



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Arthur Fellipe Roders

Sistema de Monitoramento via GPS para Frotas

Blumenau
2024

Arthur Felipe Roders

Sistema de Monitoramento via GPS para Frotas

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.

Orientador: Carlos Roberto Moratelli, Dr.

Blumenau

2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Roders, Arthur Fellipe
Sistema de Rastreamento via GPS para Frotas / Arthur
Fellipe Roders ; orientador, Carlos Roberto Moratelli,
2024.
50 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Campus Blumenau,
Graduação em Engenharia de Controle e Automação, Blumenau,
2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. GPS. 3. GSM.
4. Sistemas Embarcados. I. Moratelli, Carlos Roberto. II.
Universidade Federal de Santa Catarina. Graduação em
Engenharia de Controle e Automação. III. Título.

Arthur Fellipe Roders

Sistema de Monitoramento via GPS para Frotas

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “Engenheiro de Controle e Automação” e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 10 de Julho de 2024.

Banca Examinadora:

Prof. Carlos Roberto Moratelli, Dr.
Universidade Federal de Santa Catarina - Campus Blumenau

Prof. Adão Boava, Dr.
Universidade Federal de Santa Catarina - Campus Blumenau

Prof. Ebrahim Samer El Youssef, Dr.
Universidade Federal de Santa Catarina - Campus Blumenau

AGRADECIMENTOS

Aos professores do curso de Engenharia de Controle e Automação, que nos preparam para a vida profissional e ao mesmo tempo nos ajudam a moldar de forma positiva nosso lado pessoal.

À minha família, onde sem eles não seria possível chegar onde cheguei.

Aos meus amigos, que sempre me apoiaram, e nos momentos difíceis me ajudaram a superá-los.

"Se você não correr atrás, jamais irá alcançar os objetivos que almeja."
- (Anilton Roders)

RESUMO

No mundo moderno, os meios de transporte se tornaram indispensáveis. Com a globalização e a crescente necessidade de deslocamento e segurança dos colaboradores, empresas buscam por meio de veículos próprios permitir a locomoção de seus colaboradores para fins profissionais. Este trabalho busca desenvolver um sistema embarcado que visa auxiliar estas frotas veiculares empresariais de modo a evitar possíveis problemas, acompanhando o trajeto de um determinado veículo e analisando se está de acordo com as leis de trânsito vigentes no local. Para isto, este sistema faz a integração entre hardware, disposto por diversos componentes (GPS, módulo GSM, placa de controle), um serviço web, que irá armazenar e visualizar os dados obtidos. O projeto também conta com um sistema de envio de e-mails, avisando o responsável do setor em caso de desrespeito às diretrizes impostas pela empresa. Estas aplicações buscam aumentar a segurança para quem utiliza o veículo, bem como reduzir possíveis problemas para a empresa.

Palavras-chave: 1. Sistema Embarcado. 2. Integração. 3.GPS. 4.GSM.

ABSTRACT

Nowadays, means of transportation have become indispensable. With globalization and the growing need for employee mobility and safety, companies use their own vehicles to enable the transportation of their employees for professional purposes. This work aims to develop an embedded system designed to assist these corporate vehicle fleets in avoiding potential problems by monitoring the route of a specific vehicle and analyzing its compliance with local traffic laws. To achieve this, the system integrates hardware, comprising various components (GPS, GSM module, control board), and a web service that will store and visualize the obtained data. The project also includes an email notification system, alerting the sector manager in case of non-compliance with company guidelines. These applications aim to increase safety for vehicle users and reduce potential issues for the company.

Keywords:

Keywords: 1. Embedded System. 2. Integration. 3. GPS. 4. GSM.

LISTA DE FIGURAS

Figura 1 – Diagrama de blocos do sistema.	15
Figura 2 – Diagrama de comunicação do projeto.	17
Figura 3 – Modelos da plataforma Arduino.	19
Figura 4 – Interface do software Arduino IDE.	20
Figura 5 – Exemplo da Constelação de Satélites da Rede GPS.	21
Figura 6 – Exemplo de Arquitetura da Rede GSM.	23
Figura 7 – Exemplo de Arquitetura da Rede GPRS.	24
Figura 8 – Ideia de funcionamento do MQTT.	25
Figura 9 – Funcionamento de um Sistema de Rastreamento Veicular.	26
Figura 10 – Sistema para rastreamento veicular.	28
Figura 11 – Modelo Arduino Mega 2560.	29
Figura 12 – Pinagem Arduino MEGA.	30
Figura 13 – Shield SIM900.	32
Figura 14 – Imagem da conexão serial entre SIM900 e Arduino.	33
Figura 15 – Fonte 9V 3A.	34
Figura 16 – Fonte para uso embarcado.	35
Figura 17 – Shield SIM900 com cartão SIM.	35
Figura 18 – Conexão do Módulo Neo-6M.	36
Figura 19 – Função loop().	37
Figura 20 – Função setup().	37
Figura 21 – Função displayInfo().	38
Figura 22 – Função email().	38
Figura 23 – Diagrama de funcionamento da plataforma Thinger.io	40
Figura 24 – Cadastro de Dispositivo na Plataforma Thinger.io	40
Figura 25 – Configuração da ferramenta <i>bucket</i> na Plataforma Thinger.io.	41
Figura 26 – Configuração da ferramenta <i>dashboard</i> na Plataforma Thinger.io.	41
Figura 27 – Protótipo Final.	43
Figura 28 – Dados populados no <i>bucket</i>	44
Figura 29 – Rota criada a partir dos dados obtidos via GPS.	45
Figura 30 – Rota criada a partir dos dados obtidos via GPS.	45
Figura 31 – Lista de e-mails recebidos por excesso de velocidade.	46
Figura 32 – Conteúdo do e-mail recebido ao ultrapassar as condições de velocidade.	46

LISTA DE TABELAS

Tabela 1 – Valores aproximados dos componentes utilizados no projeto.	46
---	----

LISTA DE ABREVIATURAS E SIGLAS

AMPS	<i>Advanced Mobile Phone System</i>
CSS	<i>Cascading Style Sheets</i>
GPRS	<i>General Packet Radio Service</i>
GPS	<i>Global Positioning System</i>
GSM	<i>Global System for Mobile</i>
HTML	<i>HyperText Markup Language</i>
IBGE	<i>Instituto Brasileiro de Geografia e Estatística</i>
IDE	<i>Integrated Development Environment</i>
IoT	<i>Internet of Things</i>
MMS	<i>Multimedia Messaging Service</i>
MQTT	<i>Message Queue Telemetry Transport</i>
NMT	<i>Nordic Mobile Telephone</i>
PHP	<i>PHP: Hypertext Preprocessor</i>
PPS	<i>Precise Positioning Service</i>
JSON	<i>Protocol JSON</i>
SGBD	<i>Sistema de Gerenciamento de Banco de Dados</i>
SMS	<i>Short Message Service</i>
SPS	<i>Standard Positioning Service</i>
SQL	<i>Structured Query Language</i>

SUMÁRIO

1	INTRODUÇÃO	13
1.1	OBJETIVO GERAL	13
1.2	OBJETIVOS ESPECÍFICOS	14
2	TRABALHOS RELACIONADOS	15
2.1	SISTEMA DE RASTREAMENTO VEICULAR EM TEMPO REAL ANTI-FURTO VIA GPS E GSM	15
2.2	SISTEMA DE RASTREAMENTO VEICULAR COM UTILIZAÇÃO DA NUVEM	16
3	TECNOLOGIAS UTILIZADAS	18
3.1	SISTEMAS EMBARCADOS	18
3.2	DESENVOLVIMENTO EMBARCADO	18
3.2.1	Arduino	18
3.2.2	IDE Arduino	20
3.3	GPS	21
3.4	COMUNICAÇÃO MÓVEL	22
3.4.1	GSM	22
3.4.2	GPRS	23
3.5	PROTOCOLO MQTT	24
3.6	RASTREAMENTO VEICULAR	25
4	PROPOSTA E IMPLEMENTAÇÃO DE UM SISTEMA DE RASTREAMENTO VEÍCULAR	27
4.1	REQUISITOS E ARQUITETURA PROPOSTA	27
4.2	HARDWARE	27
4.2.1	Arduino Mega 2560	28
<i>4.2.1.1</i>	<i>Alimentação</i>	<i>29</i>
<i>4.2.1.2</i>	<i>Pinout do Arduino Mega</i>	<i>30</i>
<i>4.2.1.3</i>	<i>Programação</i>	<i>31</i>
4.2.2	Shield GPRS SIM900	32
4.2.3	Módulo GPS Neo-6M	33
4.3	FIRMWARE	34
4.4	THINGER.IO	39
4.4.1	Configuração	39
5	RESULTADOS	43
5.1	PREPARAÇÃO PARA TESTE PRÁTICO	43
5.2	COLETA DE DADOS	44
5.3	POSICIONAMENTO	44
5.4	ALARME DE VELOCIDADE	45

5.5	CUSTOS DO PROJETO	46
6	CONCLUSÃO	47
	REFERÊNCIAS	48

1 INTRODUÇÃO

A locomoção é parte essencial da vida cotidiana. Ir ao trabalho, mercado, lojas e restaurantes se tornou comum. Existem diversos meios de locomoção, sendo particulares, públicos e até mesmo sob demanda, como aplicativos de transporte. Junta-se a isso o aumento nas vendas online, que demandam o transporte de mercadorias de um determinado local para o cliente causando um crescimento exacerbado da frota de veículos. Segundo o IBGE (Instituto Brasileiro de Geografia e Estatística), nos últimos 10 anos houve um aumento de quase 30 milhões de veículos em circulação no país. Focando somente no âmbito empresarial, estima-se que em 2018 houve um crescimento de 32 % para frotas comerciais dentro do território nacional (ESTADÃO... , 2023)

Com o aumento de carros circulando, por consequência o número de infrações de trânsito tem aumentado cada vez mais. Estima-se que em 2022 mais de 76 milhões de multas tenham sido aferidas no território brasileiro, onde mais de 35 milhões estão atreladas a multa de excesso de velocidade [(JORNAL... , 2023)]. Outro fator alarmante é o número de acidentes e óbitos no trânsito. Apesar de ter um leve declínio em alguns dos últimos anos, o Brasil ainda é o terceiro colocado no ranking de mais mortes por acidentes de trânsito no planeta [(VRUM... , 2023)].

Tendo em vista este cenário, busca-se uma solução para empresas que seja segura, rápida e capaz de analisar o comportamento de seus motoristas no trânsito. Para isso, a localização e velocidade do veículo são obtidas e alertas em tempo real são gerados. Desta forma, aumentado a segurança para a empresa, colaboradores e para outros usuários em geral.

O sistema proposto neste trabalho realiza a coleta de dados de localização e velocidade em tempo real por meio de coordenadas geográficas, além de identificar e reportar anomalias, como velocidades acima do permitido. Esses dados são retornados a um *backend*, onde é possível acessar as rotas percorridas e os dados de velocidade. O objetivo é reduzir os riscos no trânsito para os motoristas e evitar problemas legais, como multas, para as empresas.

1.1 OBJETIVO GERAL

O objetivo geral deste projeto visa o desenvolvimento de um sistema de monitoramento de frotas via GPS de baixo custo, baseado em microcontroladores. Com isto, busca-se uma forma de monitorar o posicionamento do motorista, bem como o seu comportamento durante o uso do veículo da empresa, assim visando evitar transtornos administrativos e também acidentes de trânsito.

1.2 OBJETIVOS ESPECÍFICOS

- a) Desenvolver um projeto de *hardware* e *software* buscando obter informações de localização e dados de viagem.
- b) Transmissão de dados em tempo real. Os dados serão processados através de um microcontrolador Arduino MEGA 2560.
- c) Integração com módulo GPS (Neo-6M) para obtenção dos dados posicionais e de movimentação.
- d) Utilização de módulo GPRS/GSM para o envio dos dados para a rede.
- e) Conectar o dispositivo IoT com um serviço na nuvem.
- f) Integrar os sistemas previamente citados para que o conjunto se comporte de forma esperada.
- g) Coleta dos dados via módulo GPS, processando-os no microcontrolador e enviando para a nuvem por meio do módulo GPRS/GSM SIM900, onde os dados estarão disponíveis em um *backend*.
- h) Envio de email para alertar o usuário de velocidade excessiva do veículo.

