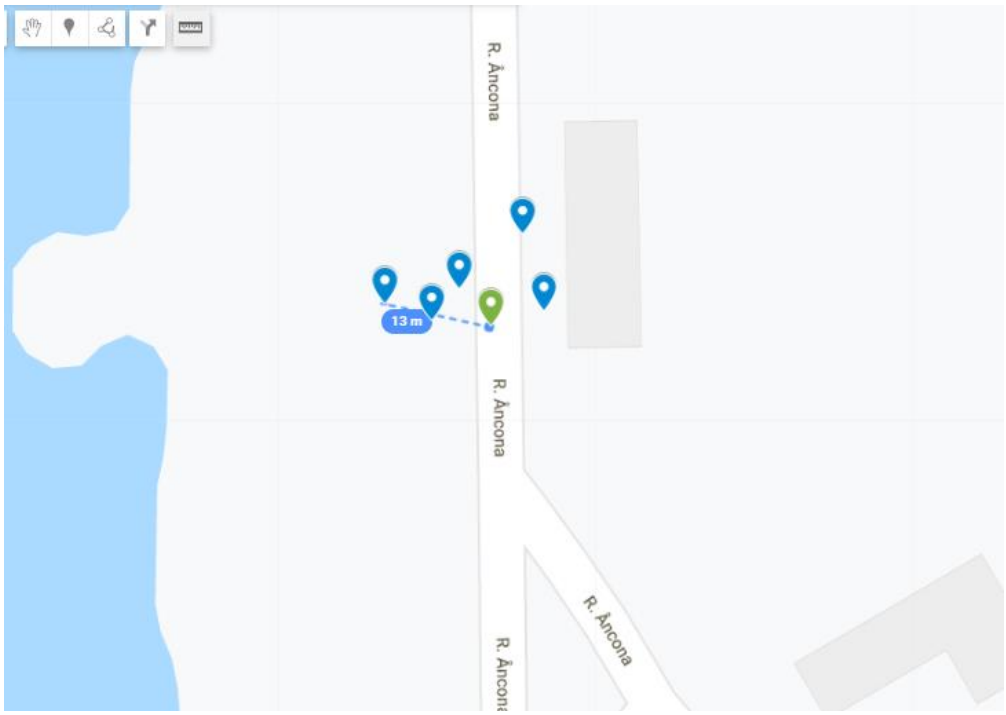

Figura 15: plotagem dos pontos obtidos no teste do GPS no mapa.

Nota-se na Figura 15 que a localização de referencia é indicada pelo marcador verde, enquanto os pontos obtidos pelo GPS da *shield* estão em azul. A partir deste mapa, nota-se que o ponto de maior distancia da localização de referencia se encontra a 13 metros em linha reta dele, o que extrapola a precisão de 2 metros definida pelo fabricante de *shield*. Esta distancia, embora mais alta do que a esperada, por não não se tratar de nenhum absurdo (ou seja, nenhuma grandeza que influencie de maneira significativa a experiencia do usuario) acaba por não comprometer o uso da *shield* selecionada.

TESTE DE PRECISÃO DO ALTÍMETRO

Para o teste de precisão do altímetro baseado em GPS, posicionou-se a *shield* estática em um ponto, em seguida, com o auxilio do Google Maps e um computador, aferiu-se a altitude deste ponto (que resultou em 707 metros de altitude).

Com o resultado do Google Maps em mãos, ligou-se a *shield*, e realizaram-se as medições. Pelos dados obtidos, notou-se que as primeiras 3 aferições (Tabela 4) apresentaram grande flutuação em relação a altitude de referencia obtida com o Google Earth Pro, sendo os resultados de altitude obtidos pela shield estabilizados a partir da quarta medição. Os resultados obtidos pela medição da *shield* após a quarta medição e consequente estabilização dos valores estão compilados na Tabela 5.

Medição	Altitude Aferida
1ª Medição	481 m
2ª Medição	552.4 m
3ª Medição	659.9 m

Tabela 4: resultados obtidos a partir dos testes de altitude da *shield* antes da estabilização dos valores aferidos.

Medição	Altitude Aferida
4ª Medição	684.5 m
5ª Medição	686.7 m
6ª Medição	686.8 m
7ª Medição	683.4 m
8ª Medição	679.4 m

Tabela 5: resultados obtidos a partir dos testes de altitude da *shield* após a estabilização dos valores aferidos.

De acordo com dados fornecidos pelo fabricante da *shield* [33], constatou-se que o altímetro possui uma precisão de 5%. Em relação ao Google Earth Pro, não se encontrou informações referentes a sua precisão.

Considerando a precisão de 5% da *shield*, os resultados apresentariam uma flutuação que iria da faixa de 671,65 metros de altitude para até 742,35 metros de altitude. Comparando esta faixa de altitudes com os resultados da Tabela, conclui-se que após 3 medidas, o altímetro utilizado estará fornecendo informações de altitude dentro da precisão fornecida pelo fabricante.

TESTE DE PRECISÃO DO VELOCIMETRO

Para o teste de precisão do velocímetro baseado em GPS, guiou-se o veiculo equipado com velocímetro digital em velocidade constante em uma via com pouco tráfego, e anotou-se a velocidade exibida no velocímetro, e comparou-se com a obtida pela medição da *shield*. Os resultados obtidos se encontram na Tabela 6.

Medição	Velocidade Medida	Velocimetro
1ª Medição	92 Km/h	100 Km/h
2ª Medição	91 Km/h	98 Km/h
3ª Medição	94 Km/h	102 Km/h
4ª Medição	93 Km/h	100 Km/h
5ª Medição	92 Km/h	100 Km/h

Tabela 6: Comparação entre as velocidades aferidas pelo velocímetro do veiculo e pela *shield*.

Observando os resultados da Tabela 6, pode-se perceber que a velocidade acusada pela solução é ligeiramente menor do que a aferida pelo veiculo. Infelizmente, não foram encontrados dados referentes à precisão de velocímetro utilizado pelo veiculo no manual do proprietário, nem dados referentes à precisão do velocímetro baseado em GPS utilizado pela *shield* em sua documentação, porém pelos dados ilustrados na Tabela 6, nota-se que as velocidades se diferem em até 8%.

TEMPO MÉDIO DAS ATUALIZAÇÕES NO APLICATIVO DO USUARIO

Para se calcular o tempo médio entre cada atualização no aplicativo do usuário, usou-se a solução em um veiculo em movimento, e cronometrou-se o intervalo de tempo entre cada atualização. Os resultados encontram-se na Tabela 7.

