UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

MONITORAMENTO DA AGRESSIVIDADE NA DIREÇÃO DE CAMINHÕES ATRAVÉS DE ACELERÔMETRO E GPS

FREDY SCHLAG

BLUMENAU
2017

FREDY SCHLAG

MONITORAMENTO DA AGRESSIVIDADE NA DIREÇÃO DE CAMINHÕES ATRAVÉS DE ACELERÔMETRO E GPS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer - Orientador

BLUMENAU
2017

MONITORAMENTO DA AGRESSIVIDADE NA DIREÇÃO DE CAMINHÕES ATRAVÉS DE ACELERÔMETRO E GPS

Por

FREDY SCHLAG

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof(a). Miguel Alexandre Wisintainer, Mestre – Orientador, FURB
Membro:	Prof(a). Luciana Pereira de Araújo, Mestre – FURB
Membro:	Prof(a). Gabriele Jennrich Bambineti, Especialista – FURB

Blumenau, 30 de Junho de 2017

Dedico este trabalho a todos familiares, amigos e professores que contribuíram e incentivaram a realização deste.

AGRADECIMENTOS

Aos meus pais, pelo carinho, apoio e atenção que sempre me deram.

À minha esposa Jéssica Ringenberg, pelo apoio, incentivo e compreensão das minhas ausências.

Ao meu primo Josemar Hinz, por ceder o caminhão para os testes de campo na realização deste trabalho.

Ao meu orientador Miguel Alexandre Wisintainer, por me auxiliar no desenvolvimento do trabalho, tirando dúvidas e contribuindo com novas ideias.

Aos demais amigos e familiares que apoiaram de alguma forma e compreenderam minha ausência para que o trabalho pudesse ser concluído.

A persistência é o menor caminho do êxito.

Charles Chaplin

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo que tem o objetivo de monitorar a agressividade de motoristas na direção de caminhões. O trabalho é composto por duas partes, um dispositivo e um servidor Web. O dispositivo utiliza um microcontrolador ESP8266, juntamente com os módulos de acelerômetro, GPS e GSM. O GPS é utilizado para detectar a velocidade e posição geográfica do veículo e o acelerômetro é utilizado para detectar a inclinação do veículo, sendo que um filtro complementar é aplicado sobre os dados para minimizar os ruídos. O dispositivo envia estas informações para o servidor Web que realiza a classificação dos dados para identificar se o motorista está dirigindo de forma perigosa. O servidor Web também é responsável por enviar as notificações de perigo para os usuários do dispositivo via e-mail, além de exibir em páginas web todo o trajeto percorrido pelo motorista, apresentando informações da inclinação e velocidade. O dispositivo foi desenvolvido utilizando a linguagem de programação C++ e o servidor Web utilizando as linguagens Java e JavaScript. O protótipo foi testado em campo, percorrendo rodovias federais e estradas do interior. Por fim, o protótipo permitiu o monitoramento do veículo através do servidor Web, apresentando ao usuário informações sobre o comportamento do motorista na direção do mesmo.

Palavras-chave: ESP8266. Acelerômetro. GPS. Filtro complementar. Monitoramento.

ABSTRACT

This work presents the development of a prototype that aims to monitor the aggressiveness of drivers in the direction of trucks. The work consists of two parts, a device and a Web server. The device uses an ESP8266 microcontroller along with the accelerometer, GPS and GSM modules. The GPS is used to detect the speed and geographic position of the vehicle and the accelerometer is used to detect the inclination of the vehicle, and a complementary filter is applied on the data to minimize noise. The device sends this information to the Web server that performs the data classification to identify if the driver is driving in a dangerous manner. The Web server is also responsible for sending the notifications of danger to the users of the device via e-mail, as well as displaying the entire route taken by the driver on web pages, presenting information about the inclination and speed. The device was developed using the C ++ programming language and the Web server using the Java and JavaScript languages. The prototype was field-tested, traveling on federal highways and inland highways. Finally, the prototype allowed the monitoring of the vehicle through the Web server, presenting to the user information about the behavior of the driver in the direction of the same.

Key-words: ESP8266. Accelerometer. GPS. Complementary filter. Monitoring.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACK - Acknowledge

ADC - Analog to Digital Converter

AP - Access Point

API – Application Programming Interface

CRUD - Create, Read, Update and Delete

CSS - Cascading Style Sheets

GPIO - General-Purpose Input/Output

GPS – Global Positioning System

GPRS – General Packet Radio Services

GSM – Global System for Mobile

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

I2C - Inter-Integrated Circuit

IDE - Integrated Development Environment

IPEA - Instituto de Pesquisa Econômica Aplicada

JSON - JavaScript Object Notation

LCD - Liquid Crystal Display

LERP - Linear Interpolation

MEMS - Micro Electro Mechanical Systems

MHz - Mega Hertz

OMS - Organização Mundial de Saúde

RF - Requisito funcional

RNF - Requisito não funcional

SCL - Serial Clock

SDA - Serial Data

UART - Universal Asynchronous Receiver/Transmitter

URL - Uniform Resource Locator

UTC - Universal Coordinated Time

Wi-Fi - Wireless Fidelity

SUMÁRIO

1 INTRODUÇÃO

No ano de 2012, os acidentes de trânsito causaram 1,3 milhões de mortes, representando a oitava maior causa de mortes no mundo (WORLD HEALTH ORGANIZATION, 2014). A Organização Mundial de Saúde (OMS) estima que no ano de 2020 pode-se ter 1,9 milhões de mortes no trânsito e 2,4 milhões em 2030 (WAISELFISZ, 2012).

A OMS estima que 90% das mortes no trânsito acontecem em países de baixa e média renda e que os custos totais no mundo ultrapassem US$ 500 bilhões ao ano (BACCHIERI; BARROS, 2011). Ainda segundo os autores, no Brasil, o número de mortes e pessoas gravemente feridas em acidentes de trânsito ultrapassa 150 mil ao ano, sendo que o Instituto de Pesquisa Econômica Aplicada (IPEA) estima que os custos totais destes acidentes sejam de R$ 28 bilhões ao ano.

Nos Estados Unidos, a cada 10.000 caminhoneiros, 25 morrem em acidentes rodoviários, já no Brasil, essa taxa é de 281 caminhoneiros. Entre 2004 e 2007, os acidentes envolvendo caminhões e veículos de cargas nas rodovias federais brasileiras aumentaram em 14%. Infelizmente, os acidentes envolvendo caminhões são trágicos, pois possuem um alto número de mortes e ou pessoas feridas (BACCHIERI; BARROS, 2011).

A frota brasileira é composta por mais de 2 milhões de caminhões (BACCHIERI; BARROS, 2011). Entre os acidentes envolvendo caminhões no Brasil, o tipo mais frequente e com maior gravidade é o tombamento e capotamento, respectivamente (PAMCARY, 2007). Ainda segundo o autor, a principal causa deste tipo de acidente é a velocidade incompatível combinada com curvas fechadas e ou pista mal conservada. Os veículos mais vulneráveis a este tipo de acidente são os articulados ou caminhões extremamente carregados.

Diante deste cenário, este trabalho se propõe disponibilizar um monitoramento em tempo real da direção de caminhões, permitindo aos proprietários dos caminhões identificar o nível de agressividade do motorista na direção. Através deste monitoramento, os proprietários dos veículos poderão tomar providências a fim de evitar acidentes.

1.1 OBJETIVOS

Este trabalho tem como objetivo desenvolver um sistema de monitoramento da agressividade na direção de caminhões.

Os objetivos específicos são:

a) desenvolver uma placa que envie as informações de um acelerômetro e GPS à um servidor web;

b) desenvolver um servidor web para identificar a atitude agressiva na direção do caminhão;
c) detectar o tombamento do caminhão.

1.2 ESTRUTURA

O trabalho está organizado em quatro capítulos. O primeiro capítulo apresenta a introdução e os objetivos do trabalho. O segundo capítulo apresenta a fundamentação teórica utilizada para embasar este trabalho e finaliza com os trabalhos correlatos. A fundamentação teórica aborda os temas: acidentes envolvendo caminhões, ESP8266, sensores inerciais, módulo MPU6050, sistema de posicionamento global e o módulo SIM808. O terceiro capítulo apresenta o desenvolvimento do trabalho com requisitos, especificação, implementação, operacionalidade do protótipo e análise dos resultados. Por fim, o quarto capítulo apresenta a conclusão do trabalho juntamente com sugestões de extensões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo explorar os principais assuntos necessários para o desenvolvimento deste trabalho. A seção 2.1 apresenta estatísticas sobre acidentes de trânsito envolvendo caminhões e também apresenta os riscos de manobras perigosas realizadas pelos caminhoneiros. A seção 2.2 aborda o microcontrolador ESP8266 que será utilizado para interagir com o acelerômetro e módulos de GPS e GSM. A seção 2.3 aborda sensores inerciais, detalhando a funcionalidade do acelerômetro e giroscópio. A seção 2.4 aborda o módulo MPU6050 que tem a função de acelerômetro e giroscópio. A seção 2.5 aborda o filtro complementar que é utilizado para minimizar ruídos dos sensores. A seção 2.6 aborda a origem e funcionamento do GPS. A seção 2.7 apresenta três trabalhos correlatos.

2.1 ACIDENTES ENVOLVENDO CAMINHÕES

No Brasil, o tipo de acidente mais frequente envolvendo caminhões é o tombamento, representando 47%, seguido do capotamento com 10%, sendo que o capotamento é o tipo de acidente com maior gravidade. A principal causa destes tipos de acidentes é a velocidade incompatível combinada com curvas fechadas. A falha do motorista está presente em 66% dos acidentes envolvendo caminhões e a maioria destes acidentes são causados pela imprudência do motorista (PAMCARY, 2007).

Os motoristas de caminhões não são totalmente qualificados e frequentemente são submetidos a condições de trabalho terrivelmente exigentes, forçando-os a ficar dirigindo durante longos períodos. Os motoristas trabalham em média 92,5 horas por semana, o sistema de remuneração também é baseado em produtividade, ou seja, o motorista é incentivado a dirigir o mais veloz possível (PAMCARY, 2007).

Uma manobra muito perigosa realizada pelos caminhoneiros é a “quebra de asa”, ela consiste em balançar a carroceria do caminhão de um lado para o outro, tirando as rodas da pista, contorcendo todo o veículo. Esta manobra é considerada uma brincadeira entre os caminhoneiros, sendo que os mesmos disputam entre si quem realiza a manobra da forma mais perigosa. A “quebra de asa” pode resultar em um grave capotamento, podendo custar a vida de inocentes e do próprio motorista. Na Figura 1 é exibida a manobra “quebra de asa” (BELLINI, 2016).

Figura 1 - Manobra “quebra de asa”

Fonte: Bellini (2016).

2.2 ESP8266

ESP8266 é um microcontrolador desenvolvido pela Espressif Systems, o mesmo começou a ser produzido em 2014. Além do ESP8266 suportar a execução de programas, suporta também conexão Wireless Fidelity (Wi-Fi), podendo atuar como Access Point (AP), estação ou ambos. O ESP8266 é fornecido em vários modelos, sendo que a principal diferença entre os modelos é a quantidade de General-Purpose Input/Output (GPIO) disponíveis (KOLBAN, 2015).

Um aspecto que chama atenção, é o preço do módulo, que custa em torno de US$5,00. Todos os modelos do ESP8266 podem servir como uma ponte serial Wi-Fi, ou seja, um Arduino pode-se conectar a uma rede Wi-Fi através da comunicação serial com o ESP8266. Na Figura 2 são apresentados os modelos ESP-01 até ESP-11, como pode-se observar, todos os modelos possuem um tamanho reduzido (CURVELLO, 2015).

Figura 2 – Variações do ESP8266

Fonte: Curvello (2015).

O processador do ESP8266 é de 32-bits e possui uma frequência de 80MHz, podendo operar em até 160MHz. A tensão de alimentação do ESP8266 é de 3,3 volts. O *firmware* padrão do ESP8266 vem com um interpretador de comandos AT, para realizar a comunicação com redes Wi-Fi. A ferramenta Arduino IDE possui integração com o ESP8266 para gravar o *firmware* e o programa escrito na linguagem C++ através da comunicação serial, porém, também existem *firmwares* para interpretação de outras linguagens de programação como: Lua, Javascript e Python (ACROBOTIC, 2016).

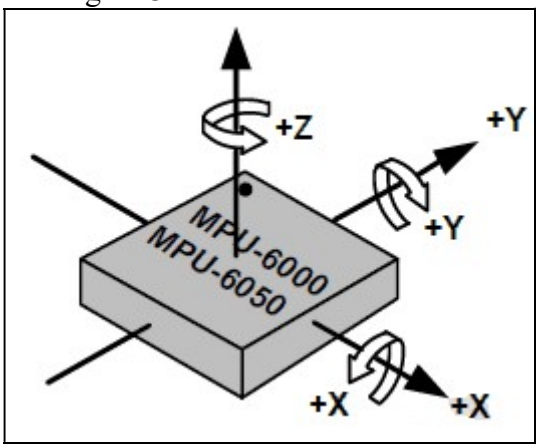
2.3 SENSORES INERCIAIS

Sensores inerciais são dispositivos baseados na tecnologia Micro Electro Mechanical Systems (MEMS). Estes sensores têm como objetivo perceber os efeitos da ação de forças que provoquem uma mudança do estado inercial de sistemas sobre os quais estas forças são exercidas. Com isto, os sensores são capazes de perceber as variações de velocidade e aceleração, linear ou angular. Os sensores inerciais são categorizados em dois tipos, os acelerômetros e giroscópios (TORRES, 2014).

O acelerômetro é um equipamento que serve para medir a aceleração de um corpo em relação à gravidade. Esta aceleração é diferente de velocidade, pois geralmente é mensurada como força g, que é basicamente a aceleração sentida como peso. Os tipos mais comuns de acelerômetros são: acelerômetro *piezoelétrico*, acelerômetro por indução magnética e acelerômetro de capacitância. A maioria dos acelerômetros possuem três eixos, sendo: x, y e z. Através destes eixos, é possível detectar a orientação do equipamento, tendo como resultado os valores de *pitch* (inclinação) e *roll* (rotação) (PAULA, 2015).

Na Figura 3 são apresentados os três eixos de um acelerômetro, os valores de x, y e z representam o ângulo de *pitch*, *roll* e *yaw*, respectivamente. O giroscópio possui a capacidade de capturar e quantificar precisamente a aceleração e a velocidade angular nos seus eixos espaciais, ele produz suas medidas baseado na preservação do movimento de um referencial fixo, ou seja, a inércia. Para se ter uma melhor precisão, é possível combinar os dados do giroscópio com os dados do acelerômetro (DINIZ, 2010).

Figura 3 - Eixos do acelerômetro

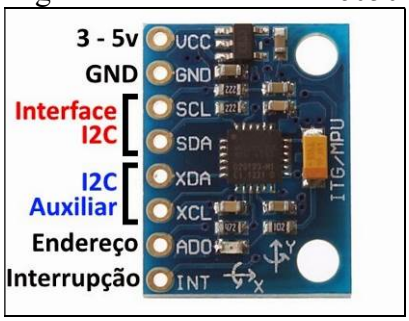


Fonte: Souza (2015).

2.4 MPU6050

O módulo MPU6050 é composto por um sensor acelerômetro e um sensor giroscópio, além de possuir conversores analógico para digital (ADC). Cada sensor possui 3 eixos, fornecendo assim um total de 6 valores de saída. A alimentação do módulo pode variar entre 3V e 5V e a comunicação entre um microcontrolador e o módulo é realizada por meio do protocolo Inter-Integrated Circuit (I2C). Na Figura 4 são exibidos os pinos do módulo (THOMSEN, 2015).

Figura 4 – Pinos do MPU6050



Fonte: Thomsen (2015).

A comunicação I2C é realizada por meio de dois barramentos sendo um para dados (Serial Data – SDA) e outro para *clock* gerado pelo dispositivo mestre (Serial Clock – SCL). A comunicação possui um *clock* de 100kHz, a transmissão de dados entre o mestre e o escravo é iniciada com a transição do sinal SDA para nível baixo, mantendo-se o SCL em nível alto e o encerramento é realizado com a transição do sinal SDA para nível alto mantendo o SCL também em nível alto. A transmissão de dados ocorrerá somente no intervalo em que o sinal SCL estiver em nível baixo. A transmissão de bytes (8 bits) sempre é encerrada com um bit de reconhecimento (ACK) estabelecido pelo dispositivo escravo.