2 TRABALHOS RELACIONADOS

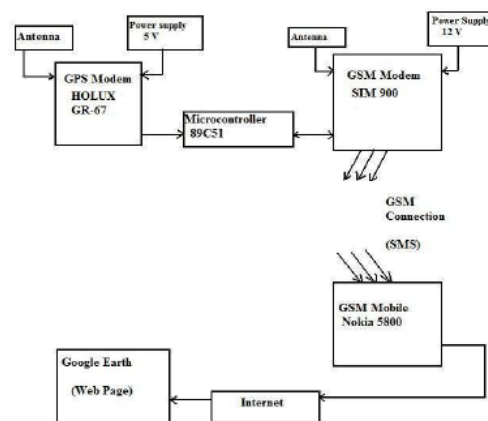
Esta seção busca mostrar alguns trabalhos existentes que tem relação com este projeto. A Seção 2.1 mostra a utilização da tecnologia GPS para o uso no rastreamento veicular. A Seção 2.2 busca mostrar a implementação das tecnologias de *cloud* para a implementação de um sistema de rastreamento veicular.

2.1 SISTEMA DE RASTREAMENTO VEICULAR EM TEMPO REAL ANTI-FURTO VIA GPS E GSM

O sistema de rastreamento de veículos em tempo real apresentado no artigo (MAURYA; SINGH, Ravi; SINGH, Ritu, 2012) é composto por um microcontrolador AT89C51, um módulo GPS e um modem GSM. O microcontrolador é programado para receber sinais do módulo GPS usado para determinar a localização do veículo e transmiti-los para um servidor remoto usando tecnologia GSM. O sistema é projetado para ser de baixo custo e fácil de instalar, tornando-o adequado para uma ampla gama de aplicações.

O sistema é capaz de rastrear a localização do veículo em tempo real e enviar alertas em caso de acesso não autorizado ou roubo. Quando o usuário envia uma solicitação para o número no modem GSM, o sistema envia automaticamente uma resposta indicando a posição do veículo em termos de latitude e longitude em tempo real. O sistema também pode ser usado para gerenciamento de frota, permitindo que os gerentes monitorem a localização e o status dos veículos em tempo real. A Figura 1 mostra o esquema de funcionamento do projeto, onde um módulo GPS está conectado a um microcontrolador e um módulo GSM. A partir da rede GSM o sistema se comunica com a internet, amostrando os dados em um mapa gerado via Google Earth.

Figura 1 – Diagrama de blocos do sistema.



Fonte: (MAURYA; SINGH, Ravi; SINGH, Ritu, 2012)

O artigo também discute a implementação do sistema, incluindo o circuito eletrônico, o software e a interface do usuário. O software é escrito em linguagem C e é usado para programar o microcontrolador. A interface do usuário é projetada para ser simples e fácil de usar, permitindo que os usuários solicitem a localização do veículo com facilidade.

Os resultados do estudo mostram que o sistema de rastreamento de veículos em tempo real é uma ferramenta eficaz para proteção contra roubo e gerenciamento de frota. O sistema é capaz de rastrear com precisão a localização de um veículo, fornecendo informações valiosas ao proprietário ou gerente. O sistema também é capaz de enviar alertas em caso de acesso não autorizado ou roubo, permitindo uma ação rápida. O sistema é uma solução econômica e confiável para rastreamento e monitoramento de veículos.

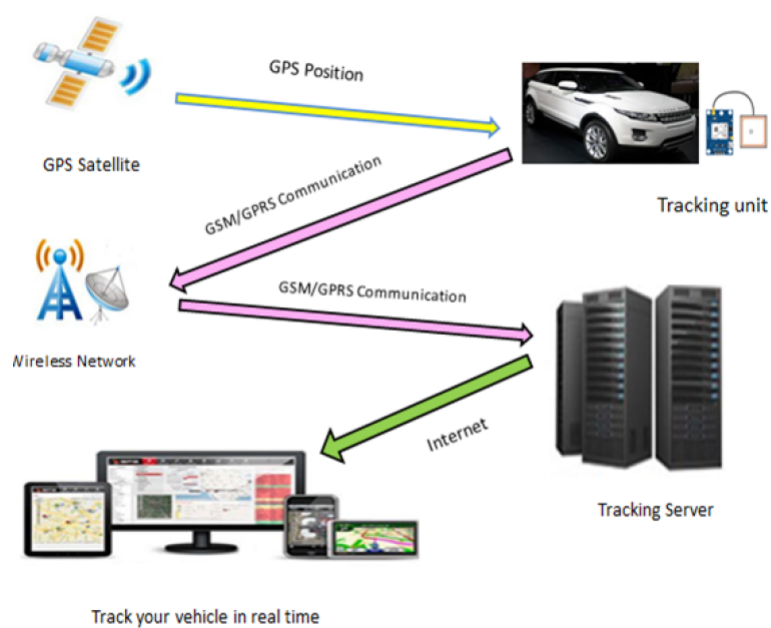
2.2 SISTEMA DE RASTREAMENTO VEICULAR COM UTILIZAÇÃO DA NUVEM

O sistema apresentado no artigo (MUSTAFA, 2019) apresenta um sistema de rastreamento de veículos baseado em nuvem que permite o monitoramento em tempo real da localização de veículos, reduzindo o custo de transmissão de dados para servidores. O sistema é composto por duas partes: um Arduino com GPS/GPRS que é instalado no veículo e é responsável por fornecer informações de rastreamento em tempo real e enviá-las para o servidor, e um servidor em nuvem que permite que o administrador verifique e rastreie a localização dos veículos. O sistema também possui um método inovador para enviar a localização do veículo apenas se ele estiver em movimento, reduzindo assim a quantidade de dados enviados. A Figura 2 mostra as etapas de funcionamento do projeto.

O sistema foi desenvolvido utilizando o Arduino UNO R3, GPS/GPRS SIM808, e sensores de combustível como sistema embutido no veículo. O sistema utiliza um servidor remoto e o Google Maps para exibir a localização dos veículos e usa HTML, CSS, SQL para banco de dados e PHP e JavaScript para a interface e processamento. O sistema também possui um método para falsificar os dados enviados do veículo para o servidor, permitindo que apenas pessoas autorizadas rastreiem os veículos.

Em conclusão, o sistema de rastreamento de veículos baseado em nuvem apresentado neste artigo é uma solução inovadora que permite o monitoramento em tempo real da localização de veículos, reduzindo o custo de transmissão de dados para servidores. O sistema utiliza um Arduino embutido com GPS/GPRS que é anexado ao veículo e um servidor em nuvem que permite que o administrador verifique e rastreie a localização dos veículos. O sistema também possui um método inovador para enviar a localização do veículo apenas se ele estiver em movimento, reduzindo assim a quantidade de dados enviados. Os resultados mostraram que o sistema proposto tem melhor desempenho em termos de quantidade de dados e consumo de energia, especialmente em estradas com tráfego intenso.

Figura 2 – Diagrama de comunicação do projeto.



Fonte: (MUSTAFA, 2019)

3 TECNOLOGIAS UTILIZADAS

Neste capítulo será demonstrada uma revisão teórica sobre os temas e tecnologias mais pertinentes ao desenvolvimento deste projeto. A revisão tem como principais tópicos: sistemas embarcados, desenvolvimento embarcado, tecnologias de posicionamento, comunicação móvel, *webservices* e rastreamento veicular;

3.1 SISTEMAS EMBARCADOS

Quando falamos em sistemas embarcados, falamos de cotidiano. Este tipo de tecnologia está presente no nosso dia-a-dia, desde o momento que acordamos. Seja pelo despertador de nosso celular, o passeio de carro, o trabalho no computador, o filme na televisão. Todos estes objetos citados anteriormente tem em sua composição sistemas embarcados, e este tipo de dispositivo não está presente somente no âmbito pessoal. Diversas áreas como medicina, aeronáutica e indústria utilizam de sistemas cyber-físicos, sendo estes definidos por Seshia et al.(LEE; SESHIA, 2017) como "a integração da computação com processos físicos cujo comportamento é definido tanto pelas partes cibernéticas quanto físicas do processo". Este tipo de sistema e muitos outros são sistemas embarcados.

De acordo com Li (LI; YAO, 2003), "sistemas embarcados são sistemas computacionais com *hardware* compacto e alta integração de *software* desenhados para praticar uma determinada função"onde

a palavra embarcado reflete o fato destes sistemas serem partes de outros sistemas maiores, onde múltiplos sistemas embarcados podem fazer parte de um sistema embarcado, porém também podendo um único sistema embarcado ser o sistema completo. (LI; YAO, 2003)

Muitos destes sistemas podem estar conectados entre si por meio de redes, recebendo e transmitindo dados entre si. Sistemas embarcados que possuem a capacidade de transmitir e enviar dados formam o conceito de Internet das Coisas (IoT), que cresce cada vez mais no nosso dia-a-dia e vêm se tornando parte essencial da nossa sociedade moderna. A fácil integração de sistemas devido a simplicidade de construção, a forma compacta de sua estrutura física, o baixo custo de implementação e o crescimento da Internet das Coisas (IoT) fizeram com que os sistemas embarcados se popularizassem de uma forma muito rápida nos últimos anos, impulsionando o desenvolvimento tecnológico cada vez mais.

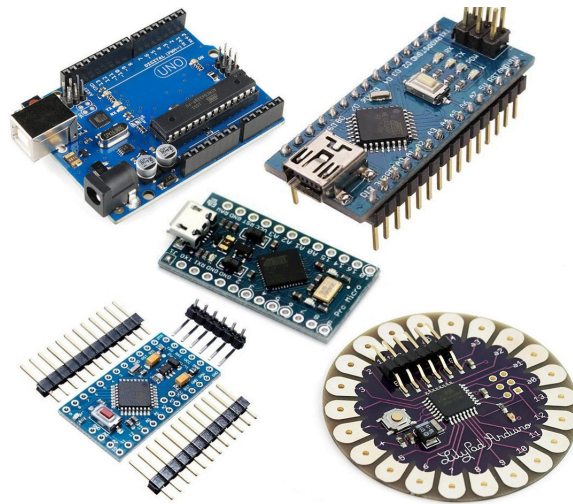
3.2 DESENVOLVIMENTO EMBARCADO

3.2.1 Arduino

A plataforma Arduino é uma plataforma de prototipagem eletrônica de código aberto (*open-source*) que foi projetada para tornar a criação de projetos eletrônicos mais

acessível, amigável e versátil (ARDUINO..., 2023). Foi desenvolvida em 2005 por um grupo do Interaction Design Institute Ivrea, na Itália, como uma plataforma de hardware de código aberto destinada a simplificar a criação de projetos interativos para que estudantes, designers e entusiastas desenvolvam uma ampla gama de dispositivos interativos.

Figura 3 – Modelos da plataforma Arduino.



Fonte: (ELETROGATE, 2023)

As placas Arduino são o componente físico da plataforma. Elas vêm em várias versões, cada uma com suas próprias características e especificações. Algumas das placas mais populares incluem o Arduino Uno, Arduino Mega, Arduino Nano, entre outras. Cada placa Arduino contém um microcontrolador, sendo este um chip de processamento que executa programas gravados em sua memória. Dentre as diversas características e benefícios que a plataforma traz, pode-se citar algumas importantes como:

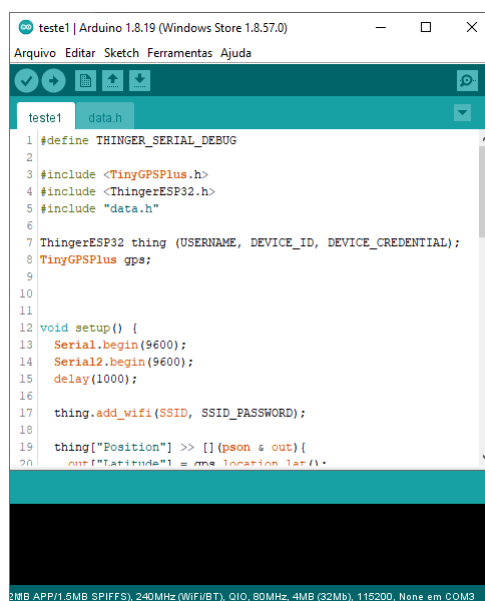
- **Facilidade de Uso:** A plataforma Arduino é conhecida por sua facilidade de uso, tornando-a acessível a iniciantes em eletrônica e programação. É uma ótima maneira de aprender os fundamentos da eletrônica.
- **Ampla Comunidade e Suporte:** Existe uma grande comunidade de usuários Arduino, com fóruns, tutoriais online e grupos de discussão que ajudam a resolver problemas e compartilhar conhecimento.
- **Versatilidade:** A plataforma Arduino pode ser usada para criar uma ampla gama de projetos, desde robôs e sistemas de automação residencial até dispositivos de arte interativa e wearables.
- **Código Aberto:** Tanto o hardware quanto o software Arduino são de código aberto, o que significa que você pode modificá-los e compartilhá-los livremente.
- **Custo Acessível:** As placas Arduino são relativamente econômicas, tornando a eletrônica acessível para um grande número de pessoas.