Atualização	Tempo Necessário
1ª para 2ª	21,3 s
2ª para 3ª	20,4 s
3ª para 4ª	19,8 s
4ª para 5ª	20,2 s
5ª para 6ª	21,8 s

Tabela 7: diferença de tempo entre as atualizações do aplicativo do usuário.

Observando-se os dados da Tabela 7, nota-se que os tempos de atualização flutuaram em um intervalo de 19 segundos para até aproximadamente 22 segundos. Devido a essa diferença de tempo entre cada atualização, adicionou-se um alerta no aplicativo do usuário de que as atualizações não eram instantâneas, com o intuito de evitar transtornos, e que poderia ocorrer um atraso de até 22 segundos entre as atualizações, como mostrado na Figura 16.

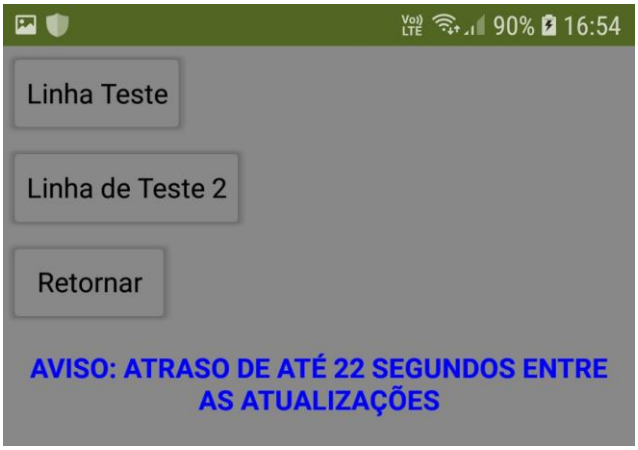

Figura 16: tela do aplicativo com a lista de linhas disponíveis no aplicativo exibindo o alerta de atraso entre as atualizações.

4.3 2ª Bateria de Testes

CALIBRAGEM DA ESCALA DE LOTAÇÃO

Para esta calibragem, foi adotado o seguinte método: com os sensores de distancia já instalados embaixo do veiculo (os pontos de fixação são, respectivamente, 7 centímetros a frente dos para-lamas traseiros, e 5 centímetros após os para-lamas dianteiros), com os pneus do veiculo calibrados para os condições normais de uso, e com o tanque abastecido em metade de sua capacidade (estas duas ultimas condições foram adotadas para buscar uma distancia com o solo mais próximo das condições normais de uso do veiculo, e para diminuir a influencia da massa do combustível nesta calibragem) foram feitas as seguintes medições:

1) Com somente 1 passageiro;
2) Com 2 passageiros;
3) Com 3 passageiros;
4) Com 4 passageiros;
5) Com 5 passageiro, mais carga adicional no porta malas (isto foi feito para ilustrar a situação de máxima lotação do veiculo).

Os resultados obtidos através da soma das distancias dos sensores se encontra na Tabela 8:

Medição	Distancia Obtida
1ª	80 cm
2ª	75 cm
3ª	69 cm
4ª	64 cm
5ª	56 cm

Tabela 8: Resultados usados para a calibragem da escala de lotação.

Através destes dados, pode-se notar que a cada passageiro adicionado ao veiculo, o mesmo teve a somatória das distancias reduzida em 6 centímetros em média. Essa diferença não foi perfeitamente linear devido ao fato dos passageiros não possuírem a mesma massa, porém comprovou o principio de funcionamento do método de medição de lotação (quanto mais lotado o veiculo, maior a deformação das suas molas, e maior sua proximidade com o chão). A Figura 17 ilustra um dos quatro sensores de distancia na parte de baixo do veiculo de testes com 4 passageiros.

Figura 17: sensor de distancia localizado logo a frente da roda traseira do lado do passageiro durante a calibração da escala de lotação com 4 passageiros.

Como a escala de lotação varia de 1 até 5, para definir os valores de distancias referentes a cada valor da escala adotou-se o seguinte método: subtraiu-se da maior distancia do solo (obtida com o veiculo ocupado somente pelo condutor, igual a 80 cm) múltiplos inteiros do valor obtido pela média da deformação da suspenção conforme se acrescentava um novo passageiro ao veiculo

(valor esse igual a 6 centímetros). Os valores das somas das distancias dos sensores ultrassônicos com seu respectivo valor na escala de lotação são mostrados na Tabela 9.

Valor Obtido Pela Soma dos Sensores	Valor na Escala de Lotação
Até 80 cm	1
De 80 até 74 cm	2
De 74 até 68 cm	3
De 68 até 62 cm	4
Abaixo de 62 cm	5

Tabela 9: Relação entre as distancias do veiculo com o solo e a sua lotação.

TESTE NA ROTA DE NUMERO 1

A rota utilizada neste teste é ilustrada na Figura 18.

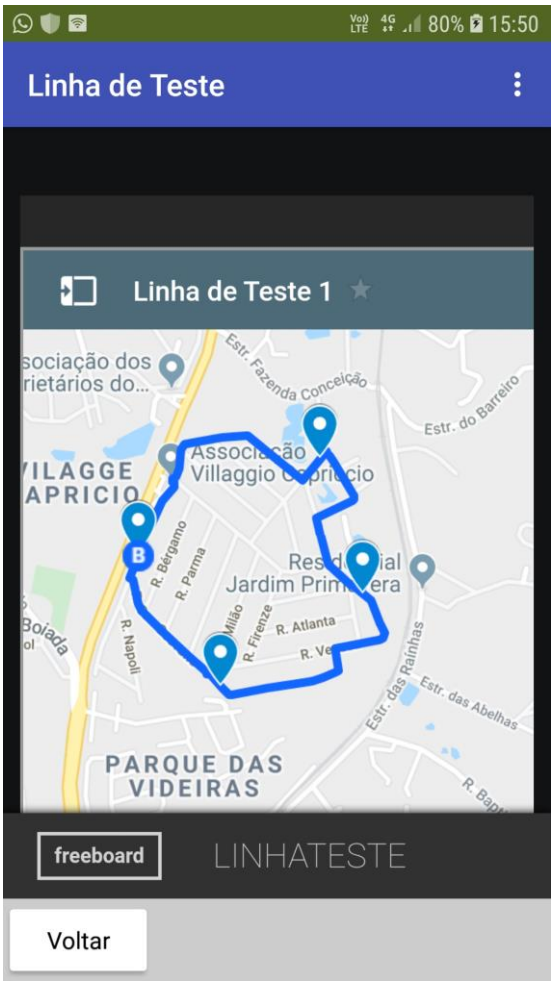

Figura 18: rota definida para o primeiro teste do veiculo.