Quando o dispositivo mestre recebe o ACK, significa que o dispositivo escravo está pronto para receber outro byte (JUNIOR, 2015).

No Quadro 1 são apresentadas as fórmulas utilizadas para calcular o ângulo de cada eixo com base nos valores brutos retornados pelo acelerômetro. A função *arctan* é a inversa da função tangente. É possível medir o ângulo de um eixo, com apenas dois eixos, porém, com três eixos é possível determinar o ângulo com maior precisão (PAULA, 2015).

Quadro 1 - Fórmulas para conversão dos valores do MPU6050

Eixo	Fórmula
Pitch	$\arctan\left(\frac{x}{\sqrt{z^2+y^2}}\right)$
Roll	$\arctan\left(\frac{y}{\sqrt{z^2+x^2}}\right)$
Yaw	$\arctan\left(\frac{z}{\sqrt{x^2+y^2}}\right)$

Fonte: Paula (2015).

2.5 FILTRO COMPLEMENTAR

O filtro complementar tem como objetivo a fusão de dados redundantes ou similares de diferentes sensores para alcançar uma estimativa ótima de uma determinada variável. Este filtro opera no domínio da frequência e pode ser definido pelo uso de duas ou mais funções de transferência as quais complementam umas às outras. Num sistema típico de duas entradas de dados, uma delas proverá informação com ruído de alta frequência, que será filtrada através de um filtro passa-baixas e outra entrada proverá informação com ruído de baixa frequência sendo filtrada por um filtro de passa-altas. Se estes filtros são matematicamente complementares, então o sinal filtrado será a reconstrução completa da variável sendo monitorada, eliminando os ruídos (BUENO; ROMANO, 2011).

Para filtragem de dados do giroscópio e acelerômetro o filtro utiliza a soma ponderada de um filtro passa alta e um filtro passa baixa, onde em altas frequências utiliza-se os valores de velocidade angular do giroscópio e em baixas frequências utiliza-se os valores do acelerômetro. O valor de qualquer ângulo com o filtro complementar é obtido através das equações listadas no Quadro 2, em que α varia de 0 a 1, ω é a velocidade angular do giroscópio e Δt é o intervalo entre uma amostra e outra (BÁSSORA; GOMES, 2014).

Quadro 2 - Equações do filtro complementar

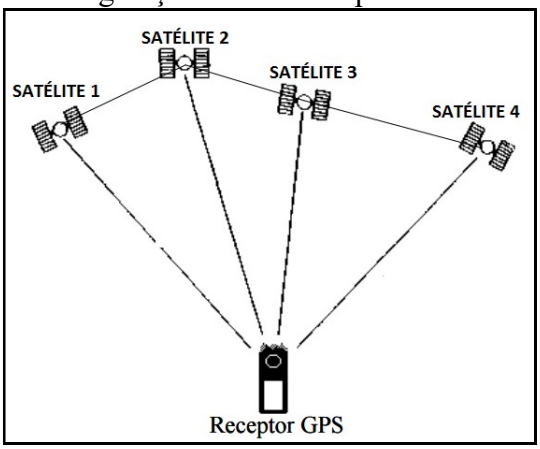
Ângulo filtrado = α * (Ângulo giroscópio) + (1 – α) * (Ângulo acelerômetro)
Ângulo giroscópio = (Ultimo ângulo filtrado) + ω * Δt

Fonte: Bássora e Gomes (2014).

2.6 SISTEMA DE POSICIONAMENTO GLOBAL (GPS)

O GPS foi projetado para se obter o posicionamento instantâneo bem como a velocidade de um ponto na superfície da terra. O GPS foi desenvolvido pelo Departamento de Defesa dos Estados Unidos da América, originalmente para fins militares, liberado com restrições para uso civil em 1977. O posicionamento do GPS é determinado por meio de uma triangulação entre os satélites e o receptor GPS, na Figura 5 observa-se como é realizada esta triangulação (NUNES ET AL., 2013).

Figura 5 - Triangulação entre o receptor de GPS e satélites



Fonte: Nunes et al. (2013).

A coordenada geográfica (latitude, longitude e altitude) é determinada através das distâncias entre o receptor com pelo menos três satélites. Como o posicionamento dos satélites no espaço são conhecidos, sabe-se todas as distâncias entre os satélites. A distância entre o satélite e o receptor pode ser calculada através da diferença de tempo que um sinal de rádio leva ao sair do satélite e chegar ao receptor e vice-versa (ALVES, 2006).

2.6.1 SIM808

O módulo SIM808 suporta as funções de GPS, GSM/GPRS e Bluetooth, sendo que o módulo deve ser alimentado com uma corrente de até 2A. O módulo de GSM/GPRS opera em quatro bandas, sendo elas 850MHz, 900MHz, 1800MHz e 1900MHz. A interação com o módulo é realizada via comunicação serial com comandos AT. A taxa de transmissão da

comunicação serial é ajustada automaticamente entre 1200bps e 115200bps. (SIMCOM, 2015b).

O conjunto de comandos AT foi criado por Dennis Hayes, em 1981. Este conjunto de comandos foi criado para permitir que os computadores interagissem com conexões telefônicas controlando um modem. Os comandos AT estão presentes nas linguagens de comando de muitos aparelhos modernos, incluindo vários periféricos de computadores (CAMPOS, 2015).

No Quadro 3 são apresentados três tipos de comandos AT. O comando de teste retorna os valores válidos para a definição do parâmetro, o comando de leitura retorna o valor definido para o parâmetro e o comando de escrita define o valor do parâmetro. Há comandos que não seguem esta sintaxe, porém, em geral os comandos iniciam com a palavra AT e terminam com uma quebra de linha (COCKINGS, 2001).

Quadro 3 - Tipos de comandos

Tipo	Sintaxe
Teste	AT<parâmetro>=?
Leitura	AT< parâmetro >?
Escrita	AT<parâmetro>=<valor>

Fonte: Cockings (2001).

No Quadro 4 são descritos os principais comandos AT para interagir com o módulo SIM808. Porém, os comandos também podem ser utilizados para interagir com qualquer módulo da série SIM800 conforme disponibilidade da função de GPS e GSM/GPRS (SIMCOM, 2015a).

Quadro 4 - Principais comandos AT do módulo SIM808

Comandos	Descrição
AT+CGNSPWR	Liga/desliga o módulo de GPS
AT+CGPSINF	Obtém as informações do GPS em diversos formatos
AT+CSQ	Obtém qualidade do sinal GSM
AT+CGATT	Verifica o estado do serviço GPRS
AT+SAPBR	Define configurações IP
AT+HTTPINIT	Inicializa o serviço HTTP
AT+HTTPTERM	Finaliza o serviço HTTP
AT+HTTPPARA	Define parâmetros das chamadas HTTP
AT+HTTPACTION	Define tipo chamada HTTP
AT+HTTPREAD	Lê a resposta do servidor HTTP

Fonte: Simcom (2015a).

O comando AT+CGPSINF=0 retorna as informações do GPS com os seguintes dados separados por vírgulas: formato, longitude, latitude, altitude, data/hora em UTC, tempo para inicialização, quantidade de satélites, velocidade de km/h e sentido. Para realizar uma

chamada HTTP, o serviço HTTP deve ser inicializado pelo comando AT+HTTPINIT, após terminar a chamada, o serviço HTTP deve ser finalizado com o comando AT+HTTPTERM (SIMCOM, 2015a).

2.7 TRABALHOS CORRELATOS

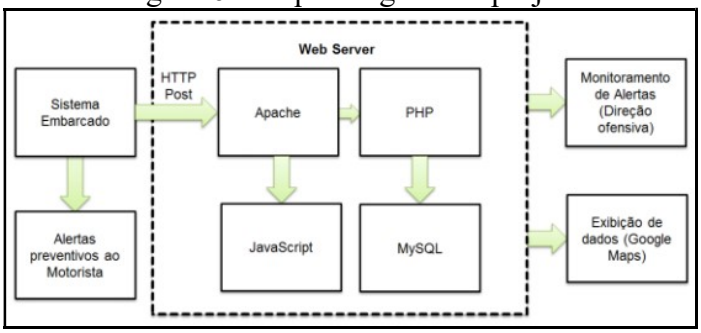
A seguir são apresentados três trabalhos correlatos. Na seção 2.7.1 será abordado o trabalho de Vallejo et al. (2013) que consiste no monitoramento rodoviário de veículos comerciais. Na seção 2.7.2 será apresentado o inclinômetro MX150 para caminhões, desenvolvido pela empresa Mestria (2016). Por fim, na seção 2.7.3 será apresentado o trabalho de conclusão de curso de Cardoso, Costa e Monteiro (2014), que consiste em um rastreador veicular.

2.7.1 SAFE DRIVER - MONITORAMENTO RODOVIÁRIO DE VEÍCULOS COMERCIAIS

O trabalho de Vallejo et al. (2013) tem como objetivo desenvolver um dispositivo afim de monitorar o comportamento de motoristas na direção de veículos de transportes comerciais, tendo como finalidade a prevenção de acidentes. Para o desenvolvimento do dispositivo foi utilizado um microcontrolador PIC24FJ64A com o módulo de GPS e GSM da Ublox. Além do dispositivo possuir as características básicas de um dispositivo de monitoramento veicular, possui também um sensor acelerômetro ADXL345 para capturar as acelerações do veículo (VALLEJO et al., 2013).

Um sensor etílico também é utilizado para verificar se o motorista ingeriu substâncias alcoólicas e se está apto para dirigir. Todos os dados coletados pelo dispositivo são enviados ao servidor Web através requisições HTTP do tipo *Post*. Através deste servidor Web é possível verificar se há anormalidades no comportamento da direção do veículo, além de visualizar a rota percorrida no Google Maps. Esta comunicação é melhor apresentada na Figura 6, onde é exibido o esquema geral do projeto (VALLEJO et al., 2013).

Figura 6 - Esquema geral do projeto



Fonte: Vallejo et al. (2013).

O servidor Web foi desenvolvido na linguagem de programação PHP, JavaScript e HTML. Todos os dados do dispositivo são armazenados em um banco de dados MySQL. O software do dispositivo foi desenvolvido na linguagem de programação C. O trabalho alcançou bons resultados quanto a detecção de comportamentos inadequados na direção, tais como, aceleração, frenagem e curvas bruscas (VALLEJO et al., 2013).

2.7.2 MX150

O inclinômetro MX150 é produzido pela Mestria. Este equipamento tem como finalidade o monitoramento dos ângulos de inclinação da carreta afim de evitar o tombamento da carreta ao descarregar a carga. O inclinômetro consiste em dois módulos, o sensor e monitor. O sensor é instalado no centro de gravidade da carreta, conectado ao monitor instalado na cabine do veículo, conforme Figura 7 (MESTRIA, 2016).

Figura 7 - Equipamento e local de instalação

Fonte: Mestria (2016).

O equipamento monitora os ângulos de inclinação horizontal e vertical durante o basculamento. Em caso de perigo durante o basculamento, o equipamento trava a báscula impedindo o tombamento lateral da carreta. A situação do inclinômetro é exibida no monitor instalado na cabine (MESTRIA, 2016).

2.7.3 APLICAÇÃO DO RASTREADOR ULLER NO MONITORAMENTO PRIVADO DE VEÍCULOS

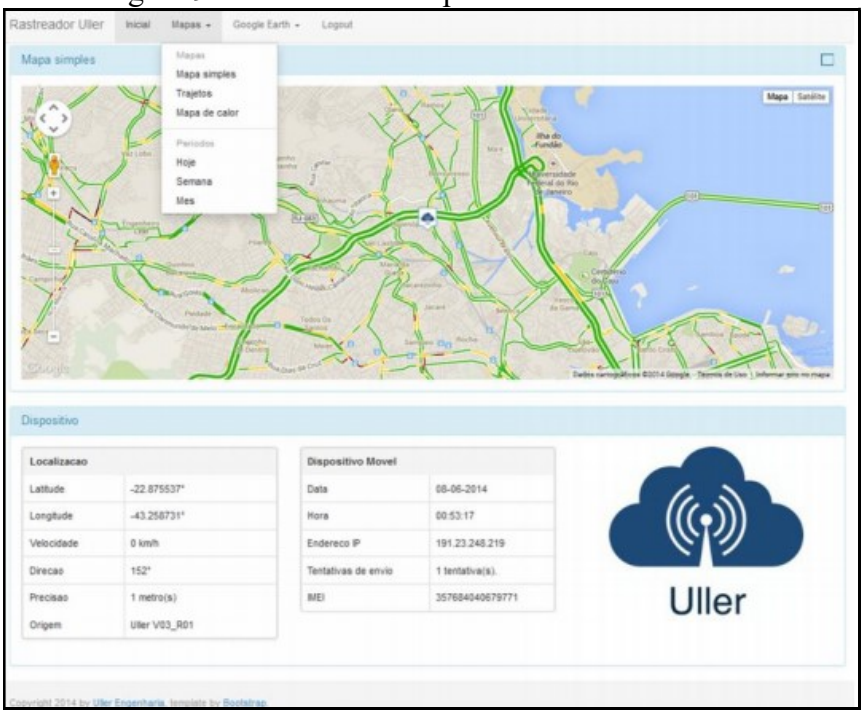
O trabalho de conclusão de curso de Cardoso, Costa e Monteiro (2014) tem como objetivo projetar um rastreador veicular de baixo custo. Além do baixo custo, o usuário do rastreador faz o próprio monitoramento sem depender de intermediários. Para o desenvolvimento do projeto, foi utilizada a plataforma Arduino juntamente com o módulo SIM908, que possui as funções de GPS, GSM e GPRS. Uma tela LCD é utilizada para verificar o funcionamento do módulo SIM908. As fases da montagem e organização do equipamento são exibidas na Figura 8. Os dados capturados pelo GPS são transmitidos à Internet através do módulo SIM908 (CARDOSO; COSTA; MONTEIRO, 2014).

Figura 8 - Fases da montagem do rastreador Uller

Fonte: Cardoso, Costa e Monteiro (2014).

As coordenadas do GPS são enviadas ao servidor Web a cada 10 segundos. A utilização do rastreador por um mês, transmite em média 45MB para o servidor Web. Para receber e tratar os dados enviados pelo módulo SIM908, foi disponibilizado um servidor Web com a plataforma de geoprocessamento GC-Tracker. Porém, a plataforma foi adaptada para interagir com o rastreador Uller, na Figura 9 é exibida a plataforma adaptada. O servidor Web foi desenvolvido na linguagem PHP e utilizou-se o banco de dados MySQL. (CARDOSO; COSTA; MONTEIRO, 2014).

Figura 9 - GC-Tracker adaptado ao rastreador Uller



Fonte: Cardoso, Costa e Monteiro (2014).

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são descritos os requisitos, a especificação do dispositivo e servidor Web. Também é apresentada a implementação, abordando a montagem, instalação e implementação do dispositivo e do servidor Web, detalhando também a sua operacionalidade. O capítulo é finalizado com a análise dos resultados, em que se realizou testes em um veículo de carga e passeio.