Devido a estas características, o uso da plataforma Arduino se torna cada vez maior para as aplicações que necessitam de prototipagem. Por outro lado, apesar de permitido o uso da plataforma para projetos comerciais, grande parte das suas aplicações não se destinam ao âmbito profissional devido ao alto custo em grande escala. Para o desenvolvimento de código, utiliza-se em grande parte dos projetos a Arduino IDE.

3.2.2 IDE Arduino

A IDE Arduino é um software usado para escrever, compilar e carregar código para as placas Arduino. Para manter a facilidade de utilização, a plataforma Arduino criou um ambiente gratuito para o desenvolvimento de códigos. Sendo aberta, é possível que outros usuários da plataforma criem bibliotecas que são compartilhadas com a comunidade. Isto permite que diversos *hardwares* além do Arduino possam ser programados com a utilização deste ambiente, como o ESP, o nodeMCU, entre outros. A Figura 4 mostra como é a atual IDE.

Figura 4 – Interface do software Arduino IDE.



```
teste1 | Arduino 1.8.19 (Windows Store 1.8.57.0)
Arquivo  Editar  Sketch  Ferramentas  Ajuda

teste1  data.h
1 #define THINGER_SERIAL_DEBUG
2
3 #include <TinyGPSPlus.h>
4 #include <ThingESP32.h>
5 #include "data.h"
6
7 ThingESP32 thing (USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
8 TinyGPSPlus gps;
9
10
11
12 void setup() {
13   Serial.begin(9600);
14   Serial2.begin(9600);
15   delay(1000);
16
17   thing.add_wifi(SSID, SSID_PASSWORD);
18
19   thing["Position"] >> [1] {person * out} {
20     out["Latitude"] = gps.location.lat();
21   }
22 }
```

2MB APP/1.5MB SPIFFS, 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 115200, None em COM3

Fonte: Autor.

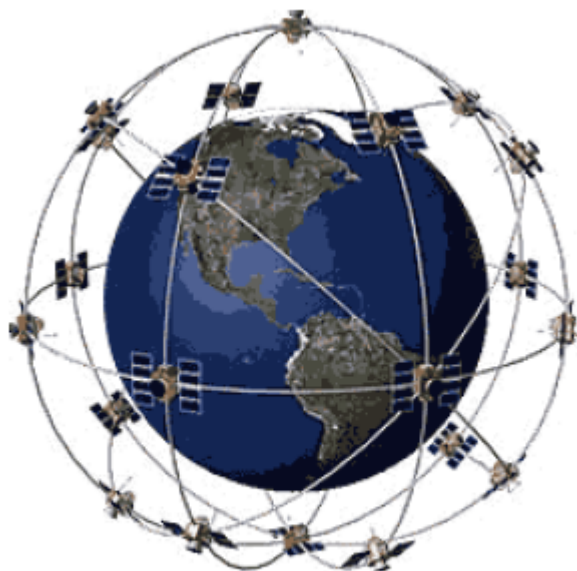
Tem como principal característica o uso das linguagens C e C++ para a programação. Além de permitir programar, a IDE Arduino permite fazer a compilação dos códigos, e também permite o carregamento deste código na placa de desenvolvimento. Além disso, ao carregar o código proveniente do software, o mesmo sobrescreve o código existente no microcontrolador, fazendo com que na próxima inicialização o microcontrolador já esteja com a última versão do código criado. Conta também com um monitor serial que permite visualizar mensagens de depuração ou saída do seu programa, sendo de suma importância

para poder entender o funcionamento do programa durante o processamento e permitindo a depuração de problemas.

3.3 GPS

Em grande parte da história humana, sempre se buscou métodos de localização que fossem mais adequados que o sistema utilizado desde os primórdios que se baseava na observação dos astros. O GPS (*Global Positioning System*) é um sistema de navegação por satélite que permite determinar a posição de um receptor GPS em qualquer lugar do mundo com precisão e em tempo real. Criado pelo departamento de defesa dos Estados Unidos como uma ferramenta militar, hoje a tecnologia é utilizada nas mais diversas áreas e nas mais diferentes aplicações, como sistemas de navegação para dispositivos móveis. Em termos de navegação, o GPS revolucionou a forma como as pessoas se orientam e se deslocam, permitindo que qualquer pessoa possa determinar sua posição geográfica com precisão em qualquer lugar do mundo e criando rotas otimizadas para determinadas tarefas, assim reduzindo custos e agilizando o processo. Isso teve um impacto significativo em áreas como transporte, logística, turismo e esportes (TEODOLINI, s.d.). A Figura 5 mostra um exemplo de constelação de satélites orbitando a Terra, de modo a cobrir grande parte da sua superfície.

Figura 5 – Exemplo da Constelação de Satélites da Rede GPS.



Fonte: (MARINO, s.d.)

O seu funcionamento se dá através da emissão de sinais de rádio dos satélites para o

receptor GPS. Cada satélite transmite um sinal que contém informações sobre sua posição e o tempo em que o sinal foi transmitido. Assim, o receptor GPS recebe os sinais de pelo menos quatro satélites e usa as informações contidas nos sinais para calcular sua posição geográfica, velocidade e altitude. Esses dados podem ser utilizados em aplicações como mapas digitais, bússolas, entre outros dispositivos de localização.

Este sistema de posicionamento pode ser classificado como PPS (*Precise Positioning System*) e SPS (*Standard Positioning System*), onde o PPS ainda é de uso restrito militar. O SPS é o serviço disponibilizado para o público geral e que permite as aplicações que existem no âmbito comercial.

O sistema divide-se em 3 partes: espacial, controle e usuário. O primeiro é composto por uma constelação de satélites (24 satélites) que orbitam a Terra em órbitas precisamente conhecidas de forma que existam, de qualquer parte da superfície terrestre, ao menos 4 satélites visíveis. A segunda utiliza estações terrestres para monitorar os satélites, garantindo o seu funcionamento correto e preciso. O terceiro são os receptores GPS que recebem os sinais dos satélites e calculam a posição geográfica, velocidade e altitude do usuário (MONICO, 2000) .

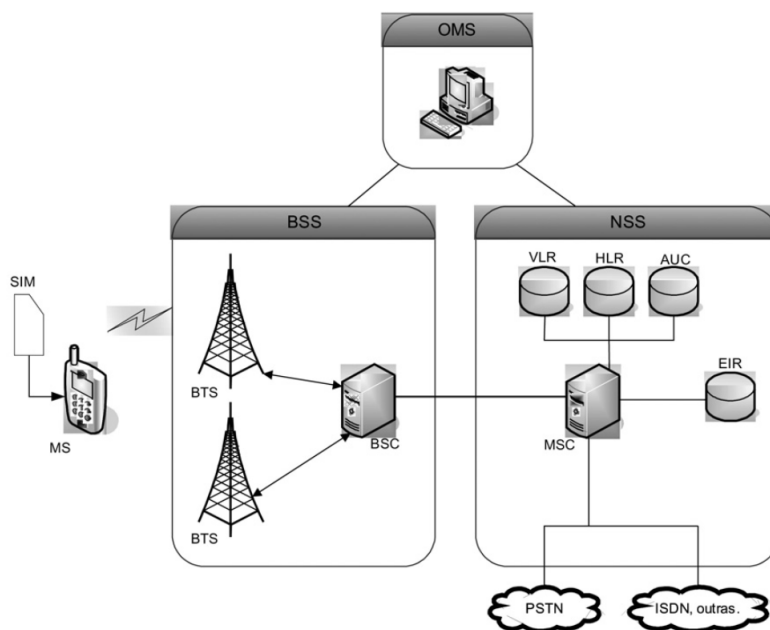
3.4 COMUNICAÇÃO MÓVEL

3.4.1 GSM

A tecnologia GSM (*Global System for Mobile Communications*) é um padrão de comunicação sem fio que foi desenvolvido para telefonia móvel. Foi introduzida pela primeira vez em 1991 na Europa e se tornou uma das tecnologias de comunicação móvel mais adotada em todo o mundo. O GSM é um sistema digital de segunda geração (2G) que substituiu os sistemas analógicos anteriores, como o AMPS (Advanced Mobile Phone System) e o NMT (Nordic Mobile Telephone). Foi um dos primeiros sistemas de telefonia móvel a adotar a tecnologia digital, o que melhorou significativamente a qualidade das chamadas e permitiu a transmissão de dados, como mensagens de texto (SMS). O fato de ter uma compatibilidade global, ou seja, a facilidade de usar dispositivos GSM por todo o mundo (se configurados corretamente) facilitou a disseminação e popularização da tecnologia. A Figura 6 mostra um exemplo da arquitetura de uma rede GSM, onde esta pode ser dividida em três subsistemas: BSS (*Base Station Subsystem*), cuja função é o recebimento de dados; NSS (*Network and Switching Subsystem*), responsável pelo gerenciamento e comutação da rede; e OMS (*Operation and Maintenance System*), responsável pelo suporte e operação da rede (PIROTTI; ZUCCOLOTTO, 2019).

O GSM utiliza a técnica de modulação conhecida como *??* (*Gaussian Minimum Shift Keying*) para permitir que vários usuários compartilhem a mesma frequência de rádio. Usa um esquema de criptografia para garantir a segurança das chamadas de voz e dados. Ele opera em várias frequências, incluindo 900 MHz e 1800 MHz, e suporta serviços

Figura 6 – Exemplo de Arquitetura da Rede GSM.



Fonte: (PIROTTI; ZUCCOLOTTO, 2019)

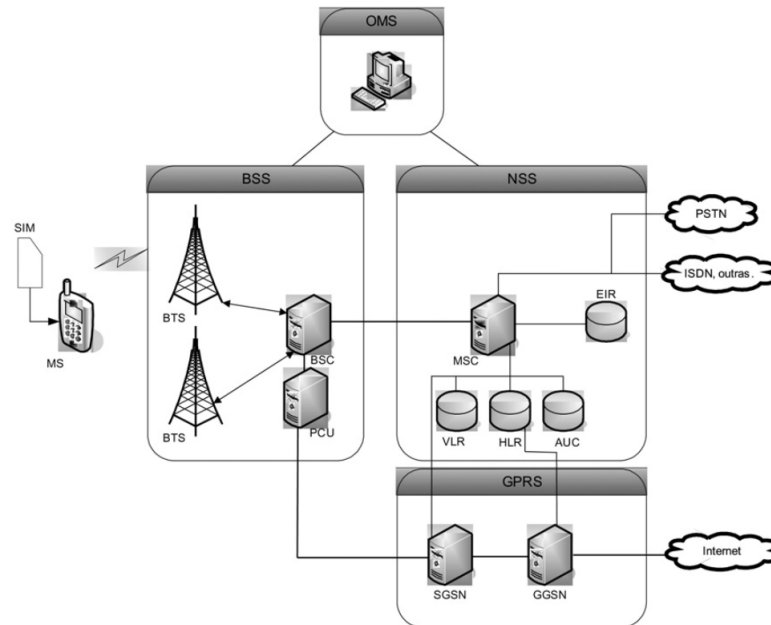
de voz e dados, como SMS, MMS e navegação na web. O GSM também é compatível com roaming internacional, permitindo que os usuários usem seus telefones em outros países e foi o primeiro sistema de comunicação móvel a se tornar amplamente adotado em todo o mundo.

3.4.2 GPRS

O GPRS é uma tecnologia de rede de dados móveis que permite que os usuários se conectem à Internet e a outros serviços de rede de dados móveis usando seus telefones celulares e outros dispositivos móveis. Ele usa uma técnica de comutação de pacotes para transmitir dados em vez de uma técnica de comutação de circuitos usada para chamadas de voz. O GPRS é capaz de fornecer velocidades de dados mais rápidas do que o GSM, permitindo que os usuários acessem a Internet, enviem e-mails e usem aplicativos de rede de dados móveis em seus dispositivos móveis. O GPRS é considerado uma tecnologia de rede de segunda geração (2.5G) e foi introduzido como parte da evolução do GSM para redes de 2G para redes de terceira geração (3G) e é capaz de fornecer velocidades de dados de até 114 kbps, o que é significativamente mais rápido do que as velocidades de dados do GSM (PIROTTI; ZUCCOLOTTO, 2019). A Figura 7 mostra um exemplo de arquitetura para uma rede GPRS, que assim como visto na Figura 6, utiliza-se dos mesmos subsistemas da rede GSM com a adição de certos elementos, como unidades de controle de pacote, servidor do nó de suporte (SGSN) e *gateway* para o nó de suporte (GGSN),

peritando o aumento de dados transmitidos.