Neste teste, o veiculo estava ocupado apenas pelo motorista. As telas obtidas pelo aplicativo estão ilustras da Figura 19 até a Figura 27. Nota-se que, por questões de segurança, o veiculo transitou durante todo este teste com velocidade inferior a 20 Km/h, devido ao fato da rota estar localizada em uma área residencial, com relativa presença de pedestres.

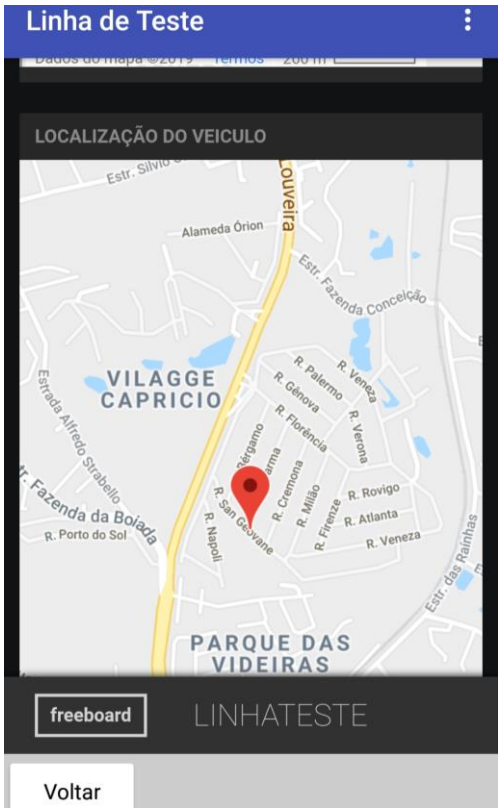

Figura 19: localização do veiculo no primeiro ponto de captura de tela do aplicativo, durante o teste da primeira rota.

Figura 20: lotação e velocidade do veiculo no primeiro ponto de captura de tela do aplicativo, durante o teste da primeira rota.

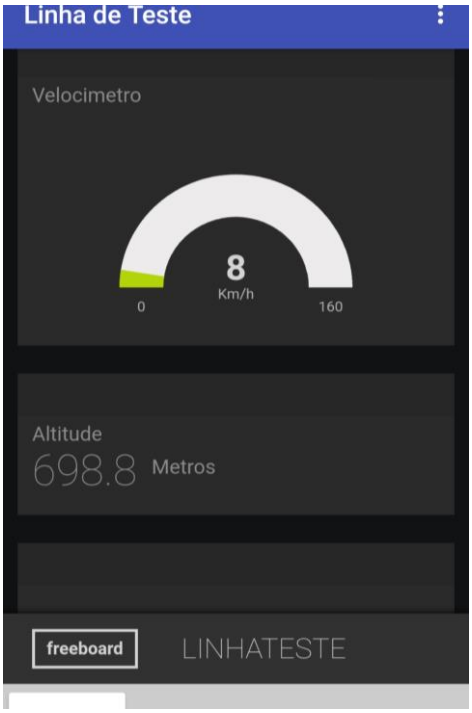

Figura 21: velocidade e altitude do veiculo no primeiro ponto de captura de tela do aplicativo, durante o teste da primeira rota.

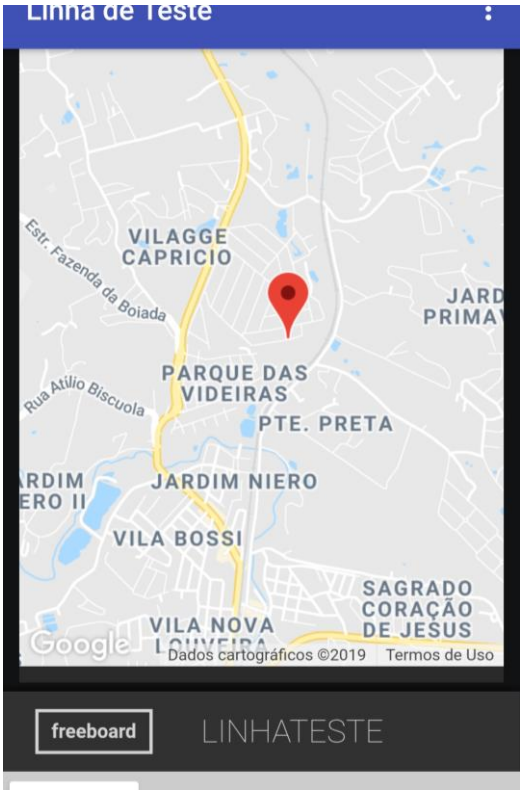

Figura 22: localização do veiculo no segundo ponto de captura de tela do aplicativo, durante o teste da primeira rota.

Figura 23: lotação e velocidade do veiculo no segundo ponto de captura de tela do aplicativo, durante o teste da primeira rota (neste momento o veiculo teve que parar devido a presença de pedestres na via, e consequentemente foi feita a medição das distancias do sensores ultrassônicos com o solo).

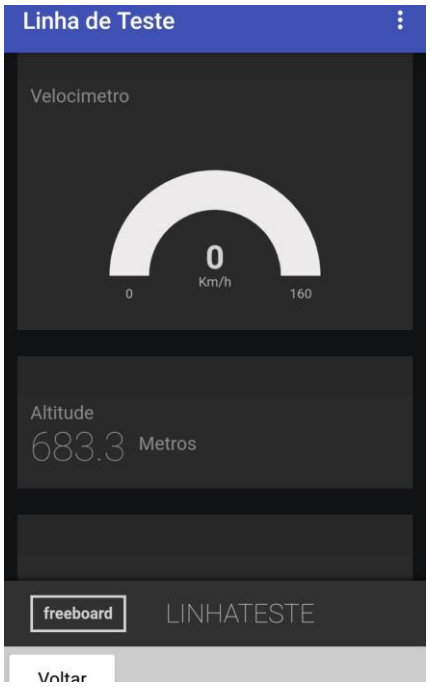

Figura 24: velocidade e altitude do veiculo no segundo ponto de captura de tela do aplicativo, durante o teste da primeira rota.