3.1 REQUISITOS

Na descrição dos requisitos foram utilizados os termos dispositivo e servidor Web, em que o dispositivo se refere ao software embarcado no ESP8266 e o servidor Web se refere ao software contido no servidor Web. Os requisitos funcionais e não funcionais do dispositivo são:

a) o dispositivo deverá enviar os dados do GPS para um servidor Web (Requisito Funcional – RF);
b) o dispositivo deverá coletar os dados do acelerômetro minimizando os ruídos (RF);
c) o dispositivo deverá enviar os dados do acelerômetro para um servidor Web (RF);
d) o dispositivo deverá coletar os dados do acelerômetro e GPS em intervalos de no máximo cinco segundos (RF);
e) o dispositivo deverá enviar os dados coletados para o servidor Web em intervalos de no máximo cinco minutos, quando houver conexão (RF);
f) o dispositivo deverá armazenar os dados coletados por até 15 minutos enquanto não houver conexão com o servidor Web (RF);
g) o dispositivo deverá utilizar o microcontrolador ESP8266 (Requisito Não Funcional – RNF);
h) o software do dispositivo deverá ser desenvolvido na linguagem C++ através da Arduino IDE (RNF);
i) o dispositivo deverá utilizar o acelerômetro MPU6050 (RNF);
j) o dispositivo deverá utilizar o módulo SIM808 para as funções de GPS e GSM (RNF);
k) o dispositivo deverá utilizar a rede GSM/GPRS para o envio de dados ao servidor Web (RNF);
l) o dispositivo deverá se comunicar com o módulo GSM/GPRS através de comandos AT (RNF);

Os requisitos funcionais e não funcionais do servidor Web são:

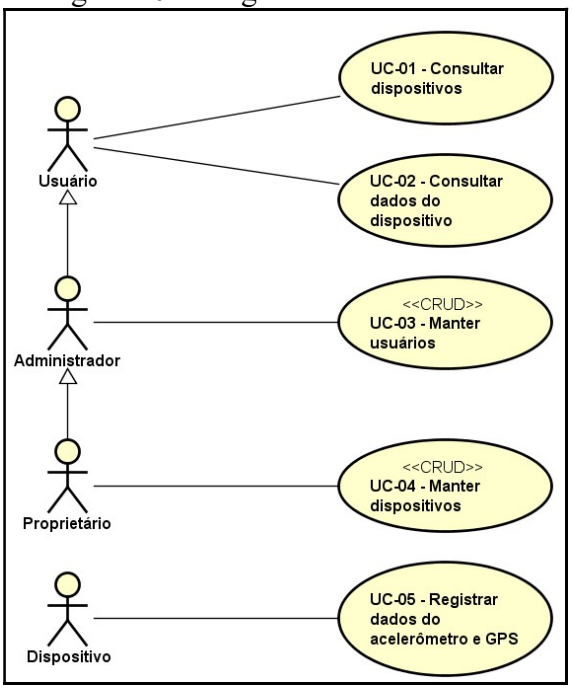
a) o servidor Web deverá disponibilizar um serviço HTTP para receber os dados do dispositivo (RF);
b) o servidor Web deverá permitir a manutenção de dispositivos (RF);
c) o servidor Web deverá permitir a manutenção de usuários por dispositivo (RF);
d) o servidor Web deverá classificar os dados reportados (RF);
e) o servidor Web deverá permitir ao usuário a consulta dos eventos do dispositivo (RF);
f) o servidor Web deverá notificar o usuário em casos que ocorra eventos de risco ou tombamento do veículo (RF);
g) o servidor Web deverá disponibilizar uma visão da rota percorrida pelo veículo (RF);
h) o servidor Web deverá possuir um usuário proprietário para cadastros iniciais (RNF);
i) o servidor Web deverá ser acessível a partir do navegador Google Chrome (RNF);
j) o servidor Web deverá ser desenvolvido na linguagem Java (RNF);
k) o servidor Web deverá se comunicar com o banco de dados PostgreSQL (RNF).

3.2 ESPECIFICAÇÃO

Nesta seção são apresentados o diagrama de casos de uso, diagrama de comunicação, diagramas de atividades, diagramas de sequência e diagrama de pacotes, sendo que estes diagramas foram desenvolvidos na ferramenta Astah Professional. Além destes diagramas, também são apresentados o modelo de entidade e relacionamento e o esquema elétrico do protótipo, sendo que estes foram desenvolvidos nas ferramentas DB Designer 4 e Gliffy, respectivamente.

Na Figura 10 apresenta-se o diagrama de casos de uso do protótipo. O ator `Proprietário` é o usuário responsável pela manutenção dos dispositivos. O ator `Administrador` é responsável por manter os usuários dos dispositivos a qual ele tem acesso, sendo que o ator `Usuario` possui apenas permissão de consulta as informações dos dispositivos associados a ele. O detalhamento dos casos de uso é apresentado no Apêndice A.

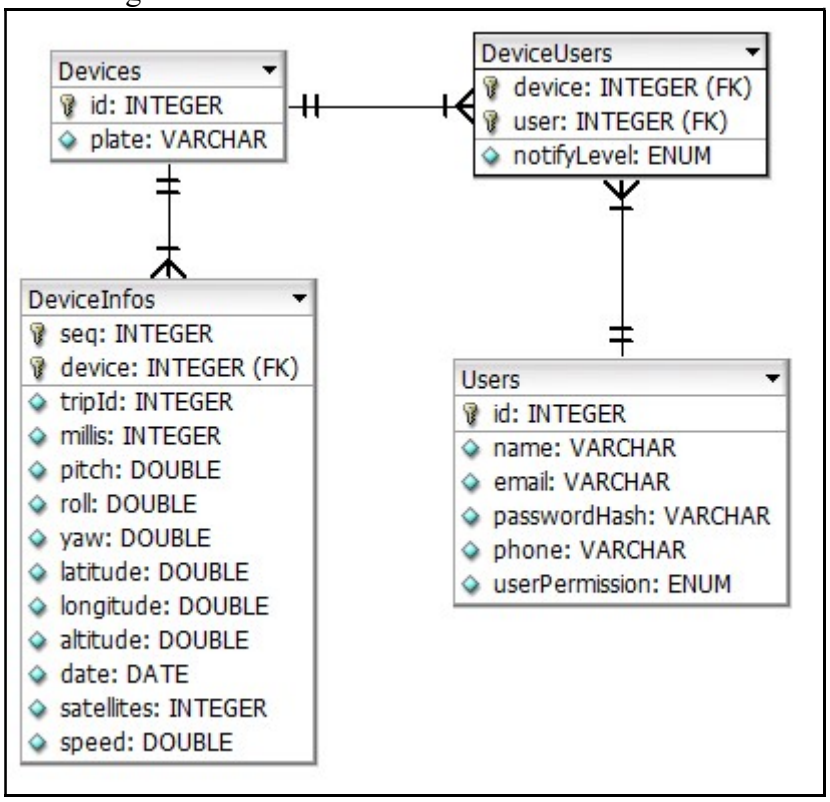
Figura 10 - Diagrama de casos de uso



Fonte: elaborado pelo autor.

Na Figura 11 apresenta-se o modelo de entidade e relacionamento do protótipo. A tabela `Devices` armazena os dados do dispositivo, tendo o `id` e a placa do veículo em qual o dispositivo é instalado. O `id` do dispositivo é o número único de identificação do módulo ESP8266. A tabela `DeviceInfos` armazena todas informações registradas pelo dispositivo, sendo que a chave é gerada conforme inserção. O campo `tripId` representa o código da viagem, a cada inicialização do dispositivo, o código da viagem é incrementado em um. A tabela `Users` armazena os dados dos usuários que acessam o servidor Web, e a tabela `DeviceUsers` define o nível de notificação de cada usuário para cada dispositivo.

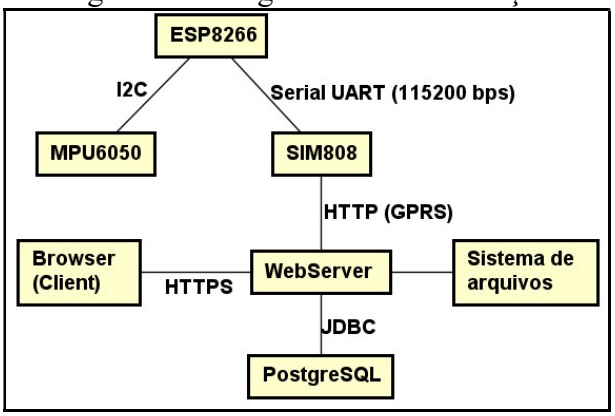
Figura 11 - Modelo de entidade e relacionamento



Fonte: elaborado pelo autor.

Na Figura 12 apresenta-se o diagrama de comunicação, exibindo os protocolos de comunicação entre todos os componentes do protótipo. O ESP8266 se comunica com o módulo SIM808 utilizando comandos AT via comunicação Serial UART com uma taxa de transmissão de 115200bps (bits por segundo). O SIM808 se comunica com o servidor Web por meio de chamadas HTTP utilizando o protocolo GPRS para se conectar à Internet via rede GSM. O cliente, acessa o servidor Web por meio de um navegador, conectando ao servidor via HTTPS.

Figura 12 - Diagrama de comunicação

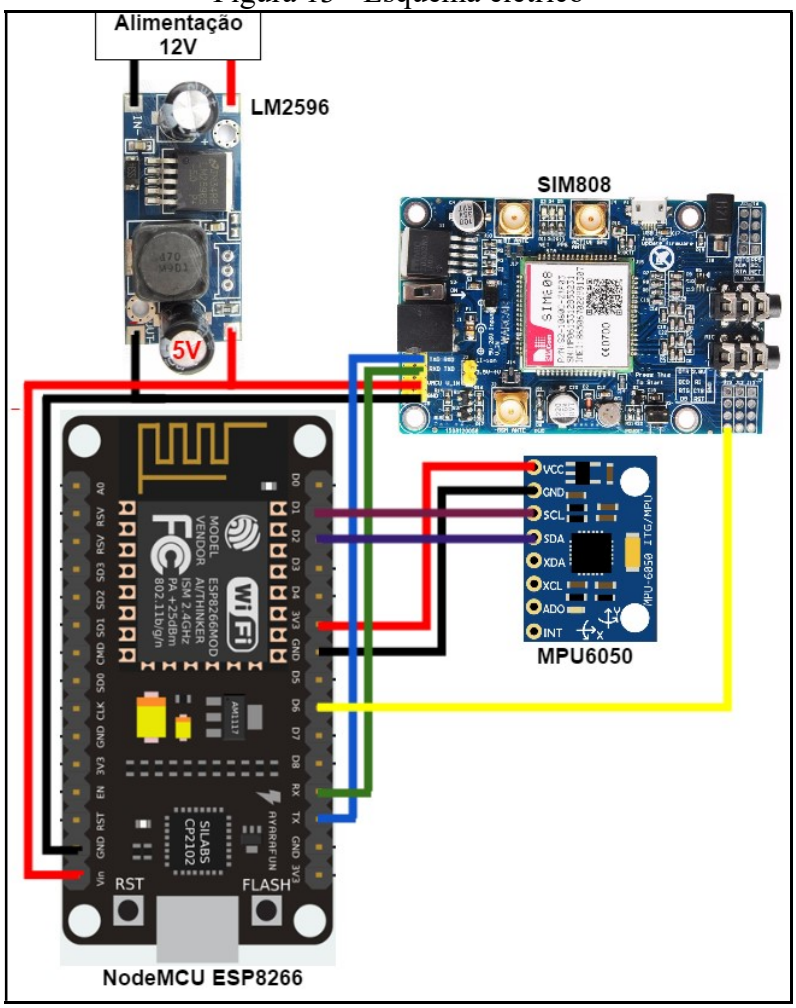


Fonte: elaborada pelo autor.

Na Figura 13 apresenta-se o esquema elétrico do protótipo. O LM2596 é responsável por regular a voltagem de entrada de 12V para 5V, pois a alimentação de 12V será provida

pela bateria do caminhão. Neste caso, os módulos ESP8266 e SIM808 são alimentados com 5V, já o MPU6050 é alimentado com 3,3V providos pelo ESP8266. O NodeMCU ESP8266 também possui um regulador de voltagem de 5V para 3,3V. O pino D6 do ESP8266 é conectado ao pino D9 do SIM808, para ser possível ligar o módulo SIM808 via software.

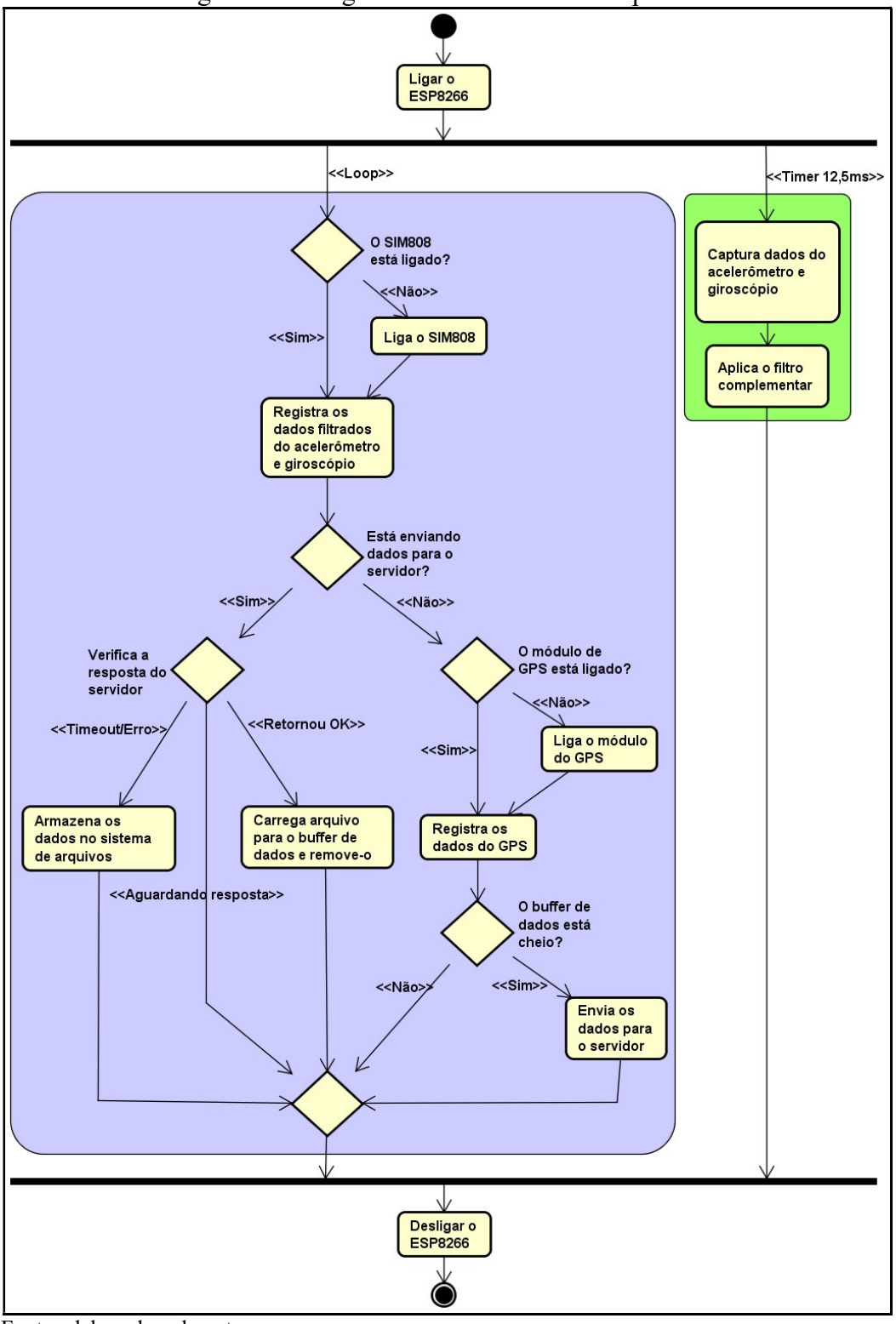
Figura 13 - Esquema elétrico



Fonte: elaborado pelo autor.

Na Figura 14 apresenta-se o diagrama de atividades do dispositivo, ou seja, o software embarcado no ESP8266. Conforme imagem, observa-se que há dois fluxos de execução, sendo um o *loop* de execução do software e outro uma interrupção a cada 12,5 milissegundos. No *loop*, inicialmente é verificado o estado do SIM808 e liga-o se necessário, porém, mesmo se não for possível ligar o SIM808, os dados do acelerômetro e giroscópio serão registrados. Os dados do acelerômetro e giroscópio são capturados e filtrados pelo filtro complementar durante a interrupção, estes dados são registrados em intervalos de no mínimo 50 milissegundos no fluxo do *loop*.

