

# Redes e Sistemas Autónomos

Relatório: Projeto Final

Eduardo Alves: n<sup>o</sup>mec 104179  
eduardoalves@ua.pt

---

**Github:**  
[https://github.com/eduardoalves001/RSA\\_Projeto](https://github.com/eduardoalves001/RSA_Projeto)

# Contents

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Contextualização . . . . .	1
1.2	Responsabilidades e Desafios . . . . .	1
1.3	Como está dividido o repositório? . . . . .	1
<b>2</b>	<b>Frontend</b>	<b>2</b>
2.1	Tecnologias e funcionamento do sistema . . . . .	2
2.2	Como foram feitas as representações dos veículos? .	2
2.3	Como foram representadas as rotas? . . . . .	2
<b>3</b>	<b>Scripts</b>	<b>3</b>
3.1	Quais foram os scripts utilizados? . . . . .	3
3.2	generate_crashed.py . . . . .	3
3.3	generate_ambulance.py . . . . .	3
3.4	generate_car.py . . . . .	3
3.5	proxy.py . . . . .	3
<b>4</b>	<b>Docker</b>	<b>4</b>
4.1	Tecnologias utilizadas . . . . .	4
4.2	Como funciona? . . . . .	4
4.3	Que veículos é que cada OBU representa? . . . . .	4
<b>5</b>	<b>Resultados</b>	<b>4</b>
5.1	Fluxo de programas . . . . .	4
5.2	Diferentes OBUs . . . . .	5
5.3	Troca de mensagens no Docker Compose . . . . .	5
5.4	Chamada do Frontend . . . . .	6
5.4.1	Sem executar as funções generate . . . . .	6
5.4.2	Ao executar as funções generate . . . . .	7
<b>6</b>	<b>Conclusão</b>	<b>8</b>

# 1 Introdução

## 1.1 Contextualização

A crescente adoção de tecnologias de comunicação veicular tem impulsionado o desenvolvimento de soluções inteligentes para a segurança rodoviária, gestão de tráfego e eficiência dos transportes. Neste contexto, existe um padrão europeu designado ITS-G5 onde se destacam dois tipos fundamentais de mensagens:

- CAMs (Cooperative Awareness Messages) – enviadas periodicamente por veículos para informar sobre a sua posição, direção, velocidade, entre outros parâmetros essenciais para a perceção situacional da rede.
- DENMs (Decentralized Environmental Notification Messages) – utilizadas para notificar eventos perigosos, como acidentes, condições perigosas da via ou obstáculos na estrada.

Este projeto insere-se nesse cenário, cujo propósito é simular um sistema de alerta de acidentes entre veículos com base na troca de mensagens CAM e DENM. A simulação é realizada com o auxílio da plataforma VANETZA, que implementa o protocolo ITS-G5 sobre MQTT, e permite uma interação realista entre OBUs (On-Board Units). Adicionalmente, foi criada uma interface visual com Leaflet e GPX para representar rotas reais e o comportamento dos veículos durante a simulação. Através desta estrutura, é possível simular cenários de acidente onde: Um veículo emite uma mensagem DENM ao detetar um acidente, os veículos próximos reagem, desviando-se automaticamente da rota original e onde uma ambulância é acionada e enviada automaticamente para o local do acidente.

## 1.2 Responsabilidades e Desafios

Um dos principais desafios deste projeto foi o facto de ter sido desenvolvido individualmente, apesar de estar pensado para ser realizado em grupo de dois. Isso exigiu uma gestão rigorosa do tempo e esforço acrescido para lidar com todas as componentes. A integração entre os diferentes sistemas, como a VANETZA em Docker, a troca de mensagens CAM e DENM via MQTT, e a visualização de rotas com Leaflet e ficheiros GPX trouxeram também algumas dificuldades técnicas, especialmente na subscrição correta de mensagens e na sincronização dos veículos com os eventos de acidente. Um problema também encontrado foi que as OBUs 2 e 3 se teleportam para o local de evento da DENM enviada pela OBU1, mas rapidamente voltam à sua posição no trajeto da rota, muitas vezes alternando entre a sua posição real e a posição da OBU1.

## 1.3 Como está dividido o repositório?

Pasta	Funcionalidade
Frontend	Pasta com o código relativo à visualização da simulação.
Scripts	Pasta com o código relativo aos scripts de cada veículo.
Docker	Pasta com o código relativo ao Docker.