Figura 7 – Exemplo de Arquitetura da Rede GPRS.



Fonte: (PIROTTI; ZUCCOLOTTO, 2019)

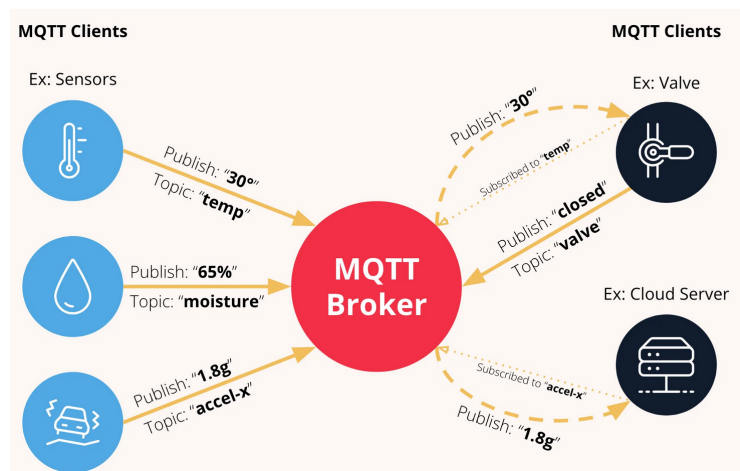
Assim como o GSM, também é amplamente utilizado em todo o mundo e é uma tecnologia importante para a conectividade móvel.

3.5 PROTOCOLO MQTT

O Protocolo MQTT (Message Queuing Telemetry Transport) é um protocolo de comunicação leve e eficiente projetado para trocar mensagens entre dispositivos em redes com largura de banda limitada e alta latência, como a Internet das Coisas (IoT). Ele foi desenvolvido por Andy Stanford-Clark da IBM e Arlen Nipper da Arcom (anteriormente Eurotech) em 1999 (HIVEMQ, 2021). A Figura 8 mostra o princípio básico de comunicação do protocolo MQTT, onde um dispositivo (*client*) irá enviar para o broker um determinado valor (onde este ato é chamado de *publish*, e um segundo dispositivo irá fazer a requisição deste valor por meio do broker, onde esta atividade é chamada de *subscribe*).

Segundo Oliveira (OLIVEIRA, 2017), o protocolo MQTT é uma solução atraente para a Internet das Coisas (IoT) devido à sua abordagem inovadora de comunicação e baixa utilização de recursos. Esse protocolo utiliza o conceito de um broker, que atua como um software servidor, recebendo e repassando solicitações quando necessário. O broker funciona como um servidor bidirecional, tanto recebendo quanto enviando dados, eliminando assim a necessidade de manter um servidor dedicado na rede, uma característica fundamental para dispositivos de IoT com recursos limitados.

Figura 8 – Ideia de funcionamento do MQTT.



Fonte: (TWILIO, 2023)

Comparando o broker MQTT a um servidor web ou um Sistema de Gerenciamento de Banco de Dados (SGBD), percebe-se que o broker tem uma abordagem mais simplificada e é orientado a aplicativos que lidam com a transferência de dados simples. Isso resulta em uma sintaxe mais simples e uma interface mais leve, tornando-o ideal para ambientes com restrições de recursos. A maioria dos sistemas IoT, incluindo dispositivos populares como o Arduino e o nodeMCU, oferecem suporte ao protocolo MQTT.

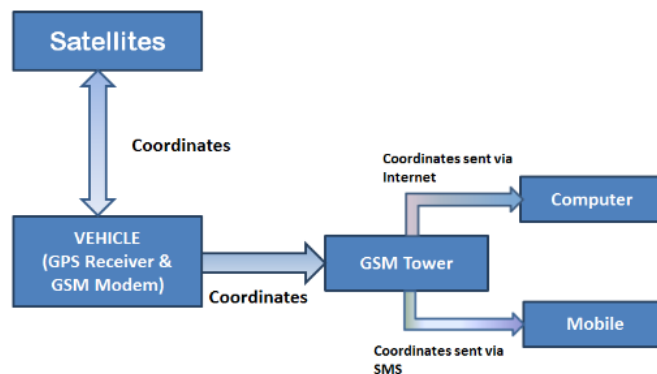
3.6 RASTREAMENTO VEICULAR

Sistemas de rastreamento veiculares são soluções tecnológicas que permitem a localização e monitoramento de veículos em tempo real. A Figura 9 mostra um exemplo de como um sistema de rastreamento veicular é estruturado, onde o veículo captura a sua posição a partir de um módulo GPS e um dispositivo GSM é utilizado para o envio dos dados de posicionamento por meio da rede, chegando ao destino final, podendo ser acessado por um dispositivo de sua preferência.

Estes sistemas utilizam tecnologias como GPS e comunicação móvel (GSM, 3G, 4G) para coletar informações sobre a localização, velocidade e direção do veículo, entre outros. Os dados coletados são enviados para um servidor central onde, em grande parte das aplicações os dados são transformados por meio de um aplicativo ou interface para a utilização do cliente final.

Tais sistemas são amplamente utilizados em diversas áreas, como transporte de cargas, serviços de emergência, segurança pública, entre outras. Além disso, os sistemas de rastreamento veiculares também podem ser utilizados para melhorar a segurança dos veículos. Eles permitem que os proprietários monitorem a localização de seus veículos em tempo real e recebam alertas em caso de roubo ou uso não autorizado. Isso pode ajudar a

Figura 9 – Funcionamento de um Sistema de Rastreamento Veicular.



Fonte: (COSTA; AHMED; RAHMAN, 2015)

recuperar veículos roubados e reduzir o risco de perda financeira. Uma importante aplicação dos sistemas de rastreamento veiculares é na otimização da logística de transporte. Eles permitem que as empresas monitorem a localização de seus veículos em tempo real e planejem rotas mais eficientes, o que pode ajudar a reduzir o tempo de entrega e os custos de transporte.

Ainda no âmbito empresarial, os sistemas de rastreamento veicular permitem que as empresas monitorem suas frotas de veículos em tempo real, o que ajuda a melhorar a eficiência operacional, reduzir custos evitando multas, otimizando rotas e consumo de combustível e aumentar a segurança dos motoristas e dos veículos, tanto da frota quanto quem circula nas estradas.

4 PROPOSTA E IMPLEMENTAÇÃO DE UM SISTEMA DE RASTREAMENTO VEÍCULAR

Este capítulo visa mostrar, de forma detalhada, todas as etapas para a implementação da solução proposta. A primeira parte deste capítulo terá como foco as especificações técnicas dos componentes utilizados e a prototipagem destes componentes para formar o hardware do projeto. Em seguida, foca-se no software desenvolvido que atuará em conjunto com o hardware para obter os resultados esperados. Por fim, será abordada a forma de comunicação entre o dispositivo e o *webservice* utilizado para validar os resultados obtidos.

4.1 REQUISITOS E ARQUITETURA PROPOSTA

Para iniciar o desenvolvimento do projeto, é necessário definir os elementos que serão utilizados para esta e o escopo do objetivo no qual se pretende alcançar. De acordo com a ideia inicial, existem alguns requisitos que precisam ser definidos para atingir o resultado esperado.

A Figura 10 representa de forma ilustrativa os requisitos do projeto, citados abaixo.

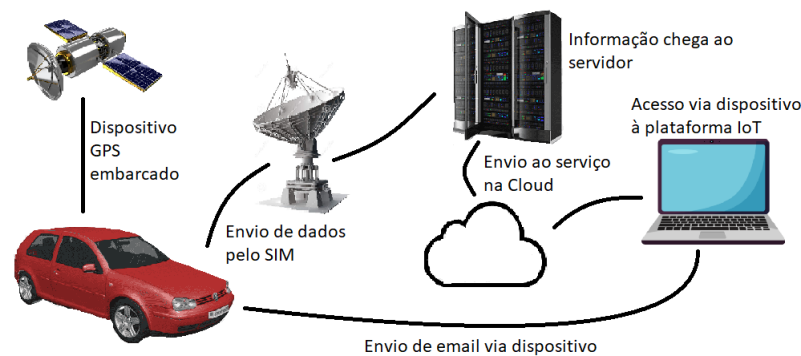
- O veículo deverá possuir um módulo embarcado que terá como objetivo obter os dados de localização geográfica, bem como de velocidade, a partir da tecnologia GPS.
- Os dados obtidos pelo módulo são enviados em tempo real para um serviço na internet, onde o envio é feito através da rede GPRS acessada por meio de um cartão SIM de uma empresa provedora de telefonia móvel.
- O serviço disponível na internet recebe os dados e armazena em uma base de dados, onde pode ser acessada por meio de um painel, bem como a validação dos dados por meio de mapas.
- O dispositivo também envia, em tempo real, alertas em caso de velocidade excessiva ao longo de um determinado período.

4.2 HARDWARE

Para a implementação do projeto, a plataforma de desenvolvimento escolhida foi a Arduino, citada na Seção 3.2.1. Além desta, é necessário alguns outros componentes, como um módulo GPS para a obtenção da localização geográfica e um módulo GSM/GPRS para a comunicação entre o dispositivo e a rede de dados. Com isto, foram selecionados os seguintes hardwares, de modo a atender da melhor forma possível os requisitos do projeto:

- Placa de desenvolvimento Arduino Mega 2560: A escolha pela versão Arduino Mega entre as diversas opções se deu pelo fato deste conter um custo relativamente baixo e suas especificações suprirem com certa folga as necessidades do projeto. Além disso, a grande gama de projetos acadêmicos que utilizam

Figura 10 – Sistema para rastreamento veicular.



Fonte: Autor.

deste modelo o tornam de fácil compreensão devido ao fato de existir diversos exemplos dos mais variados tipos. Outro fator é a disponibilidade de fóruns especializados na plataforma Arduino, facilitando a consulta de soluções em caso de problemas durante a prototipagem.

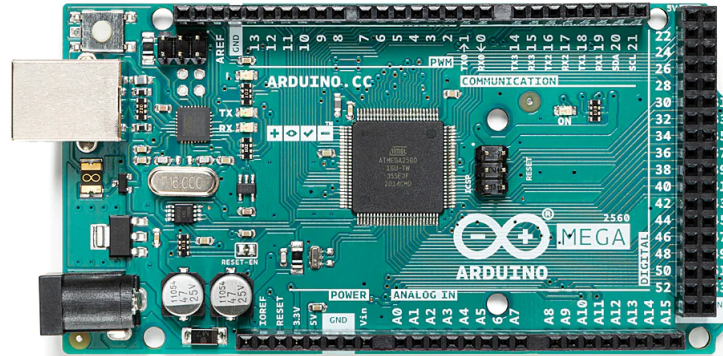
- **Shield SIM900:** A transmissão de dados da placa de controle para a internet será feita por meio do shield SIM900. Este é um módulo GSM que utiliza-se de um cartão SIM com acesso à rede GSM para comunicar dados entre dispositivos. A escolha se deu pelo fato da fácil aplicação junto à placa de controle e a disponibilidade de bibliotecas próprias para auxílio na implementação do código.
- **Módulo GPS Neo-6M:** Para obtenção dos dados de localização geográfica, optou-se pela utilização do módulo Neo-6M. Este conta com uma precisão de localização de até 2,5m e, aliado ao seu baixo custo, simplicidade de integração à placa de desenvolvimento e uma vasta variedade de bibliotecas para a aplicação do mesmo no software, o tornaram a escolha preferencial para este processo.

4.2.1 Arduino Mega 2560

O Arduino Mega 2560 é uma placa de microcontrolador que utiliza o chip ATmega2560. A placa possui um total de 54 pinos digitais de entrada e saída (I/O), dos quais 15 podem ser usados como saídas PWM, que permitem controle de potência ou intensidade em dispositivos. Além disso, há 16 entradas analógicas, que permitem a leitura de sinais variáveis. É possível conectar-se ao dispositivo utilizando-se de um cabo USB tipo A/B, permitindo assim a gravação de código na memória flash para a execução do mesmo (ARDUINO..., 2023). Esta conexão também funciona como alimentação de energia para o dispositivo, onde para isto também existe um conector P4. A Figura 11 mostra uma imagem do Arduino Mega, onde é possível notar os diversos componentes

citados anteriormente.