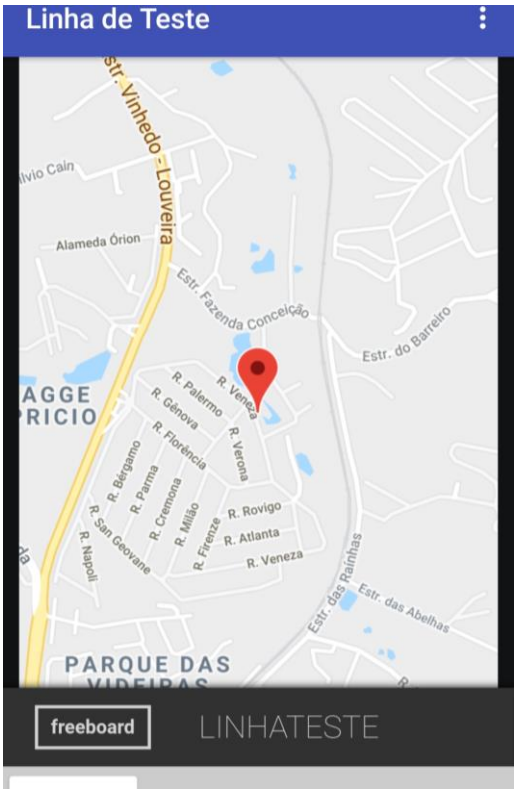

Figura 25: localização do veiculo no terceiro ponto de captura de tela do aplicativo, durante o teste da primeira rota.

Figura 26: lotação e velocidade do veiculo no terceiro ponto de captura de tela do aplicativo, durante o teste da primeira rota.

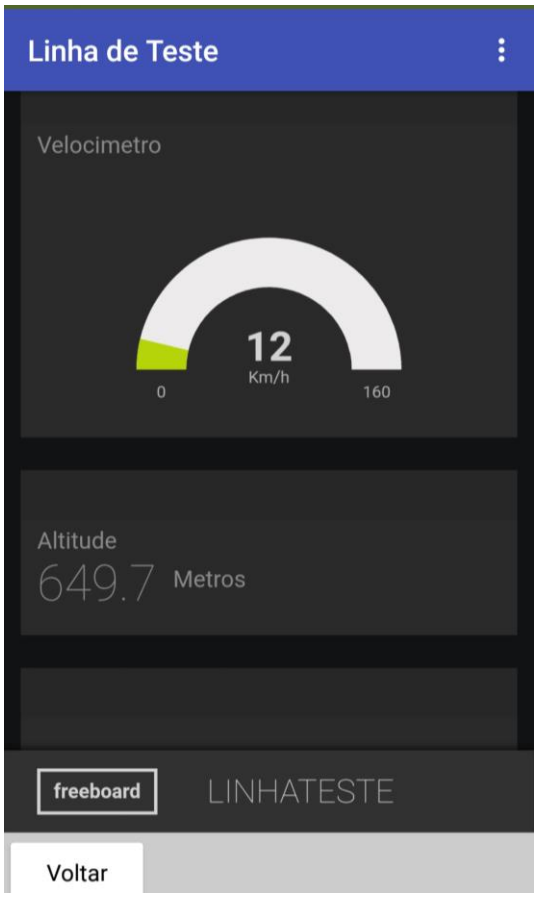

Figura 27: velocidade e altitude do veiculo no segundo ponto de captura de tela do aplicativo, durante o teste da primeira rota.

TESTE NA ROTA DE NUMERO 2

A rota utilizada neste teste é ilustrada na Figura 28.

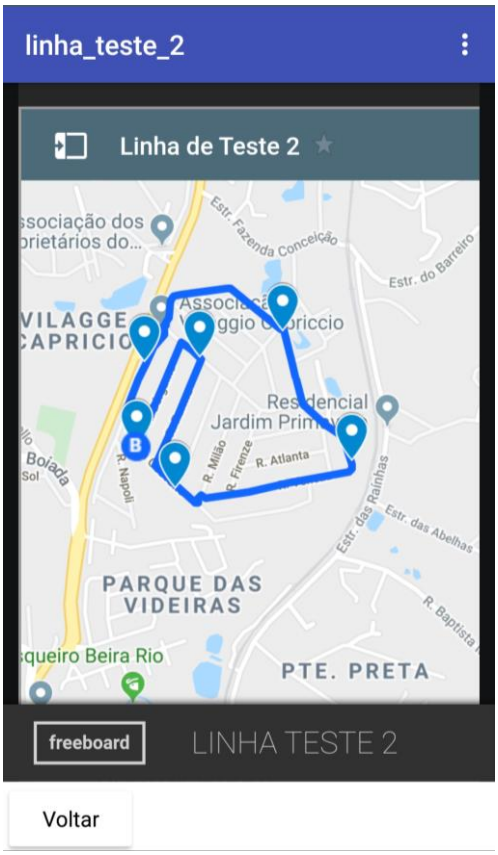

Figura 28: rota definida para o segundo teste do veiculo.

Neste teste, o veiculo estava ocupado por três pessoas no total. As telas obtidas pelo aplicativo estão ilustras da Figura 29 até a Figura 37. Nota-se que, por questões de segurança, assim como no teste da primeira rota, o veiculo transitou durante todo este teste com velocidade inferior a 20 Km/h, devido ao fato da rota estar localizada em uma área residencial, com relativa presença de pedestres. Uma observação relevante em relação a este teste é o fato de ele ter sido realizado em um dia nublado com presença de uma leve garoa no momento do teste. Apesar desta condição adversa, os sensores de distancia funcionaram de maneira correta assim como o GPS.

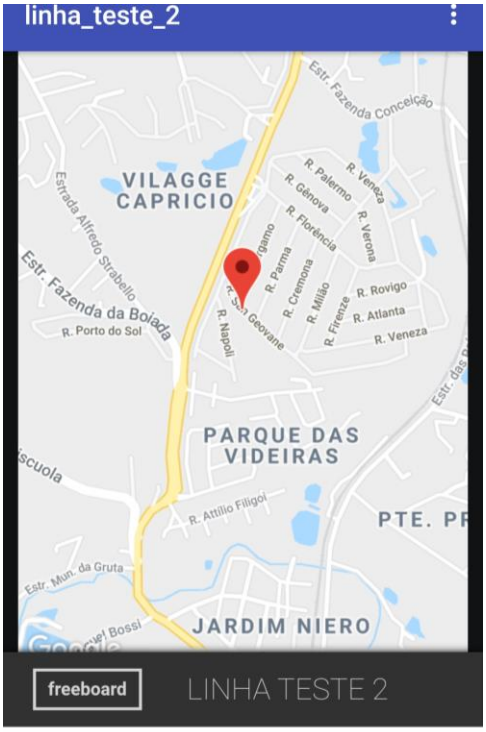

Figura 29: localização do veiculo no primeiro ponto de captura de tela do aplicativo durante o teste da segunda rota.