Table 1: Divisão de código

## 2 Frontend

### 2.1 Tecnologias e funcionamento do sistema

Este sistema conta com um frontend construído com o framework Flask, responsável por servir a interface web e disponibilizar os dados dos veículos através de uma rota REST. A interface gráfica é implementada utilizando a biblioteca JavaScript Leaflet.js, que permite a criação de um mapa interativo, onde é possível visualizar, em tempo real, a posição atual de veículos e a localização do acidente rodoviário, com atualizações constantes a partir dos dados recebidos pelo backend. O uso do Leaflet permite não só exibir marcadores dinâmicos representando carros, ambulâncias e o acidente, como também carregar rotas reais em formato GPX, que são renderizadas diretamente no mapa para contextualizar os percursos e cenários simulados. No contexto do projeto existem duas rotas:

- `rotagpx` (Rota principal) - Rota principal de todos os veículos da simulação.
- `rota_alternativa.gpx` (Rota secundária) - Rota alternativa que é usada pelo carro normal (OBU3) ao receber a DENM do carro acidentado (OBU1).

Dessa forma, o sistema proporciona uma visualização clara, intuitiva e atualizada do tráfego e dos eventos rodoviários relevantes.

### 2.2 Como foram feitas as representações dos veículos?

As representações dos veículos foram feitas usando ícones personalizados para cada tipo de veículo, o que permite distingui-los facilmente no mapa. O Carro normal, ambulância e o veículo envolvido no acidente têm ícones diferentes, esta diferenciação ajuda os utilizadores a identificar rapidamente o papel ou o estado de cada veículo na simulação. Além disso, o sistema fornece uma rota REST (`/vehicles`) que retorna a posição (longitude e latitude) de todos os veículos, permitindo que o frontend atualize a localização dos marcadores no mapa em tempo real.

### 2.3 Como foram representadas as rotas?

As rotas foram geradas através da plataforma do Google maps, utilizando após isso um conversor de rotas para um formato `.gpx`, de forma a serem traçadas e utilizadas juntamente com a biblioteca do Leaflet, na secção de frontend e para definir os trajetos de cada veículo no backend. É possível visualizar e analisar diretamente as rotas no mapa, uma vez que estas estão destacadas de forma clara. Isso permite aos utilizadores acompanhar facilmente o trajeto definido para os veículos e compreender melhor o percurso que será seguido durante a simulação.

## 3 Scripts

### 3.1 Quais foram os scripts utilizados?

O projeto contou com quatro scripts principais: `generate_crashed.py`, que simula o veículo acidentado e envia alertas DENM; `generate_ambulance.py`, responsável pela ambulância que reage ao acidente, enviando e recebendo CAMs e recebendo DENMs; `generate_car.py`, que simula veículos normais em trânsito enviando e recebendo mensagens CAM, tal como recebendo também DENMs; e o `proxy.py`, que atua como intermediário, gerenciando a comunicação MQTT entre os veículos e o frontend, garantindo a correta transmissão dos dados no sistema. Foi também criado um ficheiro bash designado `run.sh` que permite instanciar em simultâneo os vários ficheiros `generate`, permitindo assim, ativar o comportamento dos vários veículos de forma eficiente.

### 3.2 `generate_crashed.py`

Este script lê um ficheiro GPX para definir uma posição fixa de acidente e envia repetidamente mensagens DENM MQTT com essa posição para alertar os restantes veículos do acidente onde esteve envolvido.

### 3.3 `generate_ambulance.py`

Este script lê um percurso de um ficheiro GPX e caso receba uma mensagem DENM, simula o movimento de uma ambulância ao longo desse trajeto até chegar à localização do acidente, enviando periodicamente mensagens CAM com a sua posição via MQTT. Escuta mensagens DENM e CAM de outros veículos de forma a conseguir reagir ao acidente de forma eficiente.

### 3.4 `generate_car.py`

Este script simula o movimento de um veículo seguindo uma rota definida num ficheiro GPX, enviando periodicamente mensagens CAM com a sua posição via MQTT. Também ouve mensagens DENM e CAM de outros veículos para, por exemplo, mudar para uma rota de desvio em caso de acidente.