Figura 11 – Modelo Arduino Mega 2560.



Fonte: (ARDUINO..., 2023)

A seguir serão apresentados de forma mais detalhada alguns detalhes técnicos relacionados à placa de desenvolvimento Arduino Mega.

- **Microcontrolador:** ATmega2560.
- **Tensão de operação:** 5V.
- **Tensão de entrada (recomendada):** 7-12V.
- **Tensão de entrada (limites):** 6-20V.
- **Pinos digitais de I/O:** 54 (dos quais 15 podem ser usados como saídas PWM).
- **Pinos de entrada analógica:** 16.
- **Corrente DC por pino I/O:** 20 mA.
- **Corrente DC para o pino 3.3V:** 50 mA.
- **Memória Flash:** 256 KB (8 KB são usados pelo bootloader).
- **SRAM:** 8 KB.
- **EEPROM:** 4 KB.
- **Clock Speed:** 16 MHz.

4.2.1.1 Alimentação

O Arduino Mega 2560 apresenta mais de uma alternativa para a sua energização. As mais comuns são feitas através de cabo USB A/B e por meio de fontes de alimentação externa. A alimentação via cabo USB é iniciada assim que o cabo é conectado entre a placa e um dispositivo USB. Pelo fato do circuito proveniente do USB fornecer a proteção da porta em caso de problemas relacionados à tensão, não é necessário o uso de componentes

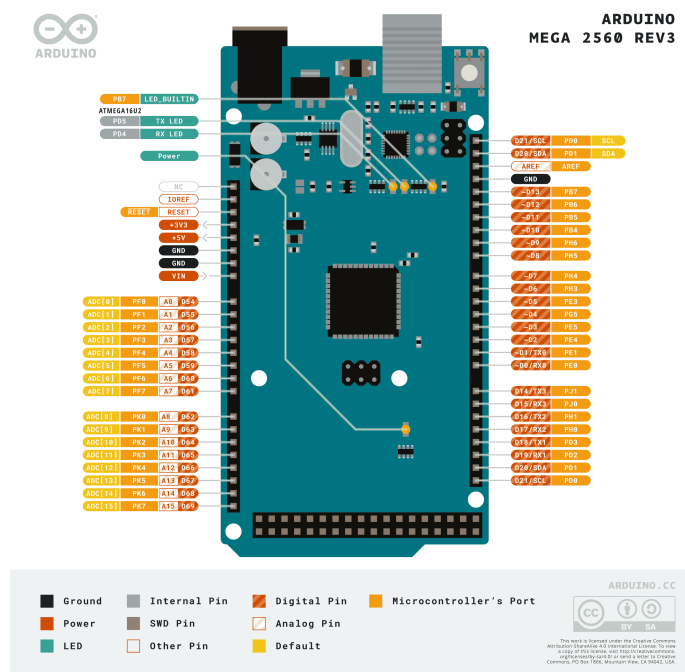
para estabilizar a alimentação. Para a alimentação vinda de uma fonte externa, esta é feita através de um conector do tipo P4, de 2.1mm. Recomenda-se que o valor de tensão fornecido por esta fonte esteja entre 7-12V, porém a placa apresenta limites entre 6-20V. Nos casos fora do intervalo recomendado, pode ocorrer problemas de funcionamento ou danos na placa.

O Arduino Mega também pode ser utilizado como uma fonte de energia para outros dispositivos que serão utilizados em conjunto com o mesmo. Este possui saídas de 3,3V que geram correntes máximas de até 50mA, saídas de 5V gerando correntes máximas de 20mA, bem como pinos de referência (GND) e pinos de alimentação ligado diretamente à fonte externa, gerando uma tensão igual a da alimentação (VIN).

4.2.1.2 Pinout do Arduino Mega

A placa Arduino Mega contém em sua estrutura pinos de entrada e saída digitais e pinos de entrada analógica. A partir destes é possível receber e enviar dados de e para outros componentes. A Figura 12 mostra o esquemático do Arduino Mega 2560, identificando cada pino na placa e seu respectivo funcionamento.

Figura 12 – Pinagem Arduino MEGA.



Fonte: (PINOUT..., 2023)

Pode-se observar na Figura 12 que o Arduino Mega possui mais de 50 pinos digitais, que podem ser usados tanto como entrada ou saída de sinal. Estes operam com uma tensão de 5V, podendo alternar entre dois níveis lógicos: Alto, onde a entrada/saída possui uma tensão de 5V, e Baixo, onde a entrada/saída contém uma tensão no valor de 0V.

A manipulação e definição de uso destes pinos é feita através de funções programadas através de software. Dentre as principais, podem ser citadas:

- `pinMode()`: Define se um determinado pino será utilizado como um sinal de entrada ou de saída;
- `digitalWrite()`: Usada para definir o estado lógico de um pino digital;
- `digitalRead()`: Usada para identificar o estado lógico de um pino digital, onde o estado é proveniente de um sinal externo.

Certos pinos digitais possuem mais de uma função, dentre estas podem ser citadas:

- PWM: O PWM (*Pulse Width Modulation*) é uma técnica utilizada para emular um sinal analógico usando um dispositivo digital, onde é amplamente usada em eletrônica para controlar a potência entregue a dispositivos. No Arduino Mega, os pinos 2 até 13 e 44 até 46 podem ser utilizados com a função de PWM, utilizando a função `analogWrite()` no software.
- Serial: A comunicação Serial se dá por meio da lógica de transmissor/receptor e é a forma mais básica de comunicação entre dispositivos. No Arduino Mega existem quatro portas diferentes, denominadas pelos pares de pinos 0 e 1, 19 e 18, 17 e 16, 15 e 14 (RX e TX, respectivamente).

Além dos pinos digitais, o Arduino Mega também possui 16 portas de entrada analógica. Para a manipulação destas, existem duas funções:

- `analogReference()`: A partir desta é possível configurar a tensão de referência para as entradas analógicas.
- `analogRead()`: A partir desta é possível obter o valor presente na entrada analógica selecionada.

4.2.1.3 Programação

A programação do Arduino Mega para este projeto foi realizada através da Arduino IDE (veja Seção 3.2.2) utilizando as linguagens C/C++. Por padrão, a plataforma Arduino contém um programa inicializador (*bootloader*) para possibilitar o carregamento de novos códigos sem a necessidade de hardware para programação externa, sendo este conhecido como AVRDUDE. AVRDUDE é uma ferramenta de linha de comando usada para programar microcontroladores como os usados em placas Arduino, e permite a gravação de firmware compilado em microcontroladores. Dentre suas funcionalidades, inclui gravação de arquivos binários ou hexadecimais na memória do microcontrolador, leitura do conteúdo da memória do microcontrolador, verificação da integridade comparando o conteúdo da memória com um arquivo e exclusão da memória do microcontrolador antes de programá-lo novamente.

4.2.2 Shield GPRS SIM900

O shield SIM900 é um módulo de comunicação GSM/GPRS que permite ao Arduino se conectar a redes móveis, facilitando a realização de chamadas, envio e recebimento de SMS, e a conexão à Internet via GPRS a partir do uso de um cartão SIM. É amplamente utilizado em projetos de Internet das Coisas (IoT) e automação que requerem conectividade móvel. Em modo *standby*, possui um baixo consumo de energia. Além disso, por possuir uma banda de frequência ampla, este shield possibilita o seu uso em quase todo o mundo (EFCOM, 2013). A Figura 13 mostra o módulo já acoplado a placa do Arduino Mega 2560.

Figura 13 – Shield SIM900.



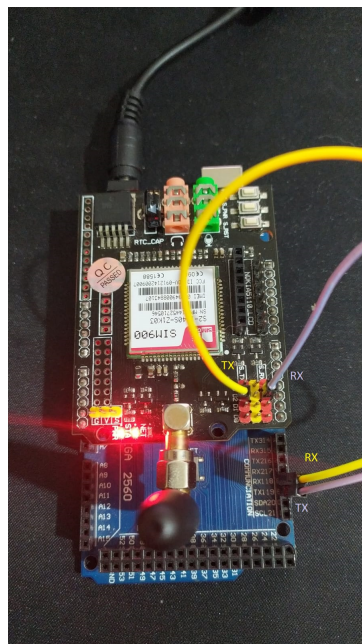
Fonte: Autor.

Já a Figura 14 mostra a conexão entre o shield e as portas seriais do Arduino utilizadas para a comunicação entre os dois dispositivos.

Por se tratar de um shield, a sua conexão com o Arduino se dá por meio do encaixe deste sobre o Arduino, onde este encaixe permite a alimentação direta do mesmo. A comunicação entre eles se dá por meio da comunicação serial. Para isto, utilizou-se das portas de comunicação serial existentes no Arduino Mega, dadas pela porta serial 1, proveniente dos pinos 19 e 18 (RX e TX).

Para tarefas que exigem maior consumo de energia, recomenda-se utilizar uma alimentação externa para alimentar o Arduino, de modo a manter a corrente do shield estável durante sua utilização. Para isto, utilizou-se uma fonte de 9V e 3A, visto que esta se encontra dentro da faixa recomendada para a alimentação da placa, e garante uma corrente suficiente para atividades do shield que exigem um consumo maior de energia. A Figura 15 mostra o modelo de fonte utilizado para a alimentação do projeto em bancada. Para a alimentação no automóvel, utiliza-se um cabo USB com jack P4 de 2.1mm. A porta

Figura 14 – Imagem da conexão serial entre SIM900 e Arduino.



Fonte: Autor.

USB é conectada em uma fonte de energia para automóveis, com saída de 5V e 2.1A. A Figura 16 mostra a fonte e o cabo utilizado para a conexão embarcada.

Como citado anteriormente na Seção 4.1, o módulo GPRS necessita de um cartão SIM para a conexão com a internet. Para isso, acopla-se na parte traseira do shield o cartão SIM, onde a Figura 17 ilustra o cartão utilizado e o local onde deve se inserir o mesmo.

Após a inserção do cartão SIM, o shield está totalmente pronto para enviar e receber dados via GSM/GPRS.

4.2.3 Módulo GPS Neo-6M

O módulo GPS NEO-6M é um dispositivo popular utilizado em projetos para fornecer dados de posicionamento global. É conhecido por sua alta sensibilidade, o que permite captar sinais GPS de satélites mesmo em condições adversas, como áreas urbanas densas, e pode fornecer atualizações de posição até 5 vezes por segundo (5Hz), o que é adequado para muitas aplicações de navegação e rastreamento em tempo real. O fato de possuir um baixo consumo de energia contribui para a escolha de sua utilização neste projeto (U-BLOX, 2011). Para interconexão com a placa Arduino Mega 2560, apenas quatro pinos são necessários.

- VCC: Pino no qual será ligada a tensão de entrada. Pode variar entre 3,3V e 5V.

Figura 15 – Fonte 9V 3A.



Fonte: Autor.

- GND: Pino de referência, ligado junto a placa que irá se conectar.
- RX e TX: Pinos de comunicação serial, utilizados para a transferência de dados entre o módulo e a placa de controle.

Para a comunicação entre o módulo e a placa Arduino, foram utilizadas portas seriais e a alimentação do módulo foi feita a partir do módulo SIM900 citado na Seção 4.2.2, com uma tensão de 3,3V. A Figura 18 mostra a conexão serial entre a placa Arduino e o módulo, bem como a alimentação de energia proveniente do shield SIM900.

Após as conexões, o módulo está pronto para uso. Para a comunicação entre o módulo e a placa de controle, utiliza-se de uma biblioteca dedicada para facilitar a troca de dados.

4.3 FIRMWARE

Para a aplicação do projeto e a comunicação entre os dispositivos apresentados, é necessária a aplicação de um *Firmware*. O *Firmware* é o binário gerado a partir do código fonte da aplicação, e responsável por inicializar e gerenciar os componentes.

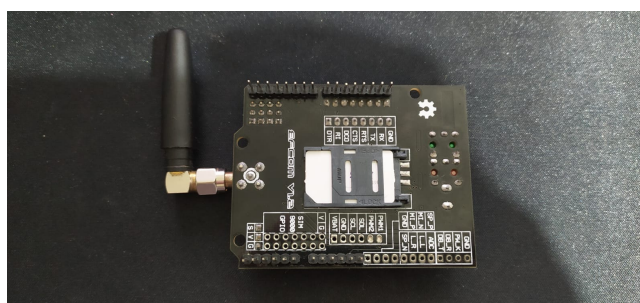
O desenvolvimento do *firmware* presente neste projeto foi feito a partir da linguagem C++ utilizando a plataforma Arduino IDE, tanto para a programação, quanto para compilação e gravação do código na placa de controle. As principais funções do *firmware* implementado são:

Figura 16 – Fonte para uso embarcado.