Figura 14 - Diagrama de atividades do dispositivo



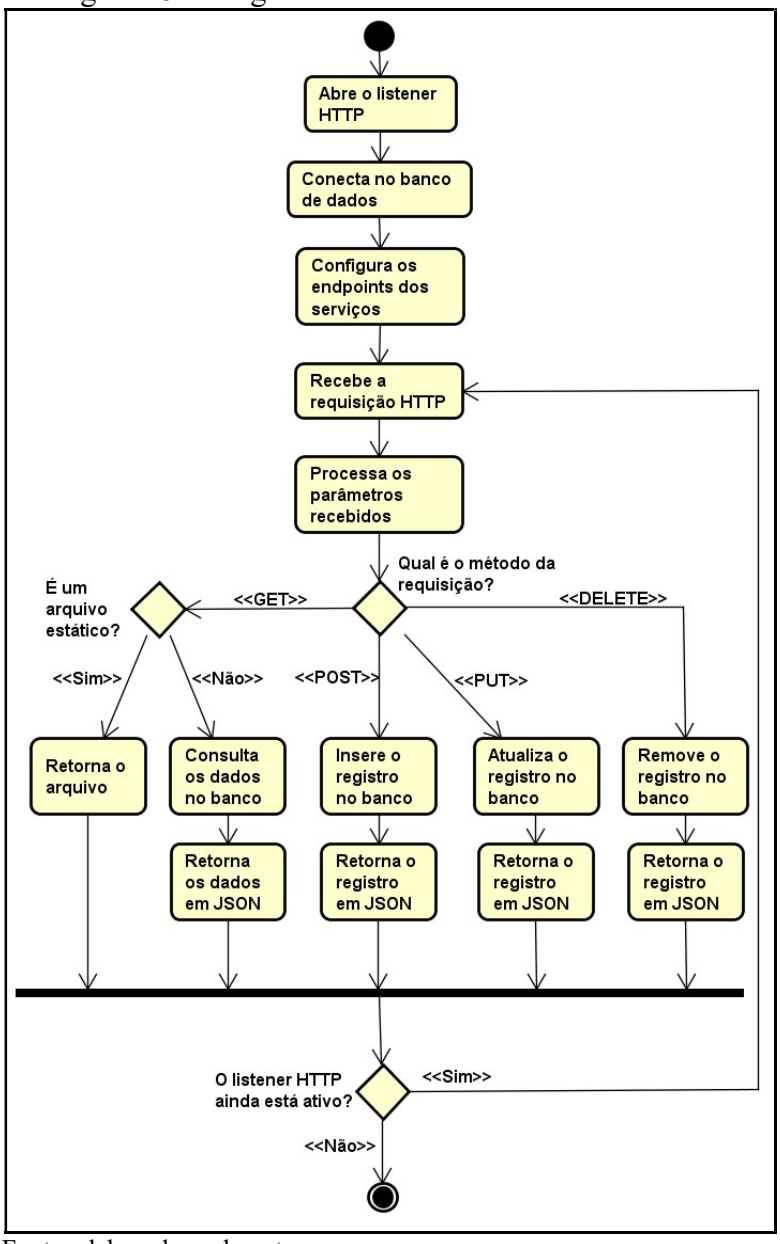
Fonte: elaborado pelo autor.

Após registrar os dados do acelerômetro e giroscópio, verifica-se se os dados estão sendo enviados para o servidor Web, pois enquanto o SIM808 está realizando uma chamada HTTP via GPRS, não é possível consultar as informações do GPS. Quando o buffer de dados atingir 2 kilobytes, os dados são enviados para o servidor Web.

Considera-se *timeout* quando o servidor não responde em 30 segundos. Quando o envio é concluído com sucesso, verifica-se se há arquivos de dados no sistema de arquivos, se houver, carrega um arquivo para o buffer e remove-o do sistema de arquivos, desta maneira, na próxima iteração do *loop*, este buffer de dados é enviado para o servidor.

Na Figura 15 apresenta-se o diagrama de atividades do servidor Web, ou seja, do software responsável por atender as requisições HTTP do dispositivo e dos usuários.

Figura 15 - Diagrama de atividades do servidor Web



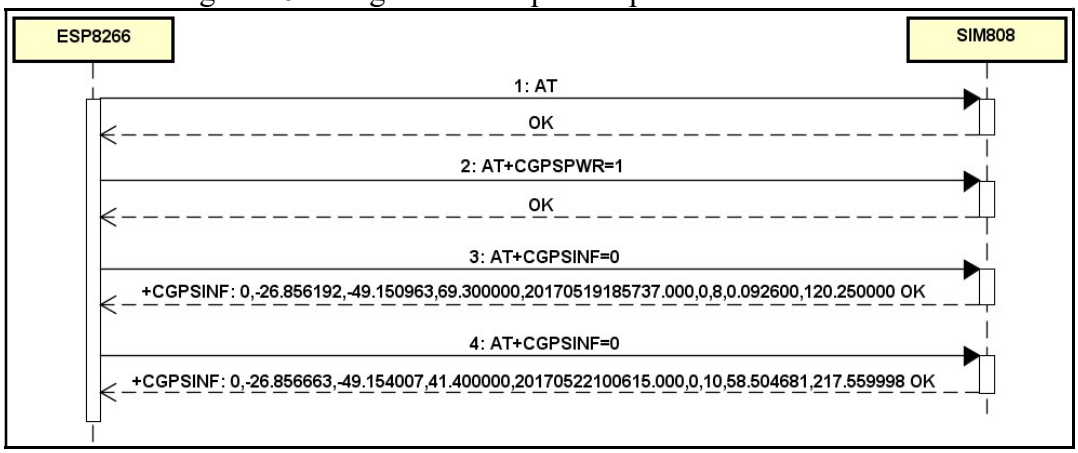
Fonte: elaborador pelo autor.

Para configuração dos *endpoints* dos serviços, foi utilizada a convenção REST. Segundo Tilkov (2008), as requisições com método GET servem para obter um recurso, para atualizar um recurso é utilizado o método PUT e para deletar é utilizado o método DELETE. O método POST normalmente é utilizado para criar o recurso, porém, também pode ser

utilizado para operações que não se encaixam em uma operação do tipo CRUD. Todas as requisições retornam os dados em formato JSON, exceto as requisições de arquivos estáticos.

Na Figura 16 apresenta-se um exemplo de troca de mensagens entre o ESP8266 e SIM808 afim de obter os dados do GPS. Inicialmente, testa-se o comando `AT`, o SIM808 retorna OK quando o mesmo estiver ligado. Em seguida, é necessário ligar o módulo de GPS através do comando `AT+CGPSPWR=1`. Estando com o módulo de GPS ligado, pode-se consultar as informações do GPS através do comando `AT+CGPSINF=0` que retorna as seguintes informações: formato da resposta, latitude, longitude, altitude, data e hora, tempo para inicialização, quantidade de satélites, velocidade (km/h) e sentido.

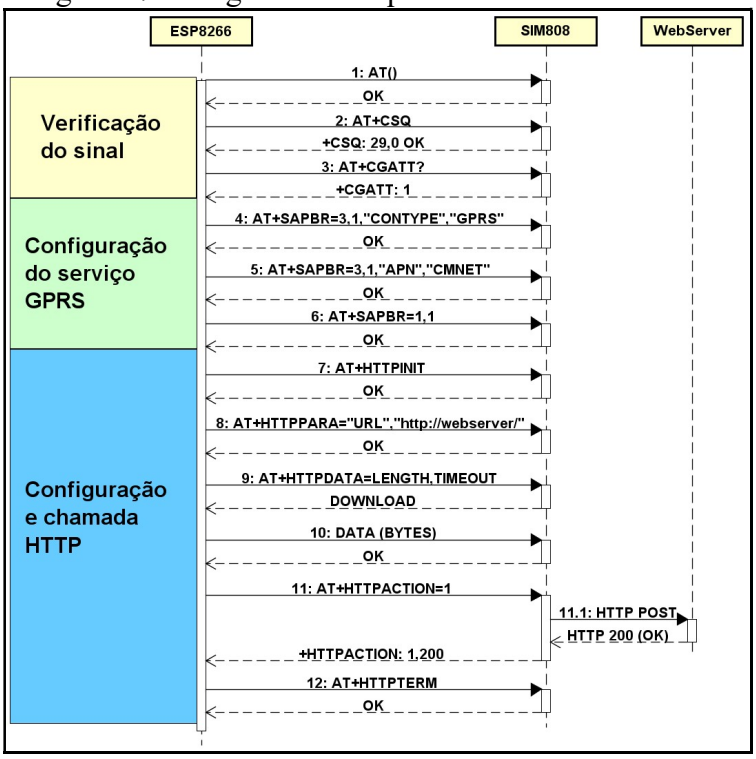
Figura 16 - Diagrama de sequência para consulta do GPS



Fonte: elaborado pelo autor.

Na Figura 17 apresenta-se o diagrama de sequência da chamada HTTP realizada pelo SIM808. A verificação do sinal e configuração do serviço GPRS é realizada entre a mensagens 2 e 6. Na mensagem 7, é iniciada a chamada HTTP, até a mensagem 10 é realizada a configuração do objeto HTTP. Na mensagem 11 é realizada de fato a chamada HTTP com o método `POST`, após isto, o SIM808 fica no aguardo da resposta HTTP do servidor. O objeto HTTP é destruído na mensagem 13. Para realizar uma nova chamada HTTP, repete-se os passos da mensagem 7 até 12.

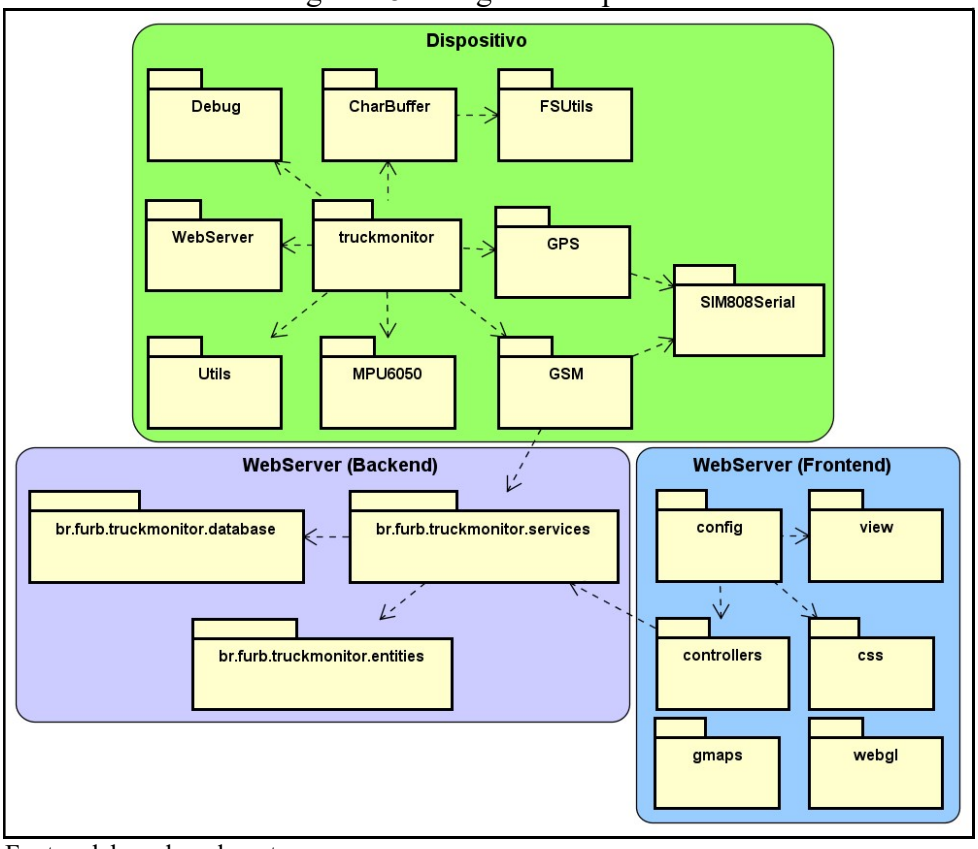
Figura 17 - Diagrama de sequência da chamada HTTP



Fonte: elaborado pelo autor.

Na Figura 18 apresenta-se o diagrama de pacotes do dispositivo e servidor Web. Cada pacote do dispositivo refere-se a um *sketch* do Arduino IDE. O pacote `truckmonitor` é o principal, onde são implementados os métodos *setup* e *loop*. Os pacotes `CharBuffer` e `FSUtils` são responsáveis por armazenar os dados do acelerômetro e GPS na memória e no sistema de arquivos. Os pacotes `GPS`, `GSM` e `SIM808Serial` são responsáveis pela interação entre o ESP8266, SIM808 e o servidor Web, sendo também responsável por obter os dados do GPS. O pacote `MPU6050` é responsável por coletar e filtrar os dados do acelerômetro e giroscópio, é nele que o timer de 13 milissegundos está implementado. Os pacotes `Debug` e `WebServer` são utilizados para depuração do dispositivo, permitindo o acesso aos logs via Wi-Fi.

Figura 18 - Diagrama de pacotes



Fonte: elaborado pelo autor.

O pacote `br.furb.truckmonitor.services` é responsável por configurar e atender as requisições dos serviços. Os serviços dependem do pacote `br.furb.truckmonitor.database` para se conectar ao banco de dados e do pacote `br.furb.truckmonitor.entities` para mapear os registros do banco de dados em objetos. O dispositivo e a parte *front-end* do servidor Web chamam os serviços do *back-end* via HTTP e HTTPS.

O pacote `config` do *front-end* é responsável por configurar as rotas, configurando o HTML (pacote `view`), CSS (pacote `css`) e *controller* (pacote `controllers`) de cada página. O pacote `gmaps` é responsável por desenhar as linhas no mapa, já o pacote `webgl` é responsável por desenhar a inclinação do veículo.

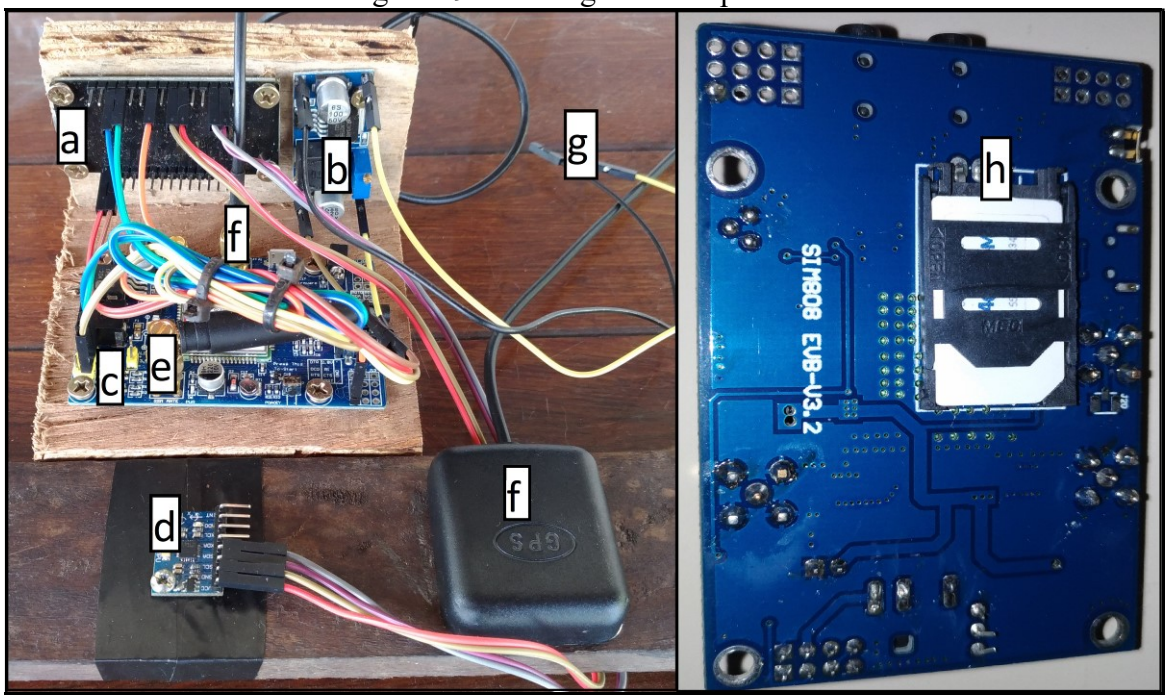
3.3 IMPLEMENTAÇÃO

Nesta seção é apresentada a montagem e instalação do dispositivo no caminhão. Também são abordadas as implementações do software embarcado no dispositivo e do servidor Web, incluindo a parte *back-end* e *front-end*. A seção é finalizada com a operacionalidade da implementação, apresentando as principais telas do servidor Web.

3.3.1 Montagem e instalação do dispositivo

Para montagem do dispositivo utilizou-se como material o MDF ultra, o qual possui uma boa resistência a água, sol e impactos. Como plano de dados utilizado pelo SIM808, escolheu-se o plano Infinity Pré Web 50 da operadora TIM. A Figura 19 apresenta como o dispositivo foi montado. No lado esquerdo da imagem são exibidos todos os módulos do dispositivo fixados ao MDF ultra. No lado direito, é exibida a parte inferior do SIM808, onde é inserido o chip SIM.