### 3.5 `proxy.py`

Este script conecta-se a múltiplos brokers MQTT representando veículos (OBUs), escuta mensagens CAM e DENM de cada um, processa as posições recebidas e encaminha essas informações simplificadas para um broker central que serve o frontend. Ele garante conexão estável, tratamento de mensagens e desligamento ordenado dos clientes MQTT.

## 4 Docker

### 4.1 Tecnologias utilizadas

Para facilitar a simulação de múltiplas OBUs (On-Board Units) e a comunicação entre elas, foi utilizado o Docker Compose. Esta ferramenta permite definir e gerir ambientes com múltiplos contêineres de forma simples, através de um único ficheiro YAML. Adicionalmente, foi incluído um serviço mosquitto-websocket, que disponibiliza um broker MQTT com suporte a WebSockets.

### 4.2 Como funciona?

O ficheiro docker-compose.yml define três serviços principais, cada um representando uma OBU (obu\_one, obu\_two e obu\_three), que executam a imagem do VANETZA, responsável por implementar as comunicações do protocolo do ITS-G5. Cada OBU é configurada com um station\_id, endereço MAC e IP fixo na rede vanetzalan0, garantindo a sua identificação e comunicação na rede simulada.

### 4.3 Que veículos é que cada OBU representa?

A OBU\_one (OBU1) cujo endereço ipv4 é 192.168.98.10 corresponde ao veículo acidentado, ou seja, o veículo que emite as mensagens DENM. A OBU\_two (OBU2) cujo endereço ipv4 é 192.168.98.20 corresponde ao veículo da ambulância, que recebe as DENMs da OBU\_one, faz a troca de CAMs com os restantes veículos e se dirige em direção ao acidente para auxiliar. E a OBU\_three (OBU3), cujo endereço ipv4 é 192.168.98.30 corresponde ao carro normal, que também recebe as DENMS do veículo acidentado, faz a troca de CAMs com os restantes veículos e que devido ao acidente terá de alterar a sua rota de forma a não perturbar no auxílio do acidente.

## 5 Resultados

### 5.1 Fluxo de programas

Para começar a executar o projeto devemos seguir os seguintes passos:

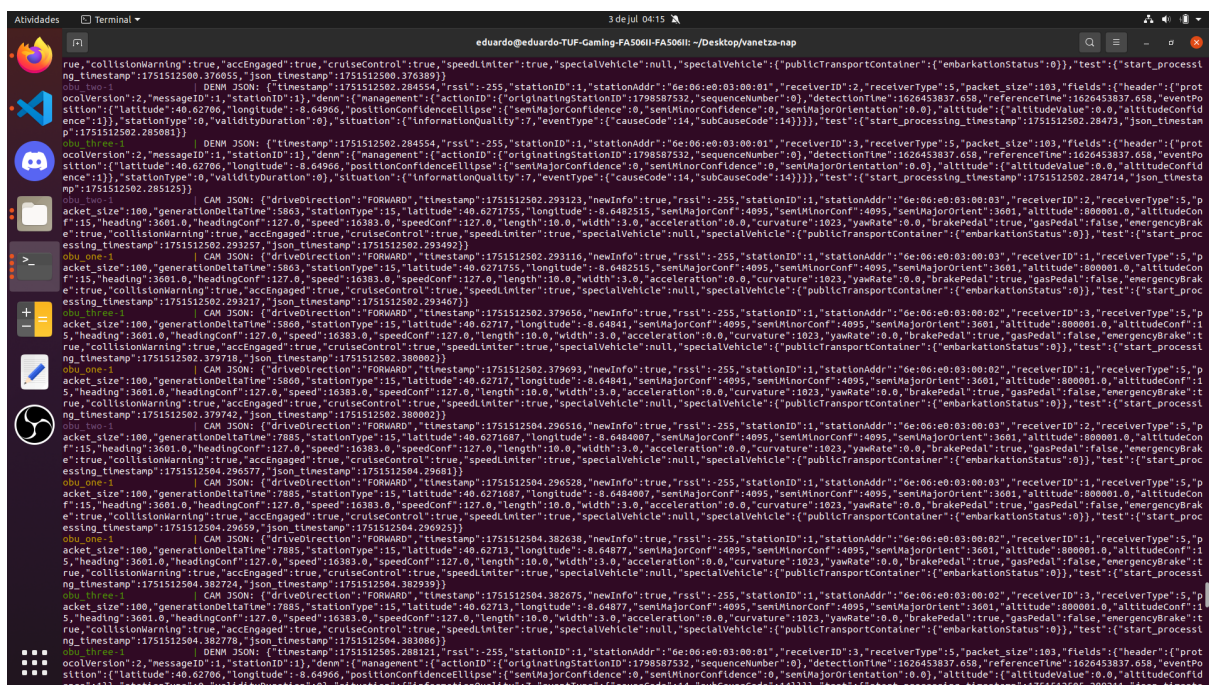
- Executar o docker compose: Docker compose up –build.
- Executar o ficheiro do proxy; python3 proxy.py
- Executar o ficheiro app.py: python3 app.py
- Executar o ficheiro bash: ./run.sh (executa os 3 ficheiros de generate).