Figura 30: lotação e velocidade do veiculo no primeiro ponto de captura de tela do aplicativo, durante o teste da segunda rota.

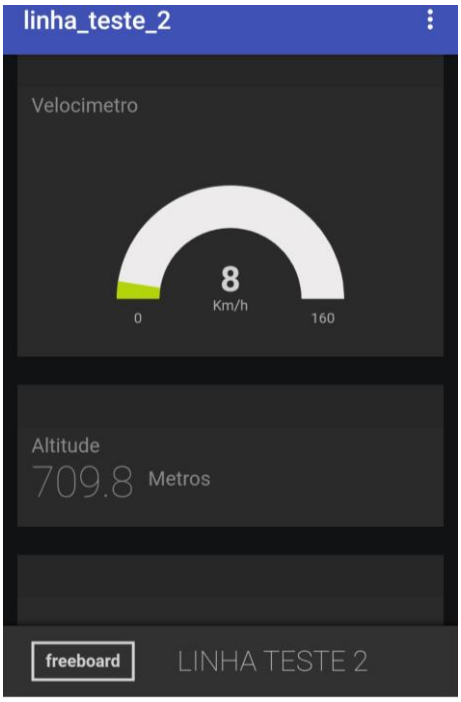

Figura 31: velocidade e altitude do veiculo no primeiro ponto de captura de tela do aplicativo, durante o teste da segunda rota.

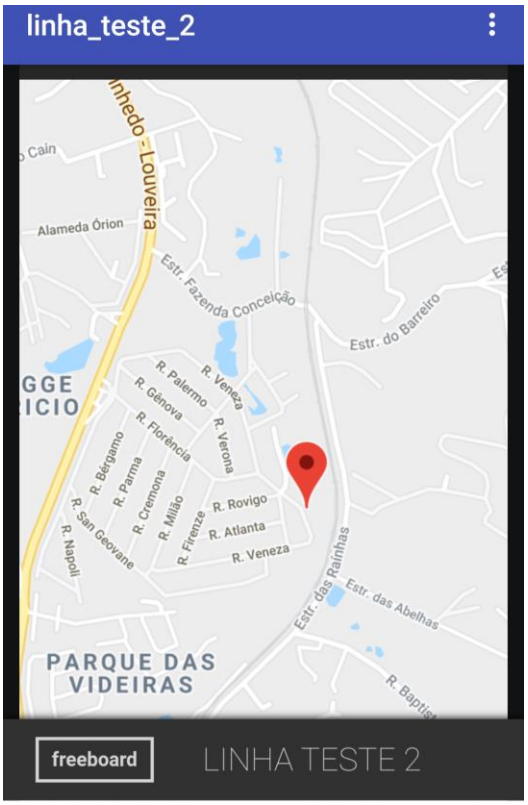

Figura 32: localização do veiculo no segundo ponto de captura de tela do aplicativo durante o teste da segunda rota.

Figura 33: lotação e velocidade do veiculo no segundo ponto de captura de tela do aplicativo, durante o teste da segunda rota.

Figura 34: velocidade e altitude do veiculo no primeiro ponto de captura de tela do aplicativo, durante o teste da segunda rota.

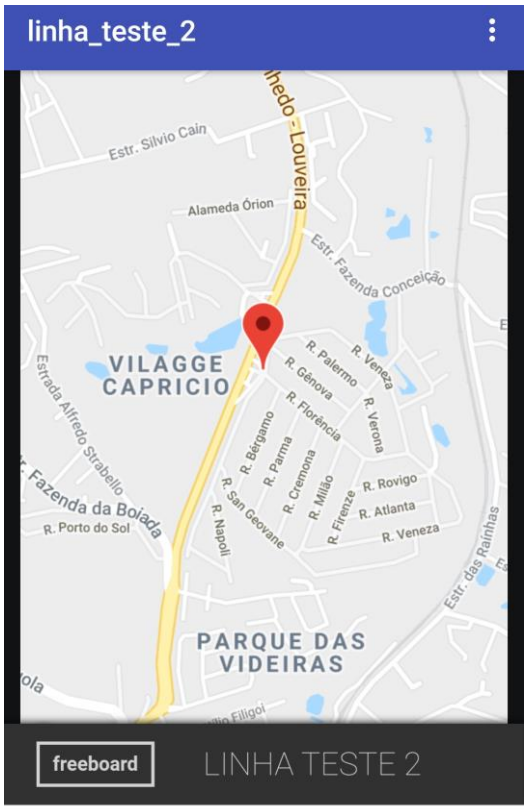

Figura 35: localização do veiculo no terceiro ponto de captura de tela do aplicativo durante o teste da segunda rota.

Figura 36: lotação e velocidade do veiculo no terceiro ponto de captura de tela do aplicativo, durante o teste da segunda rota.

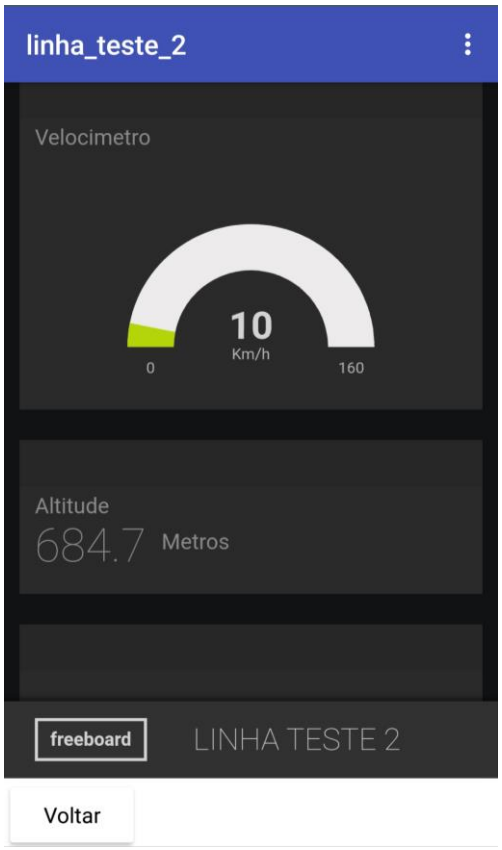

Figura 37: velocidade e altitude do veiculo no terceiro ponto de captura de tela do aplicativo, durante o teste da segunda rota.