Figura 19 - Montagem do dispositivo



Fonte: elaborado pelo autor.

Os itens enumerados na Figura 19 são identificados a seguir:

a) NodeMCU ESP8266;
b) LM2596: regulador de voltagem de 12V para 5V;
c) SIM808: módulo de GPS e GSM;
d) MPU6050: acelerômetro e giroscópio;
e) antena da rede GSM;

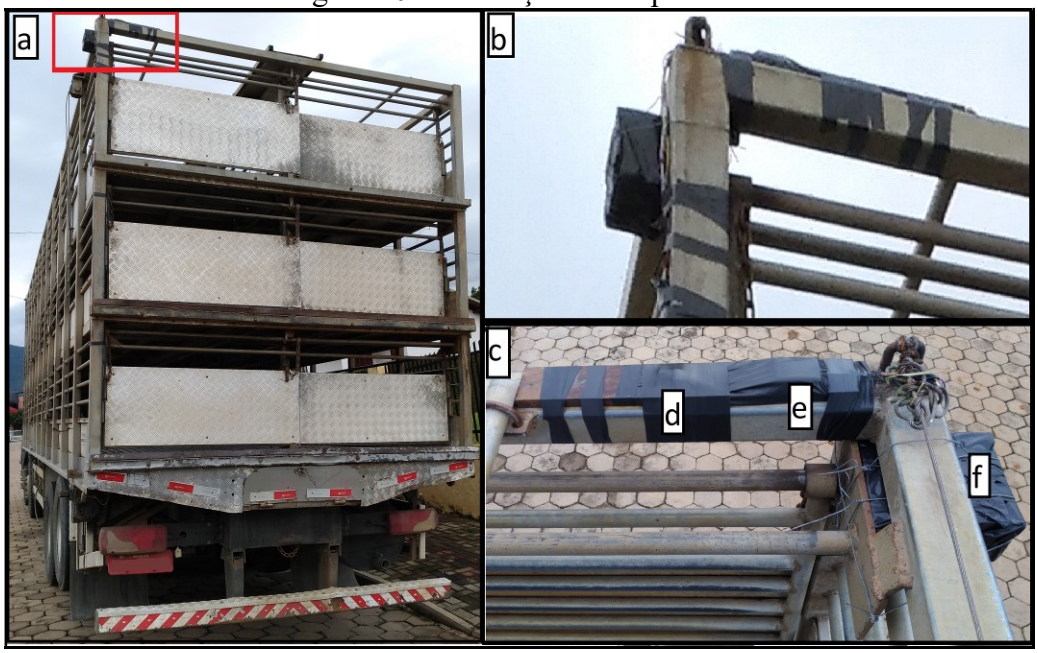
f) antena do GPS;

g) entrada de alimentação de 12V;

h) lado oposto do SIM808: compartimento do chip SIM.

O MPU6050 e a antena do GPS não são fixados junto aos outros módulos. As duas partes de MDF onde o ESP8266, SIM808 e LM2596 estão fixados, são fixadas a outras partes de MDF formando uma caixa de proteção para os módulos. Para realizar a instalação do dispositivo no caminhão, o dispositivo foi embrulhado em plástico para proteger da água. O custo dos componentes do dispositivo foi disponibilizado no Apêndice B através do Quadro 25 - Custo . A Figura 20 exibe o local de instalação do dispositivo. No item “a” da figura, é destacado o local de instalação do dispositivo, o item “b” exibe o local de instalação de forma aumentada. O item “c” exibe a visão de cima do caminhão, em que o item “d” identifica o MPU6050, o item “e” identifica a antena do GPS e o item “f” identifica a caixa de proteção com o ESP8266, SIM808 e LM2596. A alimentação do dispositivo é ligada a ignição do caminhão que fornece 12V.

Figura 20 - Instalação do dispositivo



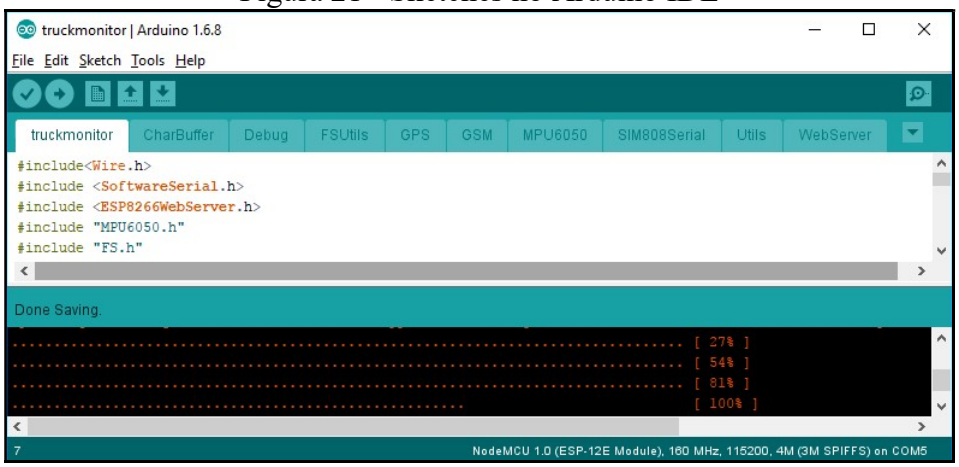
Fonte: elaborado pelo autor.

3.3.2 Técnicas e ferramentas utilizadas

O software embarcado no dispositivo foi desenvolvido na ferramenta Arduino IDE 1.6.8 com a linguagem de programação C++. Para que seja possível realizar o *upload* do software para o NodeMCU ESP8266 é necessário instalar o pacote de placas do ESP8266 no Arduino IDE e selecionar a placa NodeMCU 1.0 (ESP-12E Module). Para criar uma divisão

de responsabilidades, o código fonte foi dividido em vários *sketch*es como pode ser observado na Figura 21.

Figura 21 - Sketches no Arduino IDE



Fonte: elaborado pelo autor.

A parte *back-end* do servidor Web foi desenvolvida na ferramenta Eclipse IDE com a linguagem de programação Java, utilizando o *framework* Spark. Já a parte *front-end* foi desenvolvida na ferramenta Atom com as linguagens de programação Javascript, HTML e CSS, utilizando o *framework* AngularJS e as APIs do Google Maps e WebGL. O servidor Web e o banco de dados PostgreSQL foram hospedados na plataforma cloud Heroku.

3.3.3 Código fonte do protótipo

Nesta seção são apresentados os principais códigos do dispositivo, ou seja, do software embarcado no ESP8266. Também são apresentados os principais códigos do servidor Web, tanto da parte *back-end* como da parte *front-end*.

3.3.3.1 Dispositivo

No Quadro 5 é apresentado o código responsável pela configuração do MPU6050. Entre as linhas 2 e 7 é realizada a configuração da comunicação I2C. Na linha 9 é realizada a calibração dos sensores do giroscópio, extraindo a média dos valores do giroscópio de 10 leituras. Entre as linhas 11 e 15 é configurada a interrupção de 12,5 milissegundos, sendo que o método `timer0_ISR` será chamado na interrupção.

Quadro 5 - Configuração do MPU6050

```
 1  void setupMPU6050() {
 2    Wire.begin(4, 5); // sda, scl // D2, D1
 3    accelgyro.initialize();
 4    Wire.beginTransmission(MPU);
 5    Wire.write(0x6B);  // PWR_MGMT_1 register
 6    Wire.write(0);     // set to zero (wakes up the MPU-6050)
 7    Wire.endTransmission(true);
 8    delay(100);
 9    calibrateSensors();
10
11    noInterrupts();
12    timer0_isr_init();
13    timer0_attachInterrupt(timer0_ISR);
14    timer0_write(ESP.getCycleCount() + TIMER_CYCLES);
15    interrupts();
16  }
```

Fonte: elaborado pelo autor.

No Quadro 6 é apresentado o código que contém os métodos para habilitar e desabilitar a interrupção. A interrupção precisa ser desabilitada ao realizar alguma operação com o sistema de arquivos ou atender alguma requisição do `WebServer` do ESP8266, caso contrário, o ESP8266 é reiniciado abruptamente. A constante `TIMER_CYCLES` contém o número de ciclos que representa 12,5 milissegundos em 160MHz, sendo utilizada para calcular a próxima interrupção no método `time0_ISR`. O método `readAndFilterAccelGyro` é responsável por ler os valores brutos do acelerômetro e giroscópio e em seguida filtrar estes valores com o filtro complementar.

Quadro 6 - Método da interrupção

```
 1  void readAndFilterAccelGyro() {
 2    readAccelGyro();
 3    unsigned long xNow = millis();
 4    filterAccelGyro(xNow);
 5  }
 6
 7  /* 80Mhz -> 80*10^6 = 80000000 = 1 second | (80000000 / 1000) = 80000 = 1ms |
 8  25ms = 80000 * 25 = 2000000 cycles
 9   * 160Mhz -> 160*10^6 = 160000000 = 1 second | (160000000 / 1000) = 160000 =
10  1ms | 160000 * 12,5 = 2000000 cycles */
11  int TIMER_CYCLES = 2000000;
12
13  void timer0_ISR (void) {
14    if (!READING_ACCEL) {
15      readAndFilterAccelGyro();
16    }
17    timer0_write(ESP.getCycleCount() + TIMER_CYCLES);
18  }
19
20  void enableInterrupt() {
21    timer0_attachInterrupt(timer0_ISR);
22  }
23
24  /* Método para desabilitar a interrupção
25   * A interrupção sempre deve ser desabilitada ao atender uma requisição do
26   * WebServer ou realizar alguma operação com o sistema de arquivos */
27  void disableInterrupt() {
28    timer0_detachInterrupt();
29  }
```

Fonte: elaborado pelo autor.

No Quadro 7 é apresentado o código responsável por ler os valores brutos do MPU6050 e filtrá-los com o filtro complementar. A aplicação do filtro complementar é realizada entre as linhas 34 e 37, sendo que o giroscópio tem o fator de confiança de 96% e o acelerômetro de 4%. A conversão dos valores brutos do giroscópio para ângulo é baseada no tempo delta e no último ângulo lido e filtrado. O método `convertAccelToAngle` faz a conversão dos valores do acelerômetro para ângulo. A variável `accelgyro` é do tipo MPU6050 importada da biblioteca MPU6050.

Quadro 7 - Filtro complementar

```
 1  float RAD_TO_DEGREE = 180/3.14159;
 2  float FS_SEL = 131; // Sensibilidade do giroscópio
 3
 4  float convertAccelToAngle(int a, int b, int c) {
 5    return RAD_TO_DEGREE * atan(a / sqrt(pow(b, 2) + pow(c, 2)));
 6  }
 7
 8  void readAccelGyro() {
 9    // Lê os valores brutos do acelerômetro e giroscópio
10    accelgyro.getMotion6(&x_accel, &y_accel, &z_accel, &x_gyro, &y_gyro,
11  &z_gyro);
12  }
13
14  void filterAccelGyro(unsigned long now) {
15    // Conversão dos valores brutos do giroscópio para ângulos
16    float gyro_x = (x_gyro - base_x_gyro) / FS_SEL;
17    float gyro_y = (y_gyro - base_y_gyro) / FS_SEL;
18    float gyro_z = (z_gyro - base_z_gyro) / FS_SEL;
19
20    // Conversão os valores brutos do acelerômetro para ângulos
21    float accel_angle_x = convertAccelToAngle(y_accel, x_accel, z_accel);
22    float accel_angle_y = convertAccelToAngle(-1 * x_accel, y_accel, z_accel);
23    float accel_angle_z = convertAccelToAngle(z_accel, x_accel, y_accel);
24
25    /* Calcula o ângulo do giroscópio com base no tempo delta e ultimo ângulo
26     * filtrado */
27    float dt = (now - last_read_time) / 1000.0;
28    float gyro_angle_x = gyro_x * dt + last_x_angle;
29    float gyro_angle_y = gyro_y * dt + last_y_angle;
30    float gyro_angle_z = gyro_z * dt + last_z_angle;
31
32    /* Aplica o filtro complementar, com o fator de confiância de 0.96 para o
33     * giroscópio e 0.04 para o acelerômetro */
34    float alpha = 0.96;
35    float angle_x = alpha * gyro_angle_x + (1.0 - alpha) * accel_angle_x;
36    float angle_y = alpha * gyro_angle_y + (1.0 - alpha) * accel_angle_y;
37    float angle_z = alpha * accel_angle_z + (1.0 - alpha) * gyro_angle_z;
38
39    // Atualiza os ultimos valores filtrados
40    set_last_read_angle_data(now, angle_x, angle_y, angle_z);
41  }
```

Fonte: elaborado pelo autor.

No Quadro 8 é apresentado o método responsável por ligar o módulo SIM808, a constante `SIM808_POWERUP` refere-se ao pino D6 do ESP8622, sendo que o pino D6 está ligado ao pino D9 do SIM808. Para ligar o SIM808 via software, é necessário manter alto no pino D9 por 2 segundos, para saber se o SIM808 está ligado, testa-se o comando AT.

Quadro 8 - Ligando o SIM808

```
 1  boolean tryInitSIM808() {
 2    if (!sim808_initialized) {
 3      sim808_initialized = sendCommandSerial("AT\r\n", "OK");
 4
 5      if (!sim808_initialized) {
 6        fileLog("Tentando ligar o módulo SIM808...");
 7        digitalWrite(SIM808_POWERUP, HIGH);
 8        delay(2000);
 9        digitalWrite(SIM808_POWERUP, LOW);
10        delay(1000);
11        sim808_initialized = sendCommandSerial("AT\r\n", "OK");
12      } else {
13        fileLog("O módulo SIM808 já está ligado.");
14      }
15    }
16
17    return sim808_initialized;
18  }
```

Fonte: elaborado pelo autor.

No Quadro 9 é apresentado método responsável por obter as informações do GPS. Inicialmente, tenta ligar o GPS caso ainda esteja desligado, após isto envia-se o comando `AT+CGPSINF=0` para o SIM808. Quando o SIM808 retornar OK para este comando, o método extrai apenas a parte da resposta com as informações do GPS.

Quadro 9 - Obtendo informações do GPS

```
 1  String getGPSInfo() {
 2    String str = "";
 3    tryInitGPS();
 4
 5    if (gps_initialized && sendCommandSerial("AT+CGPSINF=0\r\n", "OK") &&
 6  (getReadedString().indexOf("+CGPSINF:") >= 0)) {
 7      str = getReadedString().substring(getReadedString().indexOf("+CGPSINF:")
 8  + 10, getReadedString().indexOf("OK"));
 9      str.trim();
10    }
11
12    return str;
13  }
```

Fonte: elaborado pelo autor.

No Quadro 10 é apresentado o método responsável pelo envio dos dados do acelerômetro e GPS para o servidor Web. O método `endBuffer` adiciona um caractere delimitador do buffer e retorna o tamanho atual do mesmo. A constante `WRITE_LIMIT` possui o valor de 1024, este limite foi criado para evitar a reinicialização do ESP8266 quando se escreve um buffer com mais de 1024 caracteres na Serial. A cada 1024 caracteres escritos na Serial do SIM808, aguarda-se 50 milissegundos. O SIM808 retorna OK após receber a quantidade de bytes previstas no comando `AT+HTTPDATA`. A chamada HTTP com o método POST é realizada na linha 28. O comando `AT+HTTPTERM` é enviado em outro método que verifica a resposta do servidor no método *loop*.

Quadro 10 - Enviando dados para o servidor Web

```
boolean sendInfo() {
  int data_size;
  int sended;
  tryInitGSM();

  if (gsm_initialized) {
     if (sendCommandSerial("AT+HTTPINIT\r\n", "OK")) {
      if (sendCommandSerial("AT+HTTPPARA=\"URL\",\"" + URL + "\"\r\n", "OK"))
{
        data_size = endBuffer();
        if (sendCommandSerial("AT+HTTPDATA=" + String(data_size) +
",3000\r\n", "DOWNLOAD")) {
          sended = 0;
          while ((sended + WRITE_LIMIT) < indexBuffer) {
              memcpy(bufferAux, charBuffer + sended, WRITE_LIMIT);
              bufferAux[1024] = '\0';
              sendCommandSerial(bufferAux);
              sended += WRITE_LIMIT;
              delay(50);
          }

          memcpy(bufferAux, charBuffer + sended, indexBuffer - sended);
          bufferAux[indexBuffer - sended] = '\0';
          sendCommandSerial(bufferAux, "OK");

          discardSerial();
          httpActionResponse = "";
          sendCommandSerial("AT+HTTPACTION=1\r\n");
          setSIM808Busy(true);
          timeoutHttpAction = millis() + 30000;
          return true;
        } else {
          fileLog("SendInfo: HTTPDATA ERROR");
        }
      } else {
        fileLog("SendInfo: HTTPPARA ERROR");
      }
    } else {
      fileLog("SendInfo: HTTPINIT ERROR");
    }
  } else {
    fileLog("SendInfo: GSM não inicializado.");
  }

  return false;
}
```

Fonte: elaborado pelo autor.