## 5.2 Diferentes OBUs

O projeto começa com o veículo acidentado a enviar DENMs, mas podemos escolher retirar no ficheiro `generate_crashed.py` o uso da função `send_denm()`, e desta forma, a OBU1 deixa de enviar DENMs. O que acontece em cada caso é o seguinte: A OBU1 que corresponde ao carro acidentado está numa posição fixa no centro da rota principal, caso este veículo envie uma DENM, a OBU2 correspondente à ambulância dirige-se ao local do acidente e a OBU3 correspondente ao carro normal utiliza a rota secundária. Caso a OBU1 não envie uma DENM, tanto a OBU 2 (ambulância) como a OBU3 (carro normal) seguem a rota normal, ignorando o acidente.

## 5.3 Troca de mensagens no Docker Compose

De seguida demonstro um pequeno excerto da troca de mensagens no Docker Compose, onde se pode analisar que a OBU 2 e 3 recebem DENMs, e à troca de CAMs entre os vários veículos presentes na simulação.



```
Atividades Terminal 3 de jul 04:15
eduardo@eduardo-TUF-Gaming-FA50611-FA50611: ~/Desktop/vanetza-nap

true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512500.376055, "json_timestamp": 1751512500.376099}}

DENM JSON: {"timestamp": 1751512502.284554, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:01", "receiverID": 2, "receiverType": 5, "packet_size": 103, "fields": {"header": {"protocolVersion": 2, "messageID": 1, "stationID": 1}, "denm": {"management": {"actionID": {"originatingStationID": 1798587532, "sequenceNumber": 0}, "detectionTime": 1626453837.658, "referenceTime": 1626453837.658, "eventPosition": {"latitude": 40.62706, "longitude": -8.64966, "positionConfidence": {"senMajorConfidence": 0, "senMinorConfidence": 0, "senMajorOrientation": 0.0}, "altitude": {"altitudeValue": 0.0, "altitudeConfidence": 0}}, "stationType": 0, "validityDuration": 0}, "situation": {"informationQuality": 7, "event": {"causeCode": 14, "subCauseCode": 14}}}}, "test": {"start_processing_timestamp": 1751512502.284714, "json_timestamp": 1751512502.285011}}

DENM JSON: {"timestamp": 1751512502.284554, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:01", "receiverID": 3, "receiverType": 5, "packet_size": 103, "fields": {"header": {"protocolVersion": 2, "messageID": 1, "stationID": 1}, "denm": {"management": {"actionID": {"originatingStationID": 1798587532, "sequenceNumber": 0}, "detectionTime": 1626453837.658, "referenceTime": 1626453837.658, "eventPosition": {"latitude": 40.62706, "longitude": -8.64966, "positionConfidence": {"senMajorConfidence": 0, "senMinorConfidence": 0, "senMajorOrientation": 0.0}, "altitude": {"altitudeValue": 0.0, "altitudeConfidence": 0}}, "stationType": 0, "validityDuration": 0}, "situation": {"informationQuality": 7, "event": {"causeCode": 14, "subCauseCode": 14}}}}, "test": {"start_processing_timestamp": 1751512502.284714, "json_timestamp": 1751512502.285125}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512502.293123, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:03", "receiverID": 2, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 5863, "stationType": 15, "latitude": 40.6271755, "longitude": -8.6482515, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512502.293257, "json_timestamp": 1751512502.293492}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512502.293116, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:03", "receiverID": 1, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 5863, "stationType": 15, "latitude": 40.6271755, "longitude": -8.6482515, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512502.293217, "json_timestamp": 