CENTRO PAULA SOUZA

FATEC SANTO ANDRE

Tecnologia em Eletrônica Automotiva

Marcos de Lima Gallego

DESENVOLVIMENTO DE UMA ROADSIDE UNIT

SANTO ANDRÉ

2022

Marcos de lima Gallego

DESENVOLVIMENTO DE UMA ROADSIDE UNIT

Trabalho de graduação de curso apresentado ao curso Tecnólogo em Eletrônica Automotiva da Fatec Santo André, orientado pelo professor ME Weslley Medeiros Torres como requisito parcial para obtenção do Título de Tecnólogo em Eletrônica Automotiva.

FICHA CATALOGRÁFICA

G166d
Gallego, Marcos de Lima
Desenvolvimento de uma roadside unit / Marcos de Lima Gallego. - Santo André, 2022. – 88f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.
Curso de Tecnologia em Eletrônica Automotiva, 2022.

Orientador: Prof. Me. Weslley Medeiros Torres

1. Eletrônica. 2. Sistema de comunicação. 3. Veículos. 4. Implementação. 5. Roadside Unit. 6. Tecnologia. 7. Sensores. 8. Eventos externos. 9. Redes de comunicação. I. Desenvolvimento de uma roadside unit.

629.2

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

Santo André, 27 DE JUNHO DE 2022.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA: **"DESENVOLVIMENTO DE UMA RSU"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. WESLLEY MEDEIROS TORRES_____

MEMBROS:
PROF. FERNANDO GARUP DALBO_____

PROF. PAULO TETSUO HOASHI_____

ALUNO:

MARCOS DE LIMA GALLEGO_____

aos meus pais, que já se foram, Julian Gallego e Alcir Cecilia de Lima Gallego.

AGRADECIMENTOS

Aos professores do curso, que se dedicaram, a ensinar, e pela atenção dispensada, pelos colegas que foram, auxiliares, no aprendizado, ao corpo de funcionários, que sempre, disponibilizaram condições adequadas, e ao centro Paula Souza, que disponibilizou este curso, que me atendeu, em meus, objetivos de aprendizado, obrigado.

“Um gênio se faz com 1% de inspiração e 99% de esforço”. Thomas Alva Edson

RESUMO

O carro, agora, agregou muitos controles, que auxiliam na dirigibilidade, evitam acidentes, mas nem todos os seus sensores, são suficientes, para monitorar, eventos externos, tais como, sinalização, tráfego, e funcionabilidades, no que se refere a infraestrutura, neste trabalho, é realizada a implementação de uma *Roadside Unit* (RSU) com esp32. Hoje existe o sistema *V2X*, que pode auxiliar, o sistema eletrônico do veículo, a tomar decisões, ou pelo menos, estar em comunicação, com fatores externos, que estejam no mesmo, padrão de comunicação. Como o carro, está se tornando, em termos “autônomo”, tudo o que que for em direção de sua automatização, somarem seus recursos. Podemos citar como exemplo, o uso da rede *V2I*; Três importantes componentes na comunicação *V2I* são a unidade de bordo (*OBU*) montada no veículo, o *Roadside Unit (RSU)* na infraestrutura rodoviária e no disponível canal de transmissão, que pode ser utilizada, tanto em estacionamentos, que por serem de dimensões grandes, ou vias expressas, que podem ter avisos, tanto de congestionamentos, como obras na pista. Podemos citar a rede V2P, que através do sistema do carro, em comunicação com o celular, pode precaver, a colisão, ou uma manobra perigosa.

Palavras-chave: Carro. Eventos. *V2x. RSU. Esp32. Wi-Fi.*

ABSTRACT

The car has now added many controls, which help in driving, avoid accidents, but not all of its sensors are enough to monitor external events, such as signaling, traffic, and functionalities, with regard to infrastructure, in this work, the implementation of a Roadside Unit (RSU) with esp32 is performed. Today there is the V2X system, which can help the vehicle's electronic system to make decisions, or at least be in communication, with external factors that are in the same communication pattern. Like the car, it is becoming, in "autonomous" terms, everything that goes towards its automation, adding up its resources. We can cite as an example, the use of the V2I network; Three important components in V2I communication are the on-board unit (OBU) mounted on the vehicle, the Roadside Unit (RSU) in the road infrastructure and in the available transmission channel, which can be used both in parking lots, and because of their large dimensions, or expressways, which may have warnings of both congestion and road works. We can mention the V2P network, which through the car system, in communication with the cell phone, can prevent a collision, or a dangerous maneuver.

Key words: Car. Events. V2x. RSU. Esp32. Wi-Fi.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

AGs. Algoritmos genéticos.

CAN Controller Area Network

CBC-MAC Cifra do Block encadeado para autenticação.

CCMP. Protocolo de cód.de msg de encadeamento do bloco de cifra do modo contador.

CI. Circuito Integrado.

CTR. Número de cliques dado em um anúncio.