No Quadro 11 é apresentado o método responsável pela inclusão de caracteres no buffer, juntamente com a compactação dos mesmos. A variável `charBuffer` tem um tamanho de 15 mil caracteres. Como o conjunto de caracteres inseridos no buffer é de apenas 15 caracteres (números e sinais), foi utilizado um byte (8 bits) para dois caracteres, pois a cada 4 bits é possível representar 16 estados. Esta compactação do buffer é realizada para otimizar o tempo de envio ao servidor Web e também consumir menos espaço em memória e no sistema de arquivos do ESP8266.

Quadro 11 - Compactação de dados

```
boolean addCharBuffer(char c) {
  /*
   * , = 0x01
   * - = 0x02
   * . = 0x03
   * 0 = 0x04
   * 1 = 0x05
   * 2 = 0x06
   * 3 = 0x07
   * 4 = 0x08
   * 5 = 0x09
   * 6 = 0x0A
   * 7 = 0x0B
   * 8 = 0x0C
   * 9 = 0x0D
   * : = 0x0E
   * ; = 0x0F
   */
  if (indexBuffer < BUFFER_SIZE) {
    if (c < 47) {
      c = c - 43; // , - . (44, 45, 46)
    } else {
      c = c - 44; // 0 = 48 , : = 58
    }

    if (bitshift) {
      c = (c << 4) & 0xF0;
      charBuffer[indexBuffer] = c;
    } else {
      c = c & 0x0F;
      charBuffer[indexBuffer] = c | charBuffer[indexBuffer];
      indexBuffer++;
    }

    bitshift = !bitshift;
    return true;
  } else {
    return false;
  }
}
```

Fonte: elaborado pelo autor.

No Quadro 12 é apresentada a disposição dos dados no buffer. O buffer sempre inicia com o id do dispositivo seguido do código da viagem. Após isto, são adicionados blocos de dados que contém dados do acelerômetro e GPS. Cada bloco inicia com o tempo de atividade do ESP8266 e os dados são separados por vírgula. Os blocos são separados pelo sinal de dois pontos (:). Todo bloco de dados possui dados do acelerômetro, porém, nem todos possuem dados do GPS. A data e hora de cada bloco de dados é calculada no servidor Web, com base na data retornada pelo GPS e tempo de atividade do ESP8266.

Quadro 12 – Disposição dos dados no buffer

Formato	`DeviceID,TripId:<Millis,X,Y,Z,[GPSINFO]>:<Millis,X,Y,Z>...`
Exemplo não compactado	`8854976,30:181177,0.03,-3.40,85.86,0,-26.833452,-49.184537,97.100000,20170519162721.000,0,11,20.779440,164.009995:182178,-0.57,-4.70,81.01;8854976,30:184263,-1.28,-5.01,86.69,0,-26.833637,-49.184520,95.500000,20170519162724.000,0,11,22.186960,166.089996:185264,-4.31,-2.41,83.79,0,-26.833702,-49.184517,94.800000,20170519162725.000,0,11,23.242599,168.929993`
Exemplo compactado (Hexadecimal)	`CC98DBA174E5C55BB143471273841C93CA14126A3C77896128D35C897B1DB35444441645B495D5A6B653444141551643BBD88415A8344DDD9E5C65BC12439B1283B41C5345FCC98DBA174E5C86A712536C1293451CA3AD14126A3C77A7B128D35C89641D939444441645B495D5A6B683444141551663 5CADA415AA34CDDDAE5C96A81283751263851C73BD14126A3C77B46128D35C895B1D83C444441645B495D5A6B6934441415516736869DD15AC3D6DDD7`

Fonte: elaborado pelo autor.

No Quadro 13 é apresentado o código responsável por iniciar o servidor Web do ESP8266, que é utilizado para depuração do dispositivo. Inicialmente, na linha 4, é inicializada a função de Wi-Fi do ESP8266 como *Access Point* (AP). Entre as linhas 5 e 10 são configurados os *endpoints* do servidor Web do ESP8266. O servidor Web é iniciado na linha 15, sendo que a porta definida para atender as requisições é definida na declaração da variável `server`.

Quadro 13 - Servidor para depuração do ESP8266

```
 1  ESP8266WebServer server(80);
 2
 3  void setupWebServer() {
 4    WiFi.softAP(ssid, password);
 5    server.on("/listFiles", HTTP_GET, handleListFiles);
 6    server.on("/deleteFile", HTTP_GET, handleDeleteFile);
 7    server.on("/info", HTTP_GET, handleInfo);
 8    server.on("/serialLog", HTTP_GET, handleSerialLog);
 9    server.on("/fileLog", HTTP_GET, handleFileLog);
10    server.onNotFound([](){
11      if(!handleFileRead(server.uri()))
12        server.send(404, "text/plain", "FileNotFound");
13    });
14
15    server.begin();
16  }
```

Fonte: elaborado pelo autor.

O método `handleListFiles` retorna um JSON com todos os arquivos do sistema de arquivos do ESP8266, juntamente com o tamanho de cada um. O método `handleDeleteFile` é responsável por deletar um arquivo cujo é definido como parâmetro da requisição. O

método *handleInfo*, retorna informações do ESP8266 e o estado do SIM808. O método `handleSerialLog` serve para controlar a escrita de log na Serial por software do ESP8266 (pinos D7 e D8), caso este log esteja habilitado, as informações de depuração, incluindo requisições e respostas do SIM808 são escritas na Serial por software. O método `handleFileLog` serve para ligar ou desligar a escrita de logs em arquivo, este log é utilizada para identificação de problemas durante o envio de dados ao servidor Web. O método *handleFileRead* retorna o conteúdo de um arquivo no sistema de arquivos, este método é utilizado para acessar o log em arquivo.

3.3.3.2 Servidor Web

No Quadro 14 é apresentado o código da classe `DeviceService`, do pacote `br.furb.truckmonitor.services`. Nas linhas 3, 27, 31 e 35 são configurados os *endpoints* dos serviços, como pode ser observado, as URLs dos serviços são as mesmas para todas operações, o que muda é o método HTTP. O método `getObject` chamado na linha 6, é responsável por converter o JSON vindo da requisição para objeto Java. Todos os serviços seguem este mesmo padrão. Para a publicação dos serviços utilizou-se o Spark Framework, para publicar um serviço basta chamar o método do Spark referente ao tipo requisição HTTP, passando como parâmetro o *endpoint* do serviço e a implementação do mesmo.

Quadro 14 - Classe DeviceService

```
 1  public class DeviceService {
 2    public static void init() {
 3      Spark.post("/device", new JsonService() {
 4        @Override
 5        public Object doHandle(Request request, Response response) {
 6          Device device = getObject(request, Device.class);
 7
 8          EntityManager em = Database.createEntityManager();
 9          try {
10                 EntityTransaction transaction = em.getTransaction();
11                 transaction.begin();
12                 try {
13                   em.persist(device);
14                   transaction.commit();
15                 } catch (Throwable e) {
16                   e.printStackTrace();
17                   transaction.rollback();
18                   throw e;
19                 }
20                 return device;
21          } finally {
22            em.close();
23          }
24        }
25      });
26
27      Spark.put("/device", new JsonService() {
28        [...]
29      });
30
31      Spark.delete("/device/:id", new JsonService() {
32        [...]
33      });
34
35      Spark.get("/device/:id", new JsonService() {
36        [...]
37      });
38    }
39  }
```

Fonte: elaborado pelo autor.

Para o mapeamento das entidades no banco de dados foi utilizado a biblioteca EclipseLink Java Persistence API (JPA). No Quadro 15 são apresentados os métodos responsáveis pela classificação das informações providas pelo dispositivo e pelo envio de notificações por e-mail. A classificação é realizada no método `checkRisk`. Considera-se velocidade de risco quando a velocidade do veículo ultrapassa 80km/h. Uma inclinação de risco é considerada quando o ângulo é maior que 20°, é considerado tombamento quando o ângulo é maior do que 70°. Chegou-se ao ângulo de 20º com base na Figura 1, onde o caminhão teve uma inclinação de aproximadamente 30º.

Quadro 15 - Notificação de eventos de risco

```
 1  private static final int SPEED_RISK = 80;
 2  private static final int ROLL_RISK = 20;
 3  private static final int ROLLOVER = 70;
 4  private static final int SPEED_LIMIT = 120;
 5  private void checkRisk(DeviceInfo info) {
 6    double roll = Math.abs(info.getRoll());
 7    if ((info.getSpeed() > SPEED_RISK) || (roll > ROLL_RISK)) {
 8
 9      NotifyRisk notify = new NotifyRisk();
10      notify.setSpeedRisk(info.getSpeed() > SPEED_RISK);
11      notify.setRollRisk(roll > ROLL_RISK);
12      if ((info.getSpeed() > SPEED_LIMIT) || (roll > ROLLOVER)) {
13        notify.setLevel(NotifyLevel.EMERGENCY);
14      } else {
15        notify.setLevel(NotifyLevel.WARNING);
16      }
17      if (notify.isSpeedRisk()) {
18        speedRisks.add(notify);
19      }
20      if (notify.isRollRisk()) {
21        rollRisks.add(notify);
22      }
23    }
24  }
25  private void sendNotifications() {
26    organizeRisks();
27    for (NotifyRisk n : risks) {
28      String toEmails = getEmails(n.getInfo().getDevice().getId());
29      StringBuilder subject = new StringBuilder();
30      if (n.getLevel() == NotifyLevel.EMERGENCY) {
31        subject.append("[Emergência] ");
32      } else {
33        subject.append("[Alerta] ");
34      }
35      if (n.isRollRisk() && (n.getLevel() == NotifyLevel.EMERGENCY)) {
36        subject.append("Tombamento");
37      } else if (n.isRollRisk() && n.isSpeedRisk()) {
38        subject.append("Inclinação e velocidade");
39      } else if (n.isRollRisk()) {
40        subject.append("Inclinação");
41      } else if (n.isSpeedRisk()) {
42        subject.append("Velocidade");
43      }
44      subject.append("
45  (").append(n.getInfo().getDevice().getPlate()).append(")");
46      StringBuilder sb = new StringBuilder();
47         /* Adiciona a data e hora do evento, placa do veículo,
48          * id do dispositivo, inclinação e velocidade */
49         [...]
50      EmailUtils.sendEmail(toEmails, subject.toString(), sb.toString());
51    }
52  }
```

Fonte: elaborado pelo autor.

O método `sendNotifications` é responsável por montar e enviar o e-mail de notificação. O e-mail de notificação é enviado para todos os usuários associados ao dispositivo, conforme o nível de notificação configurado para cada usuário. Um tombamento ou velocidade acima de 120km/h, é considerada uma notificação de emergência, para outros riscos considera-se uma notificação de alerta.

No Quadro 16 é apresentado o código responsável pela criação do mapa, com a utilização do Google Maps API JavaScript. Na linha 8 é criado o objeto de mapa, passando como parâmetro o objeto da `DIV` onde o mapa será desenhado. O mapa é centralizado com base na posição solicitada pelo navegador. Também é criado um marcador na linha 9, a posição deste marcador pode ser alterada através da propriedade `position`.

Quadro 16 - Criação do mapa

```
 1  function initializeMap(latLng) {
 2    var options = {
 3      zoom: 15,
 4      center: latLng,
 5      mapTypeId: google.maps.MapTypeId.ROADMAP
 6    };
 7    var divMap = $window.document.getElementById("divMap");
 8    $scope.map = new google.maps.Map(divMap, options);
 9    $scope.marker = new google.maps.Marker({
10      position: latLng,
11      map: $scope.map
12    });
13  }
```

Fonte: elaborado pelo autor.

No Quadro 17 é apresentado o código responsável pela inicialização e exibição da inclinação do dispositivo com WebGL. Foi adotado o algoritmo LERP para calcular as interpolações lineares entre um ponto inicial e final conforme o decorrer do tempo. A utilização do LERP se faz necessária para criar o movimento de transição entre os ângulos de inclinação, suavizando a transição. O método `drawScene` passado por parâmetro para o método `requestAnimationFrame` sempre será chamado quando o navegador atualizar o quadro de animação.

Quadro 17 - Desenho com WebGL

```
 1  function startWebGl() {
 2    canvas = document.getElementById("glcanvas");
 3    initWebGL(canvas);
 4    if (gl) {
 5      [...]
 6      initShaders();
 7      initBuffers();
 8      requestAnimationFrame(drawScene);
 9    }
10  }
11
12  function lerp(a, b, f) {
13      return a + f * (b - a);
14  }
15
16  var lastTime = 0;
17  function drawScene(now) {
18    now *= 0.001
19    var deltaTime = now - lastTime;
20    lastTime = now;
21    [...]
22    distance += deltaTime / transictionTime;
23    var roll = lerp(rollInitial, rollFinal, distance);
24    mvRotate(-1*roll, [0, 0, 1]);
25    [...]
26    mvTranslate([0, 0.9, -0.5]);
27    [...]
28    requestAnimationFrame(drawScene);
29  }
```

Fonte: elaborado pelo autor.

A biblioteca `BDCCParallelLines` é responsável por desenhar linhas paralelas em objetos de mapas do Google Maps. A biblioteca foi adaptada para permitir o uso de duas cores, uma para cada linha. No Quadro 18 é apresentado como a biblioteca foi utilizada, a variável `coords` recebe uma lista de coordenadas, já as variáveis `rollColor` e `speedColor` possuem o valor da cor para identificar o nível de inclinação e velocidade. Os três últimos parâmetros do construtor da classe `BDCCParallelLines` recebe o valor da espessura da linha, opacidade e distância entre as linhas, respectivamente. A linha é desenhada no mapa ao definir o objeto do mapa para a variável `poly` (linha 4).

Quadro 18 - Utilização da bilbioteca BDCCParallelLines

```
1  function putPolyLine(coords, rollColor, speedColor) {
2    var poly = new BDCCParallelLines(coords, rollColor, speedColor, 4, 1.0,
3  0.5);
4    poly.setMap($scope.map);
5  };
```

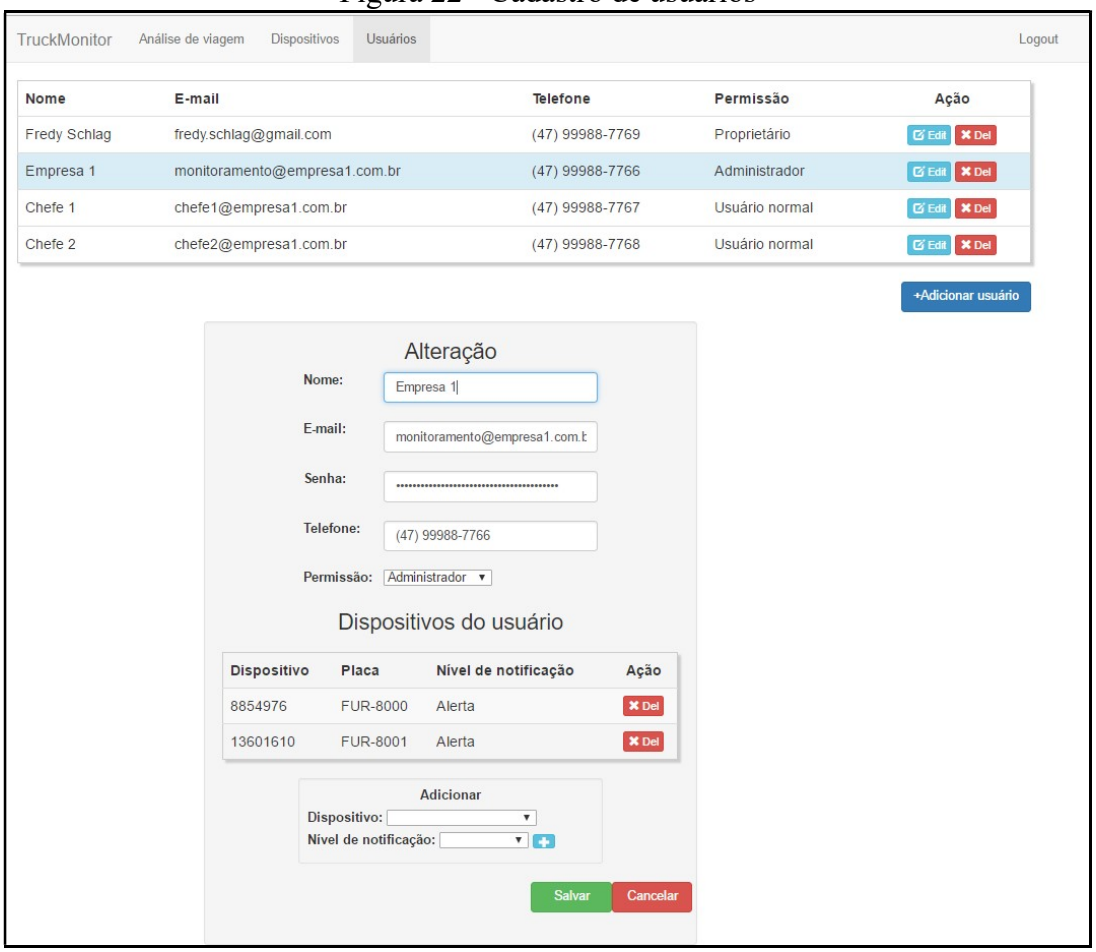
Fonte: elaborado pelo autor.