1751512502.293467}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512502.379656, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:02", "receiverID": 3, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 5860, "stationType": 15, "latitude": 40.62717, "longitude": -8.64841, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512502.379718, "json_timestamp": 1751512502.380002}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512502.379693, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:02", "receiverID": 1, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 5860, "stationType": 15, "latitude": 40.62717, "longitude": -8.64841, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512502.379742, "json_timestamp": 1751512502.380002}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512504.296516, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:03", "receiverID": 2, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 7885, "stationType": 15, "latitude": 40.6271687, "longitude": -8.6484087, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512504.296577, "json_timestamp": 1751512504.296811}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512504.296528, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:03", "receiverID": 1, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 7885, "stationType": 15, "latitude": 40.6271687, "longitude": -8.6484087, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512504.296589, "json_timestamp": 1751512504.296823}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512504.382638, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:02", "receiverID": 1, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 7885, "stationType": 15, "latitude": 40.62713, "longitude": -8.64877, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512504.382724, "json_timestamp": 1751512504.382939}}

CAM JSON: {"drivedirection": "FORWARD", "timestamp": 1751512504.382675, "newInfo": true, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:02", "receiverID": 3, "receiverType": 5, "packet_size": 100, "generationDeltaTime": 7885, "stationType": 15, "latitude": 40.62713, "longitude": -8.64877, "senMajorConf": 4095, "senMinorConf": 4095, "senMajorOrient": 3601, "altitude": 800001.0, "altitudeConf": 15, "heading": 3601.0, "headingConf": 127.0, "speed": 16383.0, "speedConf": 127.0, "length": 10.0, "width": 3.0, "acceleration": 0.0, "curvature": 1023, "yawRate": 0.0, "brakePedal": true, "gasPedal": false, "emergencyBrake": true, "collisionWarning": true, "accEngaged": true, "cruiseControl": true, "speedLimit": true, "specialVehicle": null, "specialVehicle": {"publicTransportContainer": {"embarkationStatus": 0}}, "test": {"start_processing_timestamp": 1751512504.382778, "json_timestamp": 1751512504.383086}}

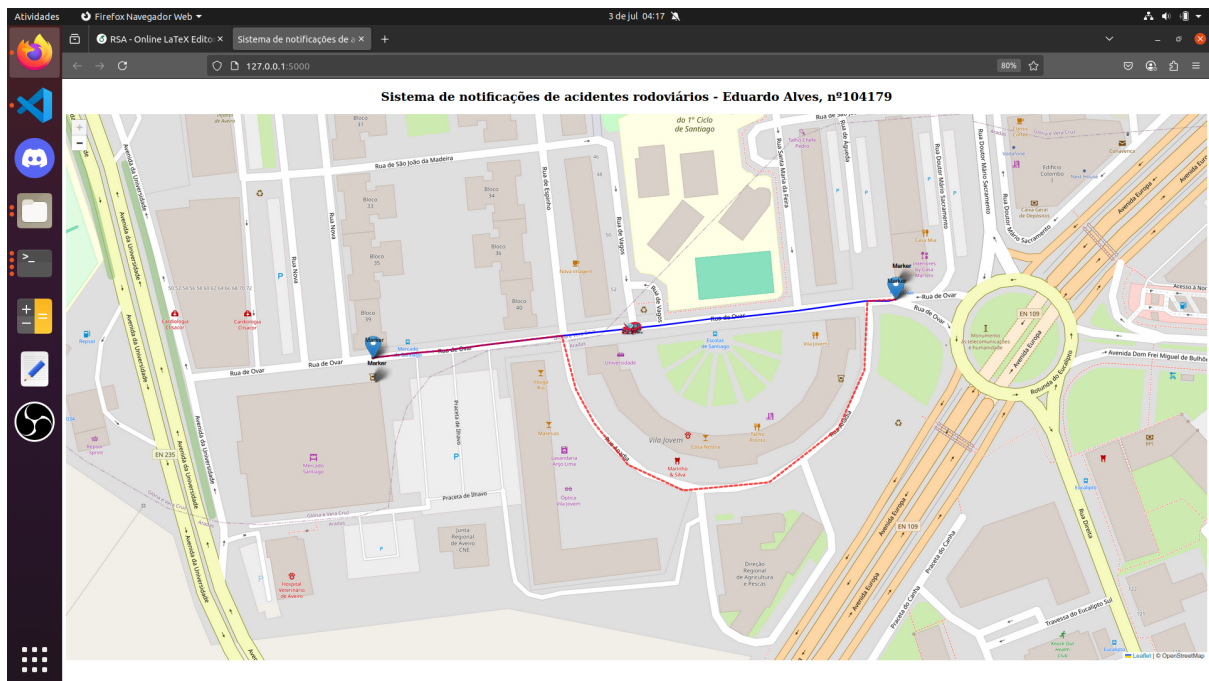
DENM JSON: {"timestamp": 1751512506.298121, "rssi": -255, "stationID": 1, "stationAddr": "6e:06:e0:03:00:01", "receiverID": 3, "receiverType": 5, "packet_size": 103, "fields": {"header": {"protocolVersion": 2, "messageID": 1, "stationID": 1}, "denm": {"management": {"actionID": {"originatingStationID": 1798587532, "sequenceNumber": 0}, "detectionTime": 1626453837.658, "referenceTime": 1626453837.658, "eventPosition": {"latitude": 40.62706, "longitude": -8.64966, "positionConfidence": {"senMajorConfidence": 0, "senMinorConfidence": 0, "senMajorOrientation": 0.0}, "altitude": {"altitudeValue": 0.0, "altitudeConfidence": 0}}, "stationType": 0, "validityDuration": 0}, "situation": {"informationQuality": 7, "event": {"causeCode": 14, "subCauseCode": 14}}}}, "test": {"start_processing_timestamp": 1751512506.298211, "json_timestamp": 1751512506.298211}}
```