Fonte: Autor.

Figura 17 – Shield SIM900 com cartão SIM.

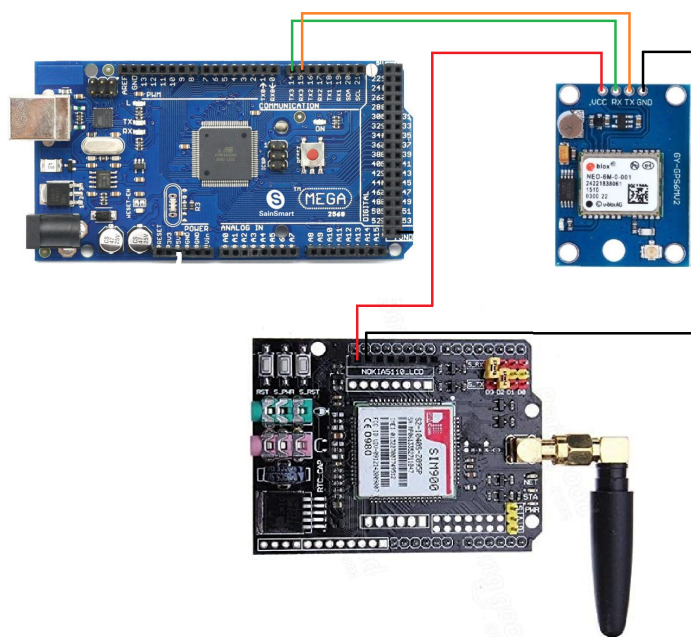


Fonte: Autor.

- Comunicação entre os componentes (Arduino, Shield SIM900 e módulo GPS Neo-6M);
- Obtenção de dados a partir do módulo GPS;
- Envio de dados para o *webservice* a partir da conexão GPRS proveniente do shield SIM900.

Como padrão, a estrutura padrão de código para programas que serão implementados em controladores Arduino seguem duas funções principais, onde as Figuras 19 e 20 mostram a aplicação destas no código do projeto.

Figura 18 – Conexão do Módulo Neo-6M.



Fonte: Autor.

- `void setup()`: Esta função é executada ao iniciar o microcontrolador e é responsável pela sua configuração, como declaração de variáveis, configurações de pinos, entre outros.
- `void loop()`: Função executada diversas vezes enquanto o microcontrolador está ligado, como um laço de repetição. Utilizado para realizar as atividades necessárias. As variáveis `vel` e `tempo` podem ser modificadas de acordo com a necessidade do usuário.

O *Firmware* contém outras funções, criadas para atividades específicas do projeto. As Figuras 21 e 22 mostram a aplicação destas funções abaixo.

- `void displayInfo()`: Utilizada para validação de funcionamento do GPS, onde a mesma retorna valores obtidos via GPS para o monitor serial.
- `void email()`: Utilizada para enviar um e-mail em caso de velocidade excessiva durante um determinado período de tempo. Sua utilização na função principal está condicionada a uma lógica *if*.

Para a execução do *firmware*, inicialmente inicia-se declarando as bibliotecas necessárias para facilitar a utilização dos componentes presentes no projeto. A seguir, na função `setup()` são declaradas as conexões entre o shield SIM900 e o Arduino por meio das portas seriais. Neste também são definidos os dados que serão enviados para o *webservice* a partir do formato PSON (Figura 20).

Figura 19 – Função loop().

```
void loop() {  
    while(gpsSerial.available() > 0){  
        if (gps.encode(gpsSerial.read()))  
            displayInfo();  
    }  
    //Lógica para enviar email em caso de velocidade excessiva  
    if (gps.speed.kmph() > vel){  
        count = count + 1;  
        if (count > tempo){  
            email();  
            Serial.println("Velocidade acima");  
        }  
    } else {  
        Serial.println("Velocidade abaixo");  
        //Serial.println(count);  
    }  
    //updateSerial();  
    thing.handle();  
    delay(1000);  
}
```

Fonte: Autor.

Figura 20 – Função setup().

```
void setup() {  
    Serial.begin(115200);  
    gpsSerial.begin(9600);  
    delay(1000);  
  
    ThingyTinyGSM thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL, gpsSerial);  
  
    thing["Position"] >> [] (pson & out){  
        out["Latitude"] = gps.location.lat();  
        out["Longitude"] = gps.location.lng();  
        out["Velocidade"] = gps.speed.kmph();  
        out["Altitude"] = gps.altitude.meters();  
    };  
}
```

Fonte: Autor.

Para a função loop(), que irá se repetir intermitentemente enquanto o microcontrolador estiver energizado, a mesma só irá iniciar no momento que a função setup() tenha sido encerrada. A função loop() irá validar a conexão do módulo GPS com os satélites, e em caso de confirmação, irá mostrar os dados de localização geográfica no monitor serial. Na sequência, com os dados de geolocalização obtidos e ainda dentro do laço de repetição, o *firmware* irá entrar em um laço condicional para verificar a velocidade obtida na última leitura do GPS, como visto na Figura 19. Caso esta seja maior que a variável "vel", onde esta é parametrizável e tem um valor definido pelo usuário (para testes este valor foi considerado 50 km/h) e uma variável chamada *count* cujo valor inicial é 0 irá receber o

Figura 21 – Função displayInfo().

```

void displayInfo(){
    Serial.print("Location: ");
    if (gps.location.isValid()) {
        Serial.print("Lat: ");
        Serial.println(gps.location.lat(), 6);
        Serial.print(("Long:"));
        Serial.println(gps.location.lng(), 6);
        Serial.print(("Velocidade:"));
        Serial.println(gps.speed.kmph());
        Serial.print(("Alt:"));
        Serial.println(gps.altitude.meters());
        Serial.println();
    } else {
        Serial.print ("Conexão falhou");
    }
}

```

Fonte: Autor.

Figura 22 – Função email().

```

SMTPSession smtp;
void email(){
    smtp.debug(1);
    ESP_Mail_Session session;

    /* Dados de email */
    session.server.host_name = SMTP_HOST;
    session.server.port = SMTP_PORT;
    session.login.email = AUTHOR_EMAIL;
    session.login.password = AUTHOR_PASSWORD;
    session.login.user_domain = "";

    SMTP_Message message;
    message.sender.name = "Carro #1";
    message.sender.email = AUTHOR_EMAIL;
    message.subject = "EXCESSO DE VELOCIDADE";
    message.addRecipient("", "arthur.rodgers@gmail.com");

    String textMsg = "A velocidade máxima permitida pelas diretrizes da empresa foi excedida por mais de 10 segundos!";
    message.text.content = textMsg.c_str();
    message.text.charset = "us-ascii";
    message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

    message.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_low;
    message.response.notify = esp_mail_smtp_notify_success | esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
    if (!smtp.connect(ssession))
        return;
    if (!MailClient.sendMail(smtp, smessage))
        Serial.println("Error sending Email, " + smtp.errorReason());
    count = 0;
}

```

Fonte: Autor.

valor dela mesma somado em 1. Então, para cada leitura do GPS na qual a velocidade esteja acima de 50 km/h, é acrescentado 1 ao valor da variável *count*. A leitura do GPS é feita a cada 1 segundo. Caso o valor da variável *count* chegue a 10, a função *email* é acionada, enviando um email e avisando que o veículo está acima da velocidade permitida por demasiado tempo.

Além disso, o laço de repetição conta com uma função proveniente da biblioteca do *webservice* utilizado, que envia os dados previamente configurados na função *setup* para o

webservice em rede.

A comunicação com o *webservice* é feita através do envio dos dados no padrão de estrutura PSON. O *webservice* recebe uma mensagem contendo 1 recurso chamado "Posição" que contém 4 campos, sendo estes:

- Latitude: Dados de latitude do GPS;
- Longitude: Dados de longitude do GPS.
- Altitude: Dados de altitude do GPS;
- Velocidade: Dados de velocidade do GPS, medidos em km/h.

Estes dados são armazenados em um *bucket* onde este é uma espécie de banco de dados da plataforma *webservice* utilizada, que será melhor explicada na Seção 4.4.

4.4 THINGER.IO

Para este projeto, o serviço de *webservice* selecionado foi o Thinger.io. A escolha desta plataforma se deve pela facilidade de implementação junto aos componentes utilizados, além das diversas ferramentas presentes na versão gratuita, como painéis, *buckets* de dados, acesso a *endpoints*, entre outros.

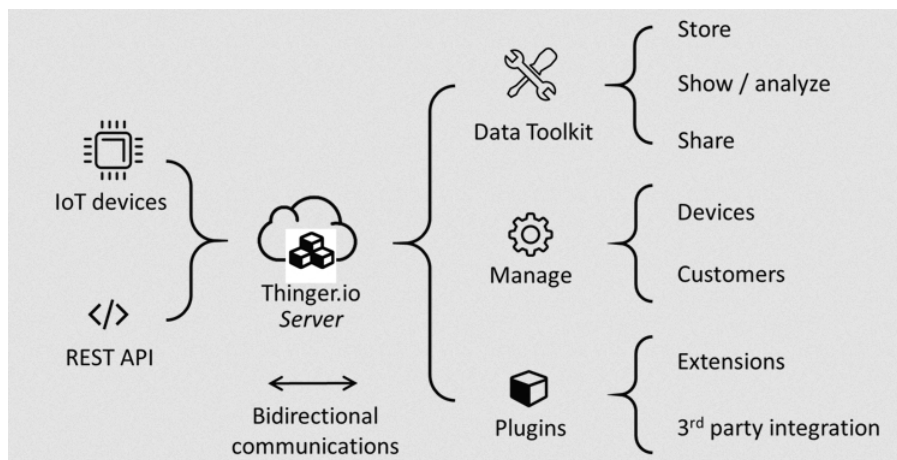
O Thinger.io é uma plataforma de Internet das Coisas (IoT) que facilita a conexão, o gerenciamento e a interação com dispositivos conectados à internet. A plataforma permite conectar e gerenciar dispositivos IoT de maneira eficiente, oferecendo uma interface intuitiva para visualizar o status dos dispositivos e suas métricas em tempo real. Um dos principais recursos do Thinger.io é sua API RESTful, que facilita a integração com outros serviços e aplicações, permitindo a leitura e a escrita de dados nos dispositivos conectados (THINGER.IO, 2024).

A plataforma pode ser utilizada de forma gratuita mediante algumas limitações, como o limite de envio de dados, onde existe um intervalo de envio de 1 minuto. A Figura 24 mostra de forma ilustrativa o funcionamento da plataforma, junto com as ferramentas presentes na mesma.

4.4.1 Configuração

Para a utilização do Thinger.io é necessário cadastrar-se na plataforma. Após cadastrado, o acesso para a utilização das suas ferramentas está liberado. O primeiro passo é cadastrar o dispositivo que irá se conectar à plataforma e o tipo do mesmo, sendo para este projeto selecionado um dispositivo MQTT. Após a criação, o dispositivo ficará disponível para acesso em uma lista de dispositivos, onde ao ser selecionado retorna uma página com diversos dados relacionados ao mesmo, como a quantidade de dados transmitidos tanto em tempo real quanto historicamente, *status* da conexão atual, tempo

Figura 23 – Diagrama de funcionamento da plataforma Thinger.io



Fonte: (THINGER.IO, 2024)

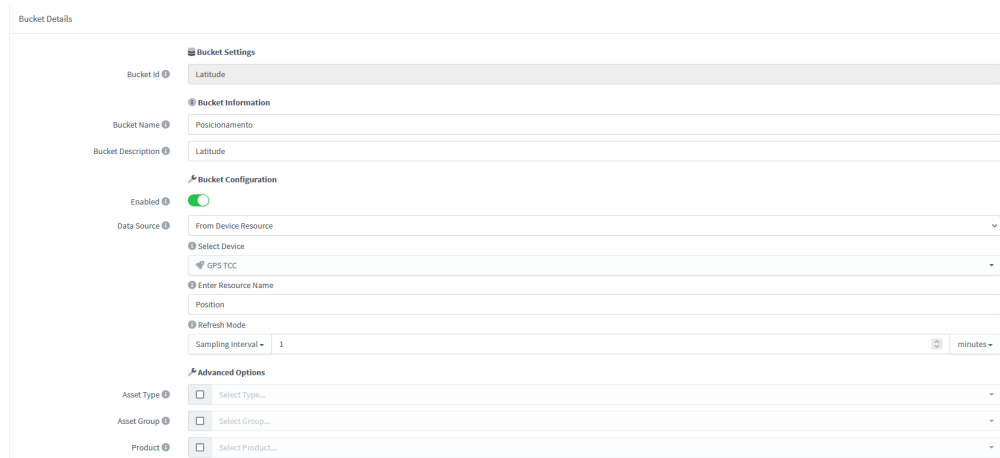
desde a última conexão, entre outros. A Figura 24 mostra a configuração feita para cadastrar o Arduino utilizado no projeto.