3.3.4 Operacionalidade da implementação

Inicialmente, o proprietário do sistema, o qual fornece os dispositivos deve cadastrar os dispositivos no servidor Web. O cadastro do dispositivo consiste em informar o ID do ESP8266 juntamente com a placa do veículo em que o dispositivo será instalado. Após o

cadastro de dispositivos, deve-se cadastrar os usuários, concedendo a permissão de `Administrador` ou `Usuário normal`. Durante o cadastro do usuário, o `Proprietário` ou `Administrador` podem relacionar os dispositivos ao usuário juntamente com o nível de notificação desejado. A Figura 22 apresenta a tela de cadastro de usuários, nela são apresentados apenas os usuários cujo o usuário autenticado possui permissão de alteração. O usuário terá acesso apenas aos dispositivos associado a ele.

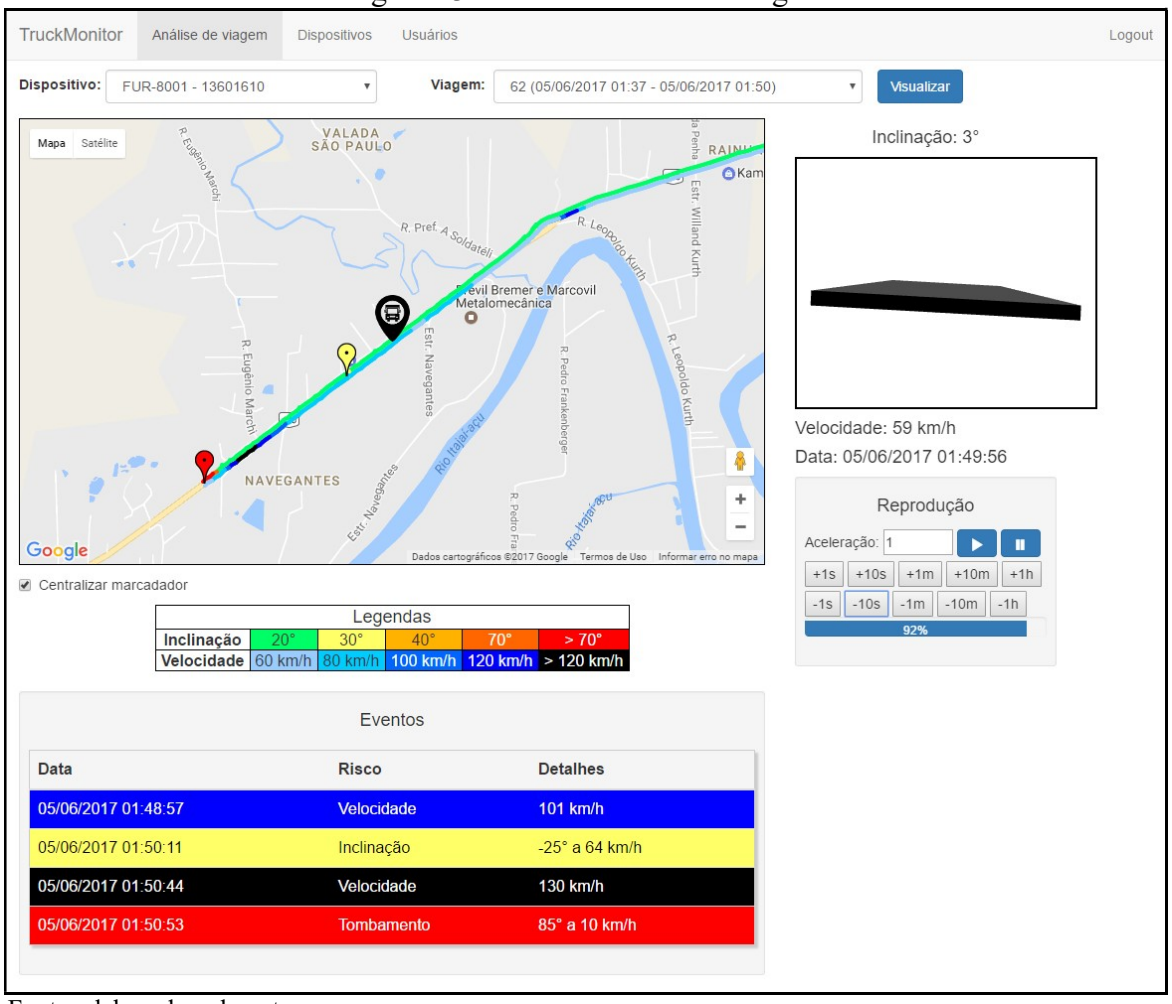
Figura 22 - Cadastro de usuários



Fonte: elaborado pelo autor.

A Figura 23 apresenta a tela de análise de viagem, que é a principal tela do protótipo. Ao abrir a tela, apenas são exibidos os campos do dispositivo, viagem e mapa. Após selecionar um dispositivo e sua respectiva viagem, são apresentadas as outras informações e inicia a reprodução do trajeto realizado pelo veículo. Para exibir o local atual do veículo utilizou-se um marcador com um ícone de caminhão no mapa. Na direita da tela, a inclinação do MPU6050 é exibida de forma textual e gráfica, além da velocidade e horário conforme reprodução. Abaixo da informação de data e hora da reprodução, há controles para manipular a reprodução veículo no trajeto.

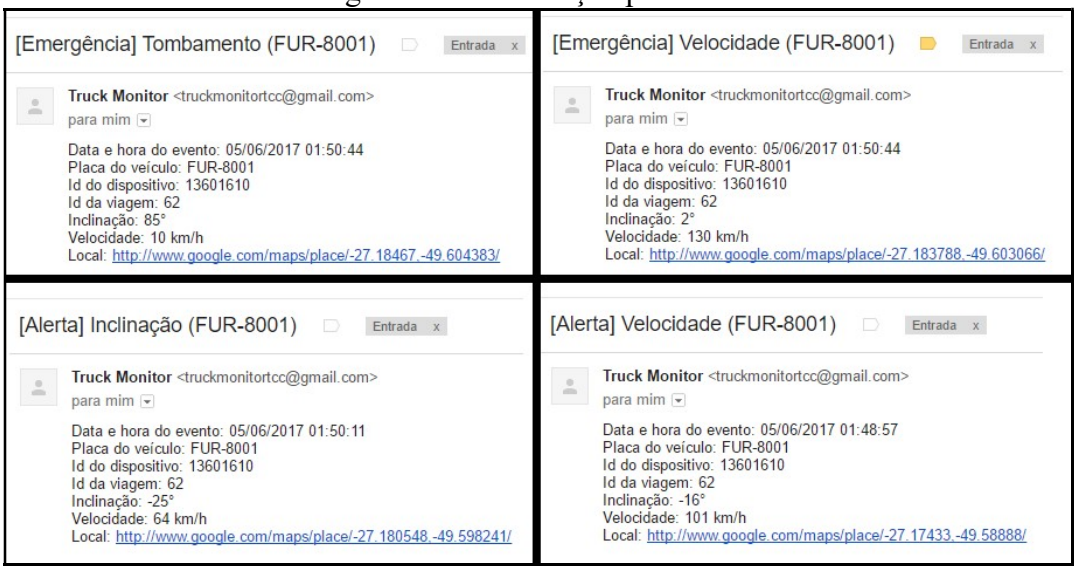
Figura 23 - Tela de análise de viagem



Fonte: elaborado pelo autor.

No trajeto exibido no mapa, há duas linhas, uma para representar a velocidade e outra para representar a inclinação através da cor. Para cada evento de inclinação é criado um marcador no mapa, com a mesma cor da legenda. Ao clicar nos marcadores ou no evento de risco, a reprodução é posicionada no evento selecionado. A Figura 24 apresenta os e-mails de notificações enviados pelo servidor Web aos usuários associados ao dispositivo. A notificação traz informações como data e hora do evento, placa do veículo, identificação do dispositivo, identificação da viagem, inclinação, velocidade e um link para o Google Maps com um marcador no local do evento.

Figura 24 - Notificação por e-mail

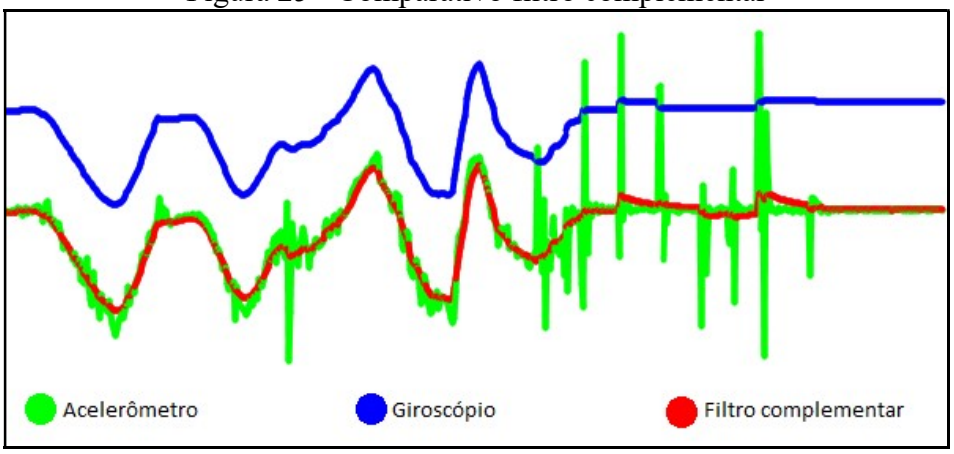


Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

Inicialmente, seria utilizado apenas os dados do acelerômetro para detectar a inclinação do veículo. Porém, com alguns testes realizados em veículo de passeio, percebeu-se uma grande quantidade de ruídos. Para minimizar os ruídos, combinou-se os dados do acelerômetro e giroscópio, aplicando o filtro complementar. Na Figura 25 é apresentado um teste de leitura do acelerômetro e giroscópio, juntamente com os dados combinados. Uma pequena vibração do acelerômetro faz com que o ângulo de inclinação calculado somente com os dados do acelerômetro resulte em uma inclinação falsa, porém, combinando com os dados do giroscópio, a resultado da inclinação ficou bem mais preciso.

Figura 25 - Comparativo filtro complementar



Fonte: elaborado pelo autor.

Para aplicar o filtro complementar, foi necessário implementar a interrupção de 12,5 milissegundos para garantir uma boa precisão. A frequência de processamento do ESP8266 também foi alterada de 80MHz para 160MHz, com o objetivo de diminuir o tempo de

intervalo entre as interrupções. O uso da interrupção apresentou problemas ao utilizar alguma função do sistema de arquivos ou WebServer do ESP8266, causando a reinicialização do mesmo sem informar o real problema. Para contornar estes problemas desconhecidos, a interrupção passou a ser desabilitada e restaurada após o uso destas funções.

Para o teste do dispositivo foi utilizado um caminhão do modelo Scania P310 com carroceria para suínos. Os testes no caminhão foram realizados em vários trajetos, incluindo rodovias federais e estradas do interior do Alto Vale de Santa Catarina. Foram percorridos aproximadamente 250 quilômetros, porém, por questões de segurança, não foi realizada nenhuma manobra arriscada. O maior ângulo de inclinação do caminhão foi de 15º e a velocidade máxima foi de 84 km/h. No caso em que a velocidade ultrapassou 80 km/h o servidor Web enviou a notificação por e-mail conforme configuração realizada no servidor.

Para testar a notificação de alerta da inclinação, foi realizado um teste inclinando o dispositivo manualmente a um ângulo superior à 20º e para o tombamento o dispositivo foi inclinado a um ângulo superior a 70º. Durante o trajeto realizado pelo caminhão, o MPU6050 era reiniciado devido a alguma falha não descoberta na alimentação do mesmo, desta forma, a comunicação entre o ESP8266 e MPU6050 era afetada, fazendo com que os dados do MPU6050 não fossem registrados corretamente. Para contornar este problema, o ESP8266 era reiniciado.

Nos pontos onde não havia sinal da rede GSM, o dispositivo armazenou os dados no sistema de arquivos e enviou ao servidor quando o sinal foi reestabelecido. Devido a compactação dos dados do dispositivo, é possível armazenar até 8 horas de informações do MPU6050 e GPS no sistema de arquivos do ESP8266. A compactação dos dados também otimizou o tempo de envio das informações para o servidor Web, em média é realizado um envio para o servidor Web a cada 30 segundos. A média do tempo de envio do dispositivo para o servidor até o retorno do mesmo é de 3 segundos com um bom sinal de GSM. Um ponto negativo do SIM808 é a questão de possuir apenas uma Serial, desta maneira, no momento que está enviado uma requisição para o servidor Web, não é possível coletar os dados do GPS.

As informações do GPS providas pelo SIM808 tiverem uma ótima precisão em relação a posição geográfica e velocidade. Em média, o GPS estabelece o sinal entre satélites em até 30 segundos em um campo aberto. Os testes do GPS foram realizados em dois módulos SIM808, porém, um módulo retornou as coordenadas no formato `DD.DDDDDD` e outro no formato `DDMM.MMMMMM`. Inicialmente apenas o primeiro formato era tratado, porém, para tratar

o segundo formato, o mesmo foi convertido para o primeiro formato. Para tentar configurar o formato de retorno, tentou-se atualizar o firmware, porém, não surtiu efeito.

No início dos testes do dispositivo, tentou-se alimentar o módulo SIM808 com uma fonte de 500mA, porém, o mesmo era reiniciado durante o envio dos dados para o servidor Web ou na coleta de dados do GPS. Para resolver o problema nos testes iniciais, utilizou-se uma fonte de 2,5A, sendo que no caminhão, o dispositivo foi alimentado pela própria bateria do caminhão, que possui uma amperagem superior.

Embora o servidor Web não verifica a relação entre inclinação, velocidade e condição da rodovia ao disparar uma notificação de alerta, ele apresenta uma visualização clara do trajeto percorrido. Permitindo que o usuário verifique se os pontos com maior inclinação ocorreram em uma curva ou uma reta, verificando também a velocidade do veículo no ponto, podendo assim concluir se o motorista agiu de forma agressiva ou não. Para exibir o trajeto percorrido, utilizou-se o recurso de `polyline` do Google Maps API de forma paralela, pois o serviço de direções da biblioteca suporta no máximo 23 pontos de referência. Com o serviço de direções não seria possível projetar os desvios do trajeto com precisão, nem as linhas paralelas do trajeto.

O Quadro 19 apresenta um comparativo entre as principais características dos trabalhos correlatos e do trabalho desenvolvido.