## 5.4 Chamada do Frontend

De seguida demonstro o que acontece ao executar o app.py, sem e com a chamada das funções generate.

### 5.4.1 Sem executar as funções generate

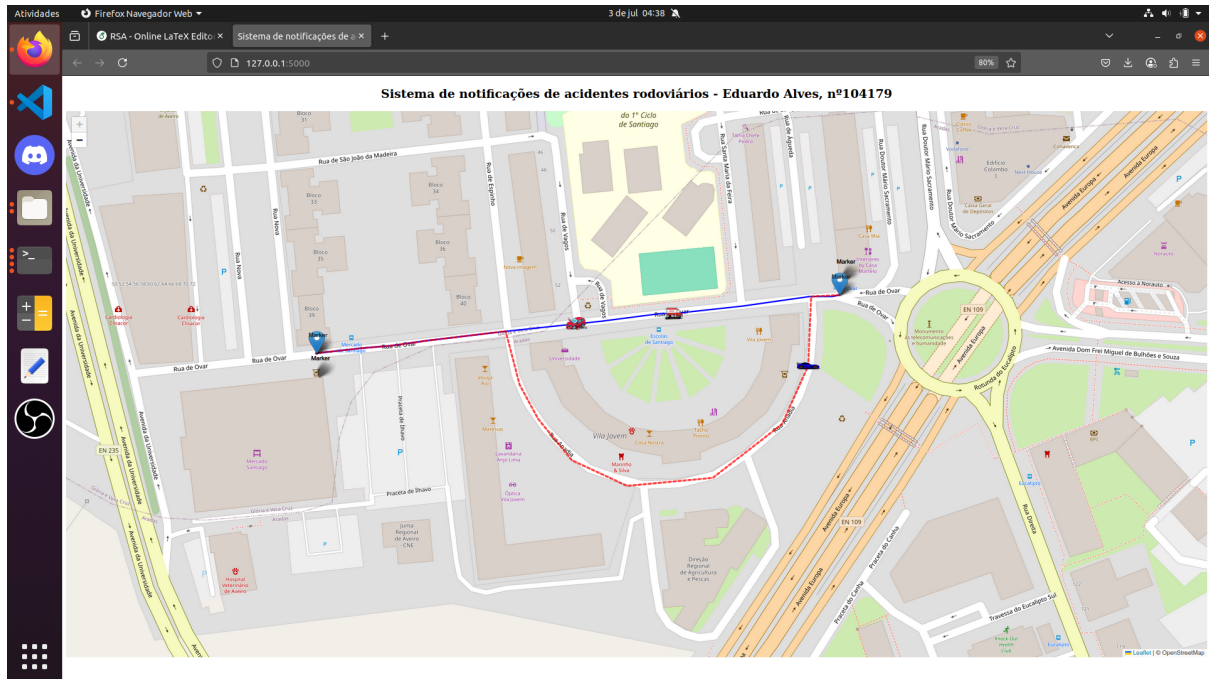
Caso não executemos as funções generate, simplesmente aparece o mapa com as duas rotas disponíveis, a azul a rota principal e a vermelho a rota secundária. Não aparece nenhum veículo, tirando o veículo correspondente à OBU1 (que está numa posição estática no centro da rota principal).