Figura 24 – Cadastro de Dispositivo na Plataforma Thinger.io

A imagem mostra a interface de usuário para o cadastro de um dispositivo na plataforma Thinger.io. O formulário é dividido em seções: 'Device Configuration' com campos para Device Type (MQTT Device), Device Id (Arduino Mega) e Device Credentials (ZR0dIhWN4nJKfIDP); 'Device Information' com campos para Device Name (GPS TCC) e Device Description (Optional device description); e 'Advanced Options' com campos para Asset Type, Asset Group e Product, todos com opções de seleção. O campo 'Enabled' está ativado com um interruptor verde. Um botão azul 'Add Device' com um ícone de checkmark está localizado na base do formulário.

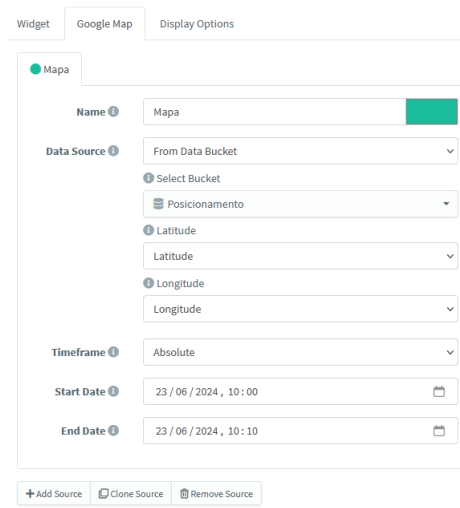
Fonte: Autor.

Após a criação do dispositivo, é feita a configuração do *bucket*. O *bucket* é um tipo de banco de dados que está atrelado ao dispositivo. Quando o dispositivo envia dados para o Thinger.io, o *bucket* será responsável por armazenar os dados e permitir a visualização dos mesmos caso acessado. É possível selecionar os dados que serão armazenados dentro deste *bucket*. A Figura 25 mostra a configuração para o *bucket*, bem como as variáveis que serão armazenadas no mesmo.

Figura 25 – Configuração da ferramenta *bucket* na Plataforma Thinger.io.

Fonte: Autor.

Com a informação presente no *bucket*, é possível usar *dashboards* para validar os dados provenientes do protótipo. Para isso, é possível selecionar um mapa que irá utilizar os dados salvos dentro do *bucket* e a partir destes marcar pontos utilizando-se da latitude e longitude enviadas. A Figura 26 mostra a configuração do mapa para validar se os dados recebidos estão condizentes com o esperado.

Figura 26 – Configuração da ferramenta *dashboard* na Plataforma Thinger.io.

Fonte: Autor.

O campo *Timeframe* é selecionado como absoluto, permitindo que o mapa criado gere o deslocamento feito pelo usuário baseando-se nos dados provenientes de latitude e longitude ao longo do intervalo selecionado.

Ao final desta seção, é possível observar que o projeto foi implementado com sucesso, atendendo todas os requerimentos impostos no início. Algumas dificuldades foram encontradas principalmente em relação à alimentação do sistema. Isso se deve principalmente pelo fato de que o *shield* SIM900 necessitar de uma corrente de aproximadamente 2A para inicializar corretamente. Nos primeiros testes, uma fonte com 1A foi utilizada, notando assim a necessidade de uma corrente maior. Além disso, em um primeiro momento para validação dos dados foi utilizado de um ESP32 com conexão Wi-fi, validando a interação entre o dispositivo GPS e a conexão com a plataforma Thinger.io.

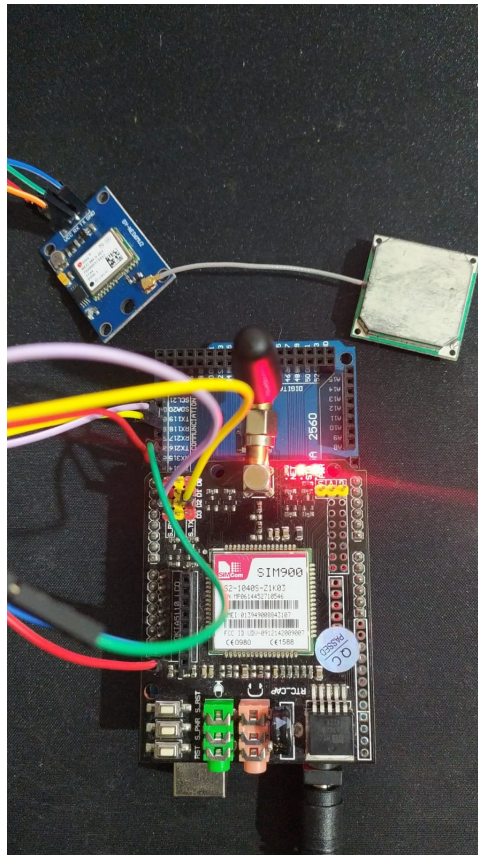
5 RESULTADOS

Este capítulo apresenta os resultados obtidos a partir da implementação completa do sistema, onde estão englobados os pontos de localização aplicados em um mapa a partir do Thingier.io e o envio de e-mails reportando sobre velocidades excessivas.

5.1 PREPARAÇÃO PARA TESTE PRÁTICO

A Figura 27 mostra o circuito completo montado em funcionamento durante testes primários, feitos de forma estacionária de modo a validar o funcionamento.

Figura 27 – Protótipo Final.



Fonte: Autor.

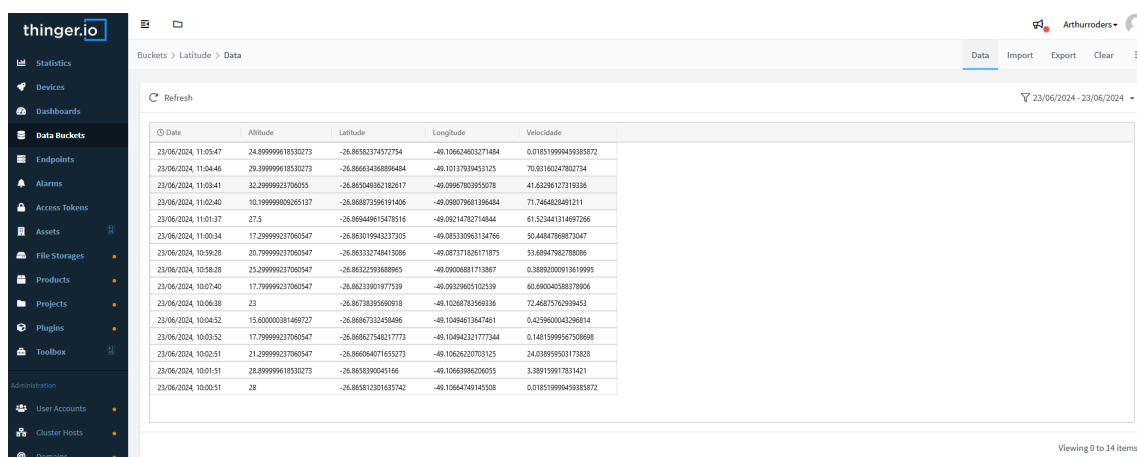
A partir da validação em bancada, a segunda etapa consistiu no embarque do protótipo no automóvel. Para isso, utilizou-se de um carregador automotivo com entrada USB e um cabo para alimentação do protótipo, conforme explicado na Seção 4.2.2. Com isto, foi possível acionar os dispositivos de modo a operarem enquanto a coleta e transmissão de dados ocorria.

5.2 COLETA DE DADOS

Esta seção trata da coleta de dados feita a partir do projeto construído. As rotas para a coleta de dados foram feitas entre a residência do autor e o shopping local. O primeiro trajeto iniciou-se na Rua Victor Bernards, em direção a Rua Max Humpl, onde esta se conecta à BR-470. O veículo segue na rodovia até chegar ao shopping local. Os limites de velocidade das vias são 40, 50 e 80km/h, respectivamente. A segunda rota passou pelas ruas Dr. Pedro Zimmermann (limite de 60km/h), rua Eng. Udo Deeke (limite de 60km/h), rua Ari Barroso (limite de 60km/h), onde com esta chegou-se a BR-470, fazendo a partir desta a rota inversa da primeira. O automóvel utilizado foi cedido pelo autor.

A partir dos dados enviados pelo Arduino Mega, o *bucket* criado para armazenar os dados foi populado. A Figura 28 mostra os dados obtidos via módulo GPS durante um percurso feito por um automóvel.

Figura 28 – Dados populados no *bucket*.



The screenshot shows the Thinger.io interface with a data bucket containing the following data:

Date	Altitude	Latitude	Longitude	Velocidade
23/06/2024, 11:05:47	24.899999618530273	-26.865833745372754	-49.106634603271484	0.01851999493835872
23/06/2024, 11:04:46	25.399999618530273	-26.866634368994804	-49.10137839453125	70.931602478027354
23/06/2024, 11:03:41	22.299999237060547	-26.865040362102617	-49.09957830955078	41.63296127319236
23/06/2024, 11:02:40	10.19999980255137	-26.868873596191406	-49.09807981305484	71.7464823491111
23/06/2024, 11:01:37	27.5	-26.868486154779316	-49.09214782714044	61.523441314897266
23/06/2024, 11:00:34	17.299999237060547	-26.863019943237305	-49.085330962134766	50.4484789873047
23/06/2024, 10:59:28	20.799999237060547	-26.86332748413086	-49.087371826171875	53.6894782788086
23/06/2024, 10:58:28	23.299999237060547	-26.8632239888965	-49.09006881713867	0.3889200913819995
23/06/2024, 10:57:40	17.799999237060547	-26.86233901977539	-49.092329605102539	60.60040588378906
23/06/2024, 10:56:38	23	-26.86738395690918	-49.10268783569336	72.46875762939453
23/06/2024, 10:54:52	15.60000381469727	-26.86867332458496	-49.10494513647461	0.42398004236814
23/06/2024, 10:53:52	17.799999237060547	-26.868627548217773	-49.10494323177344	0.14819999567508688
23/06/2024, 10:52:51	21.299999237060547	-26.86664071655273	-49.1026220703125	24.038959503173828
23/06/2024, 10:51:51	28.899999618530273	-26.86583390045166	-49.10663986206055	3.389159917831421
23/06/2024, 10:50:51	28	-26.865812301635742	-49.10664749145508	0.01851999493835872

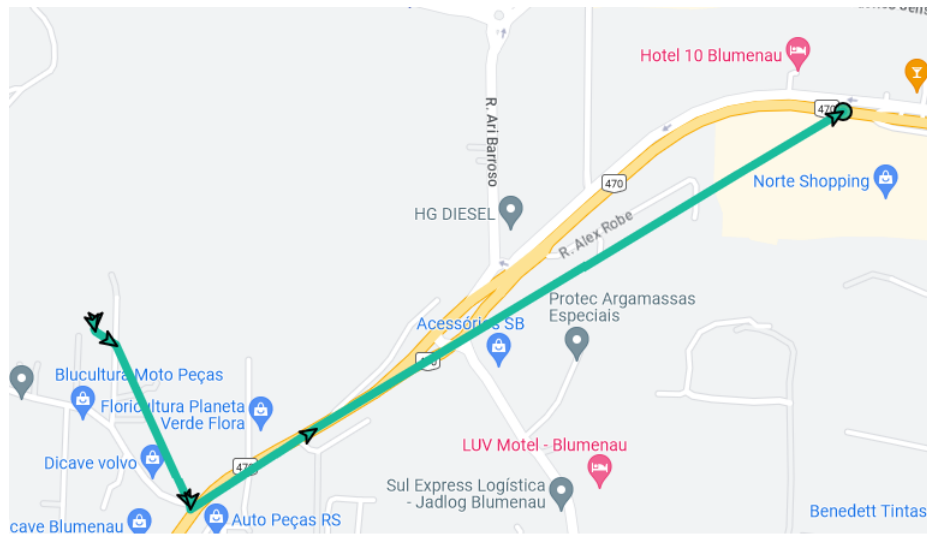
Fonte: Autor.

5.3 POSICIONAMENTO

As Figuras 29 e 30 mostram duas rotas diferentes geradas a partir dos dados armazenados no *bucket* gerado para armazenar os dados provenientes do GPS.