4.3 Resultados e Discussão

Através dos resultados obtidos na seção 4.2, pode-se observar que a solução adquiriu os dados referentes à lotação, velocidade, localização e altitude de maneira satisfatória, os exibindo da maneira como planejado no aplicativo do usuário.

Durante a fase de testes da seção 4.2, nos testes das rotas pré-definidas, notou-se falhas por parte do GPS em adquirir a localização em determinados momentos, porém após algumas tentativas houve sucesso em adquiri-la. Também notou-se que em algumas ocasiões houve o travamento do programa no Arduino, sendo necessário o seu *reset* para que voltasse a funcionar de maneira adequada. Outro aspecto negativo, que não impactou nos testes, mas que pode vir a causar problemas, foi em relação à conexão do Arduino com os sensores de distancia: em situações mais severas de uso, como em pista irregular e com muita trepidação por parte do veiculo, pode ser necessário soldar nos sensores os cabos que os conectam ao Arduino, para não ocorrer o risco de um desses de soltar.

Outro aspecto que pode ser melhorado é em relação aos mapas: como a plataforma freeboard.io não contém a opção de adicionar marcadores nem rotas para os mapas gerados com os dados da *shield*, houve a necessidade de se implementar 2 mapas distintos para cada linha: um mostrando a localização do veiculo, e outro ilustrando seus pontos de parada e rota.

Mais um detalhe que deve-se observar foi a subutilização da placa Arduino Mega. Enquanto esta possuía um total de 54 portas digitais, somente 8 estavam sendo utilizadas.

Porém, apesar de todos estes detalhes, deve-se lembrar de que o projeto Arduino foi concebido com o intuito de desenvolver placas destinadas a prototipagem, e não para a elaboração de um produto final. A solução proposta por este trabalho foi competente em cumprir com aquilo que lhe foi desejado (ou seja, desenvolver uma solução que permita ao passageiro localizar o(s) ônibus de uma determinada linha de interesse, consultar seus pontos de parada, bem como sua lotação, através de um aplicativo disponibilizado gratuitamente), comprovando a sua relevância como uma alternativa de baixo custo para atenuar 2 das principais queixas dos usuários de transporte publico: lotação excessiva e atraso dos veículos.

5 Conclusões

Através dos resultados obtidos na seção de testes e resultados, pode-se concluir que a proposta inicial deste trabalho foi alcançada, ou seja, foi possível elaborar uma solução de baixo custo utilizando uma placa de desenvolvimento Arduino, solução esta que tem como objetivo ser um atenuante para duas das principais queixas apresentadas pelos usuários de transporte publico: lotação excessiva e atraso dos veículos.

Esta solução, embora seja passível de melhorias, e apesar das dificuldades enfrentadas (como a dificuldade do GPS em adquirir a localização do veiculo em determinados momentos e alguns travamentos do programa utilizado pelo Arduino Mega) se apresentou como um ótimo protótipo. Uma continuação natural deste trabalho possivelmente seria o desenvolvimento de um *hardware* totalmente dedicado para esta solução (ou seja, um *hardware* que atenda os requisitos de desempenho necessário para seu pleno funcionamento, provavelmente baseado no uso do chip SIM5320, chip este utilizado pela *shield* conectada ao Arduino), e o desenvolvimento de um mapa capaz de se integrar diretamente com a solução (sendo ele baseado no Google Maps ou em outro serviço de mapas). Outra melhoria interessante seria diminuir o tempo das atualizações das informações do veiculo no aplicativo do usuário, e também acrescentar novas funcionalidades ao aplicativo do usuário, como uma opção de chat direto do aplicativo com um representante da viação, para que o usuário possa se comunicar mais facilmente com a empresa de transporte no caso de queixas, sugestões, ou qualquer outro motivo.

A comercialização desta solução já aprimorada pode ser interessante para os proprietários de frotas de ônibus e vans, uma vez que ela poderia ser uma ferramenta interessante para os passageiros que dependem destes veículos para se locomover, na qual poderiam acompanhar a localização dos veículos de determinada linha, e com isso otimizar seu tempo, evitando assim atrasos ou perdendo muito tempo esperando por um veiculo em um ponto de parada.

REFERÊNCIAS

[1] Elisângela Azevedo Viana Gomes da Costa, “Estudo dos Constrangimentos Físicos e Mentais Sofridos pelos Motoristas de Ônibus Urbano da Cidade do Rio de Janeiro”, https://www.maxwell.vrac.puc-rio.br/9036/9036_1.PDF. Rio de Janeiro, paginas 16-21, 30 de março de 2006.

[2] I. Néspoli, Luiz Carlos Mantovani (coord.). II. Vasconcellos, Eduardo Alcântara de (coord.). III. Pires, Ailton Brasiliense (apres.). IV. Mendonça, Adolfo Luis Machado de. V. Pelegi, Alexandre. VI. Guth, Daniel. VII. Raymundo, Hélcio. VIII. Malatesta, Maria Ermelina Brosch. IX. Álvares, Olímpio Melo. X. Stuchi, Silvia Regina. XI. Godoy, Sonia. Associação Nacional de Transportes Públicos – ANTP, “Mobilidade Humana Para Um Brasil Urbano”, São Paulo – SP, 2017, pagina 103.

[3] Zvarick, Leonardo. Queixas contra ônibus sobem em SP; veja as linhas campeãs. Folha de São Paulo, São Paulo, 6 de Junho de 2019. Disponível em: https://agora.folha.uol.com.br/sao-paulo/2019/06/queixas-contra-onibus-sobem-em-sp-veja-as-linhas-campeas.shtml. Acesso em 24 de Outubro de 2019.

[4] P. Bhattarakosol, P. Tiewthanom and S. Chitwiriya, "Bus Information System: A smart partner for commuters,"5th International Conference on Computer Sciences and Convergence Information Technology, Seoul, 2010, pp. 12-17.

[5] S. A. E. Yosif, M. M. Abdelwahab, M. A. E. ALagab and F. Muhammad, "Design of bus tracking and fuel monitoring system,"*2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, Khartoum, 2017, pp. 1-5.

[6] S. Sharad, P. B. Sivakumar and V. A. Narayanan, "The smart bus for a smart city — A real-time implementation,"*2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Bangalore, 2016, pp. 1-6.