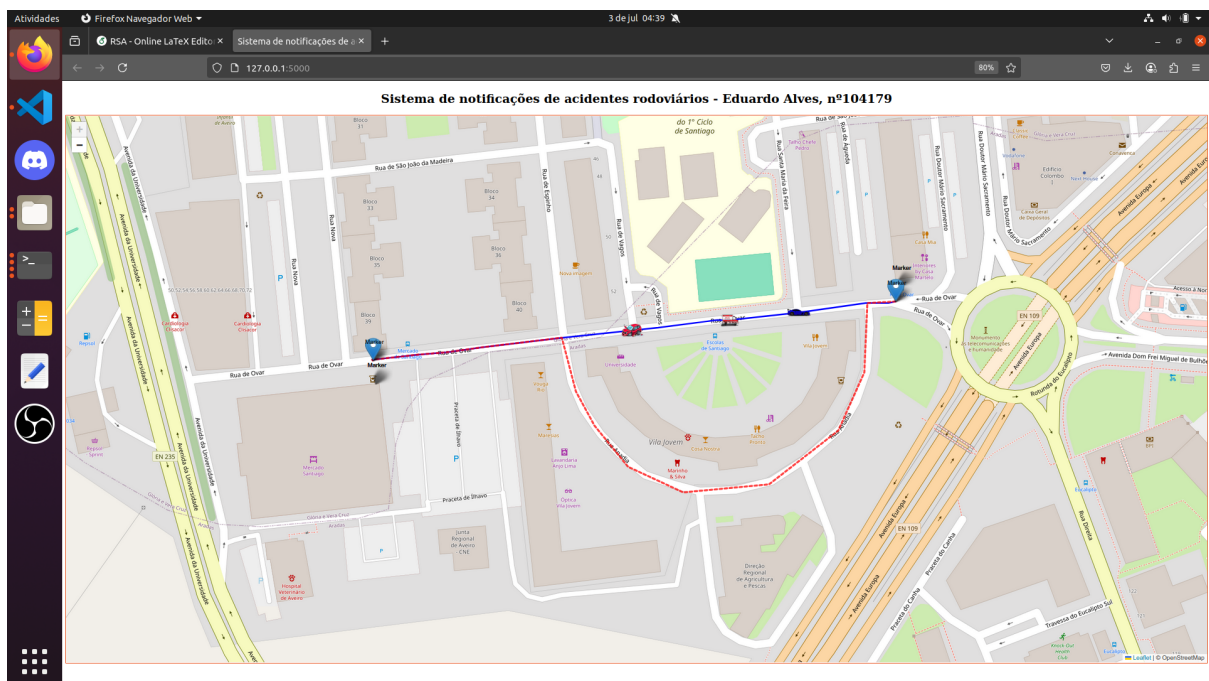


### 5.4.2 Ao executar as funções generate

Como mencionado previamente, ao executarmos as funções generate, caso escolhamos que a OBU1 envie DENMs, a ambulância irá parar junto ao veículo acidentado e o carro normal seguirá pela rota secundária. Caso escolhamos que a OBU1 não envie DENMs, a ambulância e o carro normal irão pela rota principal e ignorarão o acidente.



Se a OBU1 enviar DENMs



Se a OBU1 não enviar DENMs

## 6 Conclusão

O projeto conseguiu integrar com sucesso a comunicação entre os scripts Python de cada veículo, o proxy MQTT, o Docker Compose e o frontend responsável pela visualização veicular. Esta integração permitiu a simulação eficiente da troca de mensagens CAM e DENM. Garantiu também a atualização em tempo real das posições de cada um dos veículos e a resposta ao evento do acidente e desvios de rota, respectivamente, por parte da ambulância e por parte carro normal. Acredito que dentro das limitações referidas na introdução, o objetivo foi concluído com sucesso. A implementação demonstrou-se eficaz dentro do trabalho proposto inicialmente, cumprindo os requisitos principais do projeto.