Quadro 19 - Comparativo entre trabalhos correlatos e o trabalho desenvolvido

Características	Trabalhos correlatos			Trabalho desenvolvido
	Vallejo et al. (2013)	Mestria (2016)	Cardoso, Costa e Monteiro (2014)	
Dispositivo IoT	X		X	X
Geolocalização	X		X	X
Mede a velocidade	X		X	X
Mede a inclinação		X		X
Detecta o tombamento				X
Disponibiliza visualização da rota percorrida em plataforma Web	X		X	X
Evita tombamentos		X		
Ajuda na prevenção de acidentes	X	X		X
Módulo utilizado	PIC24FJ64A	Não informado	Arduino	ESP8266

Fonte: elaborado pelo autor.

Observa-se no Quadro 19 que o trabalho desenvolvido possui quase todas as funcionalidades dos trabalhos correlatos, se destacando pela medição da inclinação combinada com a visualização no servidor Web. O trabalho de Cardoso, Costa e Monteiro (2014) se destaca pelo baixo custo e pela interface amigável provida pelo GC-Tracker. Já o trabalho de Vallejo et al. (2013) também se destaca pelo baixo custo e pelo monitoramento de acelerações do veículo, podendo assim auxiliar na prevenção de acidentes. O trabalho de Mestria (2016) se destaca por evitar de fato um tombamento, porém, não possui nenhuma conexão com Internet.

4 CONCLUSÕES

O trabalho desenvolvido atendeu o objetivo de realizar o monitoramento da agressividade do motorista de caminhão, monitorando a velocidade e inclinação do veículo. A utilização do ESP8266 demonstrou um ótimo desempenho na coleta de dados do GPS (SIM808) e do MPU6050. O sistema de arquivos de 4MB e a frequência de processamento de 160MHz se destacam em relação a outros microcontroladores da mesma faixa de preço.

As notificações geradas pelo protótipo mostraram-se assertivas, porém, há casos em que o motorista precise realizar uma manobra arriscada para evitar um acidente. Desta maneira, as notificações de inclinação devem ser avaliadas pelo usuário responsável pelo caminhão. Se as manobras arriscadas se repetem em um curto espaço de tempo, o usuário pode concluir que o motorista está dirigindo de forma agressiva.

A notificação de velocidade é mais assertiva, pois dificilmente haverá justificava para excesso de velocidade. A notificação de tombamento também é assertiva, quando o dispositivo não é danificado e há sinal de GSM. Por meio desta notificação o usuário toma conhecimento rapidamente do acidente, quando ocorre um tombamento, é possível avaliar a velocidade anterior do caminhão, desta maneira o usuário pode concluir se o motorista estava dirigindo de forma agressiva.

O *framework* Spark apresentou um ótimo desempenho para tratar as requisições do servidor Web. A configuração do *framework* consiste em apenas duas variáveis de ambiente, uma para definir a porta HTTP e outra para definir a URL JDBC do banco de dados, desta maneira, foi possível executar a aplicação na nuvem da Heroku como também executar local. A Heroku também disponibiliza um aplicativo para leitura de logs da aplicação, estes logs auxiliaram na identificação de problemas do dispositivo, estes logs também são utilizados para identificar os tempos das requisições e principais informações da mesma.

As ferramentas utilizadas para a especificação e desenvolvimento do protótipo atenderam todas as necessidades. A utilização da nuvem da Heroku para o servidor Web e banco de dados PostgreSQL teve um ótimo desempenho a um preço acessível. O custo total do protótipo também ficou acessível em relação a rastreadores veiculares. Sendo que a substituição de módulos pode ser realizada com pequenas alterações no código fonte.

A principal vantagem do dispositivo é que não depende de rede Wi-Fi, pois utiliza a rede GSM para o envio dos dados, porém, isto impacta em custo para manter um plano de dados. O dispositivo também há algumas limitações, pois o mesmo não detecta a parada do caminhão, desta maneira os dados são enviados mesmo com o caminhão parado ou desligado.

Para ligar e desligar o dispositivo com o caminhão, é necessário realizar a ligação na ignição do veículo.

4.1 EXTENSÕES

Para dar continuidade ao trabalho, sugere-se as seguintes extensões:

a) integrar informações providas pelo veículo por meio do protocolo OBDII;
b) coletar informações sobre o clima, com o objetivo de verificar o veículo está circulando sob chuva;
c) cruzar os dados do dispositivo com os dados das rodovias, com o objetivo de verificar a velocidade do veículo com o limite de velocidade da rodovia;
d) analisar os dados do dispositivo com o objetivo de traçar o perfil do motorista;
e) implementar camada de segurança no dispositivo para o envio dos dados para o servidor Web;
f) utilizar outros módulos de GPS e GSM de forma separada, tornando possível a captura de dados do GPS durante o envio de dados para o servidor;
g) utilizar mais de um acelerômetro para identificar torção da carroceria;
h) implementar suporte a tomada de ações sobre o dispositivo via servidor, por exemplo, desligar um relê.

REFERÊNCIAS

ACROBOTIC. **Getting Started With The ESP8266 ESP-12E Development Board**. 2015. Disponível em: <http://learn.acrobotic.com/tutorials/post/esp8266-getting-started>. Acesso em: 11 set. 2016.

ALVES, Sérgio. **A Matemática do GPS.** Disponível em: <ftp://labattmot.ele.ita.br/ele/jacques/CursoNaveg/gps.pdf>. Acesso em: 12 mai. 2017.

BACCHIERI, Giancarlo; BARROS, Aluísio J. D.. Acidentes de trânsito no Brasil de 1998 a 2010: muitas mudanças e poucos resultados. **Revista Saúde Pública,** Pelotas, v. 45, n. 5, p.949-963, 04 ago. 2011. Disponível em: <http://www.scielo.br/pdf/rsp/v45n5/2981>. Acesso em: 22 ago. 2016.

BÁSSORA, Luiz Antonio; GOMES, Malcoln Vellani. **Sensoriamento Inercial de um birrotor**. 2014. Disponível em: <https://uspdigital.usp.br/siicusp/cdOnlineTrabalhoVisualizarResumo?numeroInscricaoTrabalho=1376&numeroEdicao=22>. Acesso em: 10 mai. 2017.

BELLINI, Danilo. **Os riscos da prática imprudente de "quebrar asa"**. 2016. Disponível em: <https://www.sontracargo.com.br/blog/?p=818>. Acesso em: 12 set. 2016.

BUENO, Aline Grotewold; ROMANO, Rodrigo Alvite. **Filtro complementar aplicado a medida de inclinação de plataformas móveis**. 2011. Disponível em: <http://maua.br/files/122014/filtro-complementar-aplicado-a-medida-de-inclinacao-de-plataformas-moveis.pdf>. Acesso em 10 mai. 2017.

CAMPOS, Augusto. **ESP8266:** Comandos AT. 2015. Disponível em: <http://br-arduino.org/2015/11/esp8266-comandos-at.html>. Acesso em: 26 out. 2016.

CARDOSO, Jacques Stelzer; COSTA, José Renato Magalhães da; MONTEIRO, Roberto Sampaio. **Aplicação do rastreador uller no monitoramento privado de veículos.** 2014. 132 f. TCC (Graduação) - Curso de Engenharia Elétrica, Universidade Estácio de Sá, Rio de Janeiro, 2014. Disponível em: <http://www.renatocosta.com.br/arquivos/TCC_ULLER_PXI.pdf>. Acesso em: 15 ago. 2016.

COCKINGS, Chris. **GSM AT Command Set**. 2001. Disponível em: < http://www.zeeman.de/wp-content/uploads/2007/09/ubinetics-at-command-set.pdf>. Acesso em: 26 out. 2016.

CURVELLO, André. **Apresentando o módulo ESP8266**. 2015. Disponível em: <http://www.embarcados.com.br/modulo-esp8266/>. Acesso em: 25 out. 2016.

DINIZ, Victor. **Hardware:** entenda o Giroscópio. 2010. Disponível em: <http://ipodschool.com/hardware-entenda-o-giroscopio/>. Acesso em: 09 mai. 2017.

JUNIOR, Luiz Anselmo. **Análise da comunicação I²C entre um Arduino e um sensor MPU-6050**. 2015. Disponível em: <http://www.phph.com.br/instrumentacao-menu/instrumentacao-eletronica/item/131-analise-da-comunicacao-i-c-entre-um-arduino-e-um-sensor-mpu-6050.html>. Acesso em: 17 mai. 2017.

KOLBAN, Neil. **Kolban's Book on the ESP8266**. 2016. Disponível em: <https://leanpub.com/ESP8266_ESP32>. Acesso em: 11 set. 2016.

MESTRIA (Org.). **MX150.** 2016. Disponível em: <http://mestria.com.br/produtos/mx150-y-para-carreta-basculante>. Acesso em: 16 ago. 2016.

NUNES, Emanoel Júnior da Silva, NAKAI, Érica, BARROS, Pedro Paulo da Silva. **Apontamentos de aula:** Sistema Global de Posicionamento (GPS). Piracicaba, 2013. Disponível em: <http://www.leb.esalq.usp.br/disciplinas/Topo/leb450/Fiorio/APOSTILA_GPS_pdf.pdf>. Acesso em: 12 set. 2016.

PAMCARY (Brasil) (Org.). **Um diagnóstico de acidentes de caminhões.** São Paulo: Pamcary, 2007. Disponível em: <http://www.vias-seguras.com/content/download/302/1542/file/Um diagnostico Acidentes de caminhões.pdf>. Acesso em: 03 set. 2016.

PAULA, Fabio Oliveira. **Sensores IMU:** Uma Abordagem Completa – Parte 1. 2015. Disponível em: <http://www.decom.ufop.br/imobilis/sensores-imu-uma-abordagem-completa-parte-1/>. Acesso em: 11 set. 2016.

SIMCOM. **SIM800 Series:** AT Command Manual. 2015a. Disponível em: < http://simcom.ee/documents/SIM808/SIM800%20Series_AT%20Command%20Manual_V1.09.pdf >. Acesso em: 12 set. 2016.

SIMCOM. **SIM808:** Hardware Design. 2015b. Disponível em: < http://simcom.ee/documents/SIM808/SIM808_Hardware%20Design_V1.02.pdf >. Acesso em: 16 mai. 2017.

SOUZA, Fábio. **Arduino - Interface com acelerômetro e giroscópio**. 2015. Disponível em: <https://www.embarcados.com.br/arduino-acelerometro-giroscopio/>. Acesso em: 02 jul. 2015.

THOMSEN, Adilson. **Acelerômetro e Giroscópio MPU6050**. 2015. Disponível em: <http://www.arduinoecia.com.br/2015/04/acelerometro-giroscopio-mpu-6050.html>. Acesso em: 09 mai. 2017.

TILKOV, Stefan. **Uma rápida Introdução ao REST**. 2008. Disponível em: <https://www.infoq.com/br/articles/rest-introduction>. Acesso em 25 mai. 2017.

TORRES, Henrique. **Sensores Inerciais - Parte 1**. 2014. Disponível em: <https://www.embarcados.com.br/sensores-inerciais-parte-1/>. Acesso em: 23 mai. 2017.

VALLEJO, Alexandre Biscaro, OROZCO, Eduardo Cagnacci, LEITE, Felipe Riso Bezerra, CARLET, Rafael Del Col. Safe driver: monitoramento rodoviário de veículos comerciais. In: CONGRESSO NACIONAL DE INICIAÇÃO CIENTÍFICA, 13, 2013, São Paulo. **Anais**. São Paulo: Semesp, 2013. 4 p. Disponível em: <http://conic-semesp.org.br/anais/files/2013/trabalho-1000015945.pdf>. Acesso em: 16 ago. 2016.

WAISELFISZ, Julio Jacobo. **Mapa da violência 2012:** caderno complementar 2: Acidentes de Trânsito. São Paulo: Instituto Sangari, 2012. Disponível em: <http://mapadaviolencia.org.br/pdf2012/mapa2012_transito.pdf>. Acesso em: 03 set. 2016.

WORLD HEALTH ORGANIZATION. **The top 10 causes of death**. 2014. Disponível em: <http://www.who.int/mediacentre/factsheets/fs310/en/>. Acesso em: 03 set. 2016.

APÊNDICE A – Casos de uso

O caso de uso `UC01 – Consultar dispositivos` (Quadro 20) demonstra como o ator `Usuário` consulta os seus dispositivos.

Quadro 20 - Detalhamento do UC01

Número	01
Caso de uso	Consultar dispositivos
Ator	Usuário, Administrador e Proprietário
Pré-requisitos	1. Usuário autenticado
Cenário principal	1. O usuário acessa a página de dispositivos 2. O servidor exibe a lista de dispositivos associados ao usuário autenticado

Fonte: elaborado pelo autor.

O caso de uso `UC02 – Consultar dados do dispositivo` (Quadro 21) demonstra como o ator `Usuário` consulta os dados dos seus dispositivos.

Quadro 21 - Detalhamento do UC02

Número	02
Caso de uso	Consultar dados do dispositivo
Ator	Usuário, Administrador e Proprietário
Pré-requisitos	1. Usuário autenticado
Cenário principal	1. O usuário acessa a página de Análise de viagem 2. O usuário seleciona o dispositivo 3. O usuário seleciona a viagem 4. O usuário clica em Visualizar 5. O servidor exibe os dados do dispositivo em relação a viagem selecionada

Fonte: elaborado pelo autor.

O caso de uso `UC0C – Manter usuários` (Quadro 22) demonstra como o ator `Administrador` cadastra um novo usuário.

Quadro 22 - Detalhamento do UC03

Número	03
Caso de uso	Manter usuários
Ator	Administrador e Proprietário

Pré-requisitos	1. Usuário administrador autenticado
Cenário principal	1. O administrador acessa a página de Usuários 2. O servidor exibe a lista de usuários cadastrados 3. O administrador clica em Adicionar usuário 4. O administrador informa o nome do novo usuário 5. O administrador informa o e-mail do novo usuário 6. O administrador informa a senha do novo usuário 7. O administrador informa o telefone do novo usuário 8. O administrador informa o nível de permissão do novo usuário 9. O administrador adiciona os dispositivos para o novo usuário 10. O administrador clica em Salvar 11. O servidor exibe a lista de usuários cadastrados, juntamente com o usuário adicionado
Cenário alternativo	No passo 2, se o administrador deseja alterar um usuário: 2.1. O administrador clica no ícone de alteração do usuário 2.2. O administrador alterar os dados necessários 2.3. Retorna para o passo 10 do cenário principal

Fonte: elaborado pelo autor.

O caso de uso `UC04 – Manter dispositivos` (Quadro 23) demonstra como o ator `Proprietário` cadastra um novo dispositivo.

Quadro 23 - Detalhamento do UC04

Número	04
Caso de uso	Manter dispositivos
Ator	Proprietário
Pré-requisitos	1. Usuário proprietário autenticado
Cenário principal	1. O proprietário acessa a página de dispositivos 2. O servidor exibe os dispositivos cadastrados 3. O proprietário clica em Adicionar dispositivo 4. O proprietário informa o id do dispositivo 5. O proprietário informa a placa do veículo 6. O proprietário clica em Salvar 7. O servidor exibe a lista de dispositivos, juntamente com o dispositivo adicionado

Cenário alternativo	No passo 2, se o administrador deseja alterar um usuário: 2.1. O administrador clica no ícone de alteração do dispositivo 2.2. O administrador alterar os dados necessários 2.3. Retorna para o passo 6 do cenário principal

Fonte: elaborado pelo autor.

O caso de uso `UC05 – Registrar dados do acelerômetro e GPS` (Quadro 24) demonstra como o ator `Dispositivo` registra os dados do acelerômetro e GPS.

Quadro 24 - Detalhamento do UC05

Número	05
Caso de uso	Registrar dados do acelerômetro e GPS
Ator	Dispositivo
Pré-condição	1. O módulo de GPS deve estar ligado 2. Deve possuir sinal da rede GSM
Cenário principal	1. O dispositivo coleta os dados do acelerômetro 2. O dispositivo coleta os dados do GPS 3. O dispositivo envia os dados para o servidor Web 4. O servidor Web retorna OK

Fonte: elaborado pelo autor.

APÊNDICE B – Componentes do dispositivo

O Quadro 25 apresenta o custo do protótipo, os componentes eletrônicos foram adquiridos na loja AliExpress.com.

Quadro 25 - Custo do protótipo

Componente	Valor
NodeMCU ESP8266	U$3,38
MPU6050	U$1,07
SIM808	U$19,44
LM2596	U$0,73
Chip SIM TIM	R$20,00
PostgreSQL (Heroku – Plano: Hobby Basic)	U$9,00/mês

Fonte: elaborado pelo autor.