[7] R. K. Megalingam, N. Raj, A. L. Soman, L. Prakash, N. Satheesh and D. Vijay, "Smart, public buses information system,"*2014 International Conference on Communication and Signal Processing*, Melmaruvathur, 2014, pp. 1343-1347.

[8] B. Y. O. Low, S. H. Dahlan and M. H. A. Wahab, "Real-time bus location and arrival information system,"*2016 IEEE Conference on Wireless Sensors (ICWiSE)*, Langkawi, 2016, pp. 50-53.

[9] B. Janarthanan and T. Santhanakrishnan, "Real time metroplitan bus positionin system desing using GPS and GSM,"*2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, Coimbatore, 2014, pp. 1-4.

[10] K. Sujatha, P. V. N. Rao, K. J. Sruthi and A. A. Rao, "Design and development of android mobile based bus tracking system,"*2014 First International Conference on Networks & Soft Computing (ICNSC2014)*, Guntur, 2014, pp. 231-235.

[11] Official U.S. government information about the Global Positioning System (GPS) and related topics. GPS: The Global Positioning System, Página inicial. Disponível em: https://www.gps.gov/. Acesso em 24 de Outubro de 2019.

[12] National Aeronautics and Space Administration. Nasa Space Science Data Coordinated Archive. Disponível em: https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2010-022A. Acesso em 24 de Outubro de 2019.

[13] Gunter's Space Page. Disponível em: https://space.skyrocket.de/doc_sdat/navstar-2f.htm. Acesso em 24 de Outubro de 2019.

[14] Willians, Matt. What is Hooke's Law?. Phys Org, 2015. Disponível em: https://phys.org/news/2015-02-law.html. Acesso em 24 de Outubro de 2019.

[15] 1974: General-Purpose Microcontroller Family is Announced. CHM, 2019. Disponível em: https://www.computerhistory.org/siliconengine/general-purpose-microcontroller-family-is-announced/. Acesso em 24 de Outubro de 2019.

[16] Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V. Microchip, 2014. Disponivel em: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf. Acesso em 24 de Outubro de 2019.

[17] Physics of Ultrasound. Nysora, 2019. Disponivel em: https://www.nysora.com/foundations-of-regional-anesthesia/equipment/physics-of-ultrasound/. Acesso em 24 de Outubro de 2019.

[18] Coelho, F. C. R.; Silva Junior, I. C.; Dias, B. H.; Marcato, A. L. M. Metaheurística Inspirada na Ecolocalização de Morcegos: Aperfeiçoamento e Estudo de Casos. XLIV SBPO - Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro, 2012.

[19] What is Arduino?, Arduino, 2019. Disponível em: https://www.arduino.cc/en/guide/introduction. Acesso em 24 de Outubro de 2019.

[20] Arduino Mega REV3. Arduino, 2019. Disponível em: https://store.arduino.cc/usa/mega-2560-r3. Acesso em 24 de Outubro de 2019.

[21] Arduino UNO REV3. Arduino, 2019. Disponível em: https://store.arduino.cc/usa/arduino-uno-rev3. Acesso em 24 de Outubro de 2019.

[22] Arduino Leonardo with headers. Arduino, 2019. Disponível em https://store.arduino.cc/usa/leonardo. Acesso em 24 de Outubro de 2019

[23] Ultrasonic Ranging Moduloe HC-SR04. Elec Freaks, 2019. Disponível em: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. Acesso em 24 de Outubro de 2019.

[24] SIM5320 Hardware Design V1.07. SIMCom, 2012. Disponível em: https://www.tinyosshop.com/datasheet/SIM5320_Hardware+Design_V1.07.pdf. Acesso em 24 de Outubro de 2019.

[25] SIM5320. SIMCom, 2019. Disponível em: https://www.tinyosshop.com/datasheet/SIM5320_Specification_V1202.pdf. Acesso em 24 de Outubro de 2019.

[26] 3G/GPRS/GSM Shield for Arduino with GPS - European version SIM5320E. TinySine, 2019. Disponível em: https://www.tinyosshop.com/3g-gprs-gsm-shield-for-arduino-sim5320e?filter_name=3G%20shield&filter_description=true. Acesso em 24 de Outubro de 2019.

[27] Download the Arduino IDE. Arduino, 2019. Disponível em: https://www.arduino.cc/en/Main/software. Acesso em 24 de Outubro de 2019.

[28] Documentação de Referencia da Linguagem Arduino. Arduino, 2019. Disponível em: https://www.arduino.cc/reference/pt/. Acesso em 24 de Outubro de 2019.

[29] Questions?. Dweet.io, 2019. Disponível em: http://dweet.io/faq. Acesso em 24 de Outubro de 2019.

[30] Freeboard.io. Freeboard. Disponível em: https://freeboard.io/. Acesso em 24 de Outubro de 2019.

[31] About Us. MIT App Inventor, 2019. Disponível em: http://appinventor.mit.edu/about-us. Acesso em 24 de Outubro de 2019.

[32] Conheça o novo Google Maps. Google, 2019. Disponível em: https://www.google.com/intl/pt-BR/maps/about/. Acesso em 24 de Outubro de 2019.

[33] AT Command Set SIMCOM SIM5320 ATC EN V2.02. SIMCom, 2014. Disponível em: https://www.tinyosshop.com/datasheet/SIMCOM_SIM5320_ATC_EN_V2.02.pdf. Acesso em 24 de Outubro de 2019.

APÊNDICES

Link para o download da biblioteca utilizada pela *shield*:

https://github.com/adafruit/Adafruit_FONA

Link para o download da biblioteca utilizada pelos sensores ultrassônicos:

https://github.com/filipeflop/Ultrasonic

ANEXOS

Programa utilizado:

Acelerometro_Distancia_4_GPS_Internet

```
#include <Ultrasonic.h> //biblioteca do sensor ultrasonico
#include <SPI.h>
#include "Adafruit_FONA.h" //biblioteca da shield
#include <Wire.h>
#include <SoftwareSerial.h>
#define FONA_RX 13 //define os pinos de RX, TX e RST da shield
#define FONA_TX 12
#define FONA_RST 9
#define PROTOCOL_HTTP_GET     //define o protocolo utilizado para as requisições, neste caso o GET
Ultrasonic ultrasonic1(5, 6);  //definição dos pinos de cada sensor de distancia
Ultrasonic ultrasonic2(8, 7);
Ultrasonic ultrasonic3(11, 10);
Ultrasonic ultrasonic4(23, 22);
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX); //comunicação serial com a shield
SoftwareSerial *fonaSerial = &fonaSS;
bool gps_success = false;
boolean enableGPS = true; //habilita o uso do GPS
char replybuffer[255];
Adafruit_FONA_3G fona = Adafruit_FONA_3G(FONA_RST);
float latitude, longitude, speed_kph, heading, speed_mph, altitude; //variaveis do GPS
uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);
uint8_t type;
uint8_t counter = 0;
int x1; //variavel referente ao valor de distancia obtida pelo sensor de distancia numero 1
int x2;
int x3;
int x4;
int x5;
int lot; //variavel referente ao grau de lotação
```

```
int lot; //variavel referente ao grau de lotação
SoftwareSerial SIM5320(12,13);
char latBuff[12], longBuff[12], locBuff[50], speedBuff[12],
     headBuff[12], altBuff[12], tempBuff[12], battBuff[12],
     lotBuff[12]; //variaveis que serao enviadas para o dweet.io e seu comprimento
char URL[200];
char body[200]; //tamanho do comprimento do corpo da mensagem enviada ao dweet.io
char imei[16] = {0}; //variavel referente ao IMEI do dispositivo
void(* resetFunc) (void) = 0; //função que reseta o código
int counter_reset = 0; //contador de reset
int counter_reset1 = 0;


void setup()
{
  Serial.begin(4800); //comunicação a uma taxa de 4800 bauds
  SIM5320.begin(4800);
  while (!Serial);
  Serial.println(F("FONA basic test"));
  Serial.println(F("Inicializando...")); //teste basico de inicialização da shield
  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
  Serial.println(F("Erro"));
  resetFunc();   //caso haja erro ao iniciar a shield, o programa se reinicia automaticamente
  while (1);
  }
  type = fona.type();
  Serial.println(F("FONA esta OK"));
  Serial.print(F("Encontrado "));
```

```
  Serial.print(F("Encontrado "));
  switch (type) {
  }

  // Printa o IMEI do modulo
  char imei[16] = {0};
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("IMEI do módulo: "); Serial.println(imei);
  }
  while (!fona.enableGPS(true)) {
  Serial.println(F("Falha ao ligar o GPS, tentando novamente..."));
  delay(2000); // caso o GPS nao consiga ser ativado, é feita uma nova tentantiva a cada 2 segundos
  }
  Serial.println(F("GPS ativado!"));

  }

void loop()

  {

  gps_success = 0;
  while (!fona.enableGPS(true)) {
  Serial.println(F("Falha ao ligar o GPS, tentando novamente..."));
  delay(2000);
  }
  Serial.println(F("GPS ativado!"));

 }
 Serial.println(F("GPS ativado!"));

 while (!fona.getGPS(&latitude, &longitude, &speed_kph, &heading, &altitude)) {
 Serial.println(F("Erro em adquirir a localização, tentando novamente..."));
 delay(2000); // caso o GPS nao consiga adquirir a localização, espera-se 2 segundos para uma nova tentativa

 }

 Serial.print(F("Latitude: ")); Serial.println(latitude, 6);
 Serial.print(F("Longitude: ")); Serial.println(longitude, 6);
 Serial.print(F("Velocidade: ")); Serial.println(speed_kph);
 Serial.print(F("Altitude: ")); Serial.println(altitude);
 counter_reset1 = 0;


//Esta parte é somente dos sensores de distancia

 Serial.print("Sensor 01: ");
 Serial.print(ultrasonic1.read());
 Serial.println("cm");
 Serial.print("Sensor 02: ");
 Serial.print(ultrasonic2.read());
 Serial.println("cm");
 Serial.print("Sensor 03: ");
 Serial.print(ultrasonic3.read());
 Serial.println("cm");
 Serial.print("Sensor 04: ");
 Serial.print(ultrasonic4.read());
```

```
  Serial.print("Sensor 04: ");
  Serial.print(ultrasonic4.read());
  Serial.println("cm");
  x1=ultrasonic1.read();
  x2=ultrasonic2.read();
  x3=ultrasonic3.read();
  x4=ultrasonic4.read();
  x5=x1+x2+x3+x4;
  Serial.print("Soma dos sensores: ");
  Serial.print(x5);
  Serial.println(" cm");
  lot = 3;
  Serial.print("Grau de lotação: ");
  Serial.print(lot);

//define os diferentes graus de lotação, nota-se que as definições só ocorrem com
// o veiculo parado, e os valores estao em centimetros

if ((x5 < 80) && (speed_kph == 0 ))
{
 lot  = 1;
}


if (( 80 <=  x5 < 74) && (speed_kph == 0)) {
 lot  = 2;
}

if (( 74 <=  x5 < 68 ) && (speed_kph == 0)){
}

if (( 74 <=  x5 < 68 ) && (speed_kph == 0)){
 lot  = 3;
}

if (( 68 <=  x5 < 62) && (speed_kph == 0)){
 lot  = 4;
}

if (( 62 <=  x5 ) && (speed_kph == 0)){
 lot  = 5;
}


//converte os valores para variaveis do tipo string serem enviados
  dtostrf(lot, 1, 0, lotBuff);
  dtostrf(latitude, 1, 6, latBuff);
  dtostrf(longitude, 1, 6, longBuff);
  dtostrf(speed_kph, 1, 0, speedBuff);
  dtostrf(altitude, 1, 1, altBuff);


  // para enviar os dados, sera utilizado uma requesição do tipo GET

  counter = 0; // conta o numero de tentativas de enviar que ocorreram falhas

  //a função sprintf permite construir uma string
```

```
  //a função sprintf permite construir uma string
  sprintf(URL, "GET https://dweet.io:443/dweet/for/CaioLatitude?%slat=%s&long=%s&speed=%s&alt=%s&lot=%s HTTP/1.1\r\nHost: dweet.io\r\n\r\n",
  imei, latBuff, longBuff, speedBuff, altBuff, lotBuff); //variaveis que serao enviadas

    // numero de repetições caso haja falha no envio dos dados
    while (counter < 2 && !fona.postData("www.dweet.io", 443, "HTTPS", URL)) { // Server, port, connection type, URL
   Serial.println(F("Falha ao completar requisição HTTP/HTTPS..."));
   counter++;
   delay(500);
    }
   counter = 0;
   counter_reset++;

   if (counter_reset == 5)
   {
      resetFunc();
      counter_reset = 0;
   }


}
```