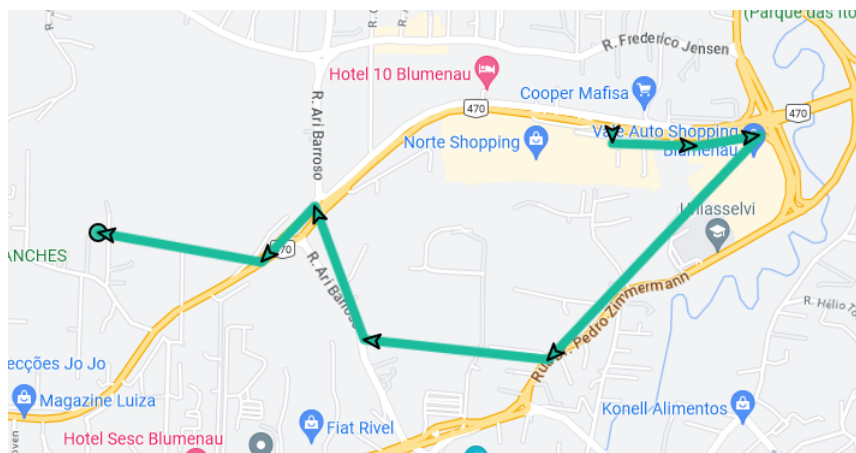
É possível observar que as rotas criadas pela ferramenta da plataforma não condiz com as rotas de ruas presentes no Google Maps. Isso se dá pelo fato da plataforma somente suportar amostras com intervalos de 1 minuto, fazendo com que os dados fiquem espaçados entre si e não tornando a rota 100% precisa, mas sim uma aproximação da rota real. A taxa atual é suficiente para um pedestre, porém para veículos acaba por não gerar um resultado tão satisfatório. Para sanar este problema bastaria uma coleta de dados com

Figura 29 – Rota criada a partir dos dados obtidos via GPS.



Fonte: Autor.

Figura 30 – Rota criada a partir dos dados obtidos via GPS.



Fonte: Autor.

uma frequência maior, visto que o intervalo entre amostras seria menor e assim, gerando uma rota mais precisa.

5.4 ALARME DE VELOCIDADE

Para os alarmes de excesso de velocidade, as Figuras 31 e 32 mostram os avisos enviados para o e-mail registrado na programação do *firmware* implementado no Arduino, neste caso o e-mail do próprio autor. Para os testes, considerou-se que, para casos onde a velocidade do veículo se mantivesse acima de 50 km/h durante 10 segundos, um e-mail

seria enviado.

Figura 31 – Lista de e-mails recebidos por excesso de velocidade.

<input type="checkbox"/>	☆	»	eu	EXCESSO DE VELOCIDADE - A velocidade máxima permitida pelas diretrizes da empresa foi excedida por mais de 10 segundos!	23 de jun.
<input type="checkbox"/>	☆	»	eu	EXCESSO DE VELOCIDADE - A velocidade máxima permitida pelas diretrizes da empresa foi excedida por mais de 10 segundos!	23 de jun.
<input type="checkbox"/>	☆	»	eu	EXCESSO DE VELOCIDADE - A velocidade máxima permitida pelas diretrizes da empresa foi excedida por mais de 10 segundos!	23 de jun.
<input type="checkbox"/>	☆	»	eu	EXCESSO DE VELOCIDADE - A velocidade máxima permitida pelas diretrizes da empresa foi excedida por mais de 10 segundos!	23 de jun.
<input type="checkbox"/>	☆	»	eu	EXCESSO DE VELOCIDADE - A velocidade máxima permitida pelas diretrizes da empresa foi excedida por mais de 10 segundos!	23 de jun.
<input type="checkbox"/>	☆	»	eu	EXCESSO DE VELOCIDADE - A velocidade máxima permitida pelas diretrizes da empresa foi excedida por mais de 10 segundos!	23 de jun.

Fonte: Autor.

Figura 32 – Conteúdo do e-mail recebido ao ultrapassar as condições de velocidade.



Fonte: Autor.

5.5 CUSTOS DO PROJETO

A Tabela 1 apresenta os valores aproximados dos componentes utilizados no desenvolvimento deste projeto.

Tabela 1 – Valores aproximados dos componentes utilizados no projeto.

Componente	Quantidade	Valor
Arduino Mega	1	R\$ 95,00
Shield SIM900	1	R\$ 120,00
Módulo GPS Neo-6M	1	R\$ 35,00
Fonte 9V	1	R\$ 20,00
Cartão SIM	1	R\$ 30,00
Cabos	10	R\$ 10,00

O custo final para a implementação será de aproximadamente R\$ 310,00.

6 CONCLUSÃO

O projeto apresentado trata do desenvolvimento de um sistema de rastreamento e acompanhamento de velocidade para veículos de frota via GPS a partir da plataforma Arduino, fornecendo informações sobre o posicionamento do veículo e sua velocidade através das posições geográficas, tudo isto em tempo real. O sistema faz o uso do microcontrolador Arduino Mega, este escolhido por sua versatilidade e vasta base de dados junto à um shield SIM900, responsável pelo envio de dados para a rede. Os dados enviados são provenientes de um módulo GPS para Arduino, o Neo-6M, onde este coleta os dados geográficos que servem como base para o projeto. A plataforma IoT Thingier.io é responsável por receber esses dados. Com esta plataforma é possível receber os dados, guardá-los em uma espécie de banco de dados e transformá-los em informações de fácil compreensão para o usuário.

O resultado final apresenta os dados coletados pelo protótipo e a transformação dos mesmos em uma interface mais amigável ao usuário, possibilitando com que o mesmo possa aproximar uma possível rota feita pelo veículo em um determinado intervalo de tempo. Além disso, em caso de excesso de velocidade, a pessoa responsável é notificada por meio de um e-mail, facilitando a comunicação e o registro deste tipo de infração.

Observando os estudos e o sistema aqui desenvolvido, é possível confirmar que o sistema proposto no início foi alcançado com sucesso. Traçando um paralelo com o trabalho apresentado na Seção 2.2, é possível notar similaridades na implementação, como os equipamentos utilizados, e diferindo principalmente na parte que diz respeito a interface utilizada para mostrar o resultado ao usuário.

Para futuros aprimoramentos do sistema aqui apresentado, algumas sugestões são propostas. Inicialmente, o desenvolvimento de um front-end que permita ao usuário acessar os dados por meio de uma interface amigável, com a personalização necessária para que o cliente visualize os dados de sua escolha. Além disso, integrar com o Google Maps para corrigir a rota real a partir dos pontos geográficos obtidos. Outra melhoria está em aumentar as funcionalidades do projeto, aproveitando os componentes já utilizados. A melhoria do tempo de amostragem atual de 1 minuto para valores menores, como cerca de 15 segundos, possibilitará a criação de rotas mais precisas. Por fim, implementar um sistema de alimentação que não dependa de fontes externas. Para fins comerciais, seria necessário a modificação do projeto e modo a evitar o uso da plataforma Arduino, utilizando-se de um projeto específico para a implementação.

REFERÊNCIAS

ARDUINO About | Arduino. [S.l.: s.n.]. Disponível em:

<https://www.arduino.cc/en/about/>. Acesso em: 18 out. 2023.

ARDUINO Mega 2560 Rev3. [S.l.: s.n.]. Disponível em:

<https://store-usa.arduino.cc/products/arduino-mega-2560-rev3?selectedStore=us>.

Acesso em: 18 out. 2023.

COSTA, Sudip Evans; AHMED, Saniah; RAHMAN, Shandee. **Real Time Vehicle Tracking System**. 2015. BRAC University, Department of Electrical e Electronics Engineering.

EFCOM. **EFcom/GPRS Shield**. [S.l.], 2013. Technical Datasheet, Accessed:

2023-09-21. Disponível em: https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.electfreaks.com%2Fstore%2Fdownload%2Fproduct%2FEFcom%2FEFcom_Datasheet.pdf&psig=AOvVaw2e2tdpFKPjfkMDuiU2Yi-2&ust=1719652876444000&source=images&cd=vfe&opi=89978449&ved=0CAYQrpoMahcKEwj4hdfA_P2GAxUAAAAAHQAAAAAQBw.

ELETROGATE. **Tipos de Placas Arduino**. [S.l.: s.n.], 2023. Acessado em:

30-junho-2024. Disponível em: <https://blog.eletrogate.com/tipos-de-placas-arduino/>.

ESTADÃO - 32% das empresas brasileiras esperam aumento da frota de veículos -

Estadão. [S.l.: s.n.]. Disponível em: <https://www.estadao.com.br/economia/coluna-do-broad/32-das-empresas-brasileiras-esperam-aumento-da-frota-de-veiculos/>. Acesso em: 18 out. 2023.

HIVEMQ. **MQTT MQTT 5 Essentials**. [S.l.]: hivemq.com, 2021. A comprehensive overview of MQTT facts and features for beginners and experts alike.

JORNAL do Carro - Veja lista das multas de trânsito mais aplicadas no Brasil.

[S.l.: s.n.]. Disponível em: <https://jornaldocarro.estadao.com.br/servicos/veja-lista-das-multas-de-transito-mais-aplicadas-no-brasil/>. Acesso em: 18 out. 2023.

LEE, Edward A.; SESHIA, Sanjit A. **Introduction to embedded systems: a cyber-physical systems approach**. Second edition. Cambridge, Massachusetts: MIT Press, 2017. ISBN 978-0-262-53381-2.

LI, Qing; YAO, Caroline. **Real-time concepts for embedded systems**. San Francisco, CA: CMP Books, 2003. ISBN 978-1-57820-124-2.

MARINO, Tiago Badre. **Como funciona o GPS?** [S.l.]: Instituto de Agronomia - UFRRJ.

MAURYA, Kunal; SINGH, Ravi; SINGH, Ritu. Real Time Vehicle Tracking System using GSM and GPS Technology- An Anti-theft Tracking System. **International Journal of Electronics and Computer Science Engineering**, v. 1, n. 3, p. 1103–1107, 2012. ISSN 2277-1956.

MONICO, João Francisco Galera. **Posicionamento pelo NAVSTAR-GPS: descrição, fundamentos e aplicações**. [S.l.]: Editora UNESP, 2000.

MUSTAFA, Ali. Cloud-Based Vehicle Tracking System. en. v. 2, n. 4, 2019.

OLIVEIRA, Sérgio de. **Internet das Coisas com ESP8266, Arduino e Raspberry PI**. São Paulo, SP: Novatec Editora LTDA., 2017. ISBN 978-85-7522-582-0.

PINOUT - ARDUINO Mega Rev3 Board - ATmega328PU. [S.l.: s.n.]. Disponível em: <https://www.elcircuits.com/2021/08/pinout-arduino-mega-rev3-board.html>. Acesso em: 18 out. 2023.

PIROTTI, Rodolfo; ZUCCOLOTTO, Marcos. Transmissão de dados através de telefonia celular: arquitetura das redes GSM e GPRS. pt. **Revista Liberato**, v. 10, n. 13, p. 81–90, 2019. ISSN 15188043, 21788820.

TEODOLINI, Alezi. **Receptores GPS de navegação e Mapeamento**. [S.l.: s.n.].

THINGER.IO. **Thinger.io Documentation**. [S.l.: s.n.], 2024. Accessed: 2024-06-28. Disponível em: <https://docs.thinger.io/>.

TWILIO. **What is MQTT?** [S.l.: s.n.], 2023. Accessed: 2024-06-28. Disponível em: <https://www.twilio.com/en-us/blog/what-is-mqtt>.

U-BLOX. **NEO-6 u-blox 6 GPS Modules Data Sheet**. [S.l.], 2011. Technical Datasheet, Accessed: 2023-09-21. Disponível em: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fcontent.u-blox.com%2Fsites%2Fdefault%2Ffiles%2Fproducts%2Fdocuments%2FNEO->

6_DataSheet_%2528GPS.G6-HW-09005%2529.pdf&psig=AOvVaw1CivGqAJ8zhex_R7hH4SK-&ust=1719652758903000&source=images&cd=vfe&opi=89978449&ved=0CAYQrpoMahcKEwj4IIHY_f2GAxUAAAAAHQAAAAAQBA.

VRUM - Mortes no trânsito: Brasil é o terceiro colocado do ranking mundial. [*S.l.: s.n.*]. Disponível em: <https://www.vrum.com.br/noticias/mortes-transito-brasil-terceiro/>. Acesso em: 18 out. 2023.