C-V2P. Comunicação de pedestre para veículo, via celular

DSRC. *(Dedicate Short Range Communications).*

ESP 32. Microcontrolador de baixo custo.

GARSUD. Sistema de inteligência artificial.

HW. Hardware.

IA. Inteligência Artificial.

ITS. Sistema de transporte Inteligente.

LIPO Lithium Polymer Battery; Bateria de polímero e lítio.

LPWAN. Low Power WAN, redes de baixo consumo e alto alcance.

OBU. Unidade *On-board*.

OSI. Padrão para protocolos de rede de comunicação.

V2I. Comunicação Veículo para Infraestrutura.

V2N. Comunicação Veículo para *NETWORK*

V2P. Comunicação veículo para pedestre

V2X. Comunicação Veículo para qualquer estrutura de comunicação

RSU. Unidade *Roadside*.

SW. Software (Programa)

SSID service set Identifier (Identificador do conjunto de exercício)

SUMÁRIO

1.INTRODUÇÃO:

Com a evolução dos automóveis, o trânsito cresceu proporcionalmente, tornando natural a necessidade dos automóveis se comunicarem no seu estado atual. Através da conexão do automóvel, podemos acreditar que se tornem mais eficientes e exista a conexão entre eles, dessa forma, os dados relacionados aos diferentes automóveis, podem ser utilizados de forma a tornar a direção mais segura e confiável. Com isso, esses dados podem ser usados para evitar acidentes, ou para indicar o fluxo de automóveis, em determinada via ou estacionamento. Essa comunicação entre automóveis é chamada de *Vehicle to Vehicle (V2V)* e com infraestrutura chamada de *Vehicle to infraestruture (V2I),* as informações compartilhadas podem ser a velocidade, eficiência de tráfego e conforto. Os principais meios de comunicação empregados para a troca de dados entre automóveis são a rede de telefonia celular e a Comunicação Dedicada de Curto Alcance (*Dedicated Short Range Communications – DSRC*). De acordo com a Cisco, o número de dispositivos interconectados será superior a 11,6 bilhão em 2021, e o volume de dados serão maiores que 49 exabytes. Portanto o uso de rede de celulares tende a aumentar a carga no sistema de telefonia celular, degradando o desempenho e tornando inviável seu uso para aplicações automotivas.

A tecnologia de comunicação de curto alcance foi escolhida pela Comissão Federal de Comunicações dos Estados Unidos (*FCC – Federal Concil of Communication*), para aplicações automotivas, inclusive sendo utilizada na América do Norte, Europa e Japão. Porém, existem algumas limitações de tecnologia, como pequeno alcance e que podem impossibilitar a troca de informações entre os veículos. Para contornar este problema, são utilizadas unidades de beira de estrada, chamadas de *Road SIDE Units,* dessa forma, permitem que a informação chegue ao destino desejado. A efetividade do V2I possui uma forte dependência do posicionamento e funcionamento correto das *RSUs* para evitar pontos cegos que podem causar acidentes.

Dentre as tecnologias existentes para melhorar a cobertura das RSU, surge a possibilidade de uso do *LoRa (Long Range)*, que é utilizado para a cobertura de grandes áreas e o uso da frequência inferior à 1 GHz, o que torna capaz de se propagar mais facilmente, e com menores perdas devido à propagação em espaço livre.

Nesse contexto, esse trabalho pretende realizar a implementação de uma *RSU* aplicando a tecnologia *LoRa* com foco na comunicação veicular, além de realizar testes em campo para a avaliação do protótipo construído (análise de métricas de desempenho na taxa de recepção, potência do sinal e relação sinal-ruido).

1.1 Objetivos:

O objetivo geral desse trabalho é a construção de um protótipo de RSU utilizando a tecnologia *LoRa* e sua avaliação na comunicação entre veículos. Para isso, os objetivos específicos são:

- Caracterizar as tecnologias de *DSRC* e *LoRa*
- Construção de protótipos de RSU – OBU e instalação em veículos;
- Testes de campo com o sistema desenvolvido;
- Análise de desempenho, utilizando métricas de desempenho na taxa de recepção, potência do sinal e relação sinal-ruido;

No meu intuito de estudo, venho a abranger o CI, ESP 32, que pode vir com sistema Wi-Fi, Microprocessador, e protocolo de comunicação adequado, sendo até que existe a possibilidade de em breve, principalmente no estudo da comunicação V2I, também podem vir os celulares, com esta tecnologia C-V2P, integrada, podendo colaborar, para que os veículos, “enxerguem” os pedestres nas vias.

1.2. Motivação

Devido à evolução das comunicações automotivas, com relação a infraestrutura, com os veículos, o assunto da comunicação V2I, veio à tona, com uso de alguns sistemas de comunicação.

1.3. Estrutura do trabalho

Devido ao fato do ESP32, atender as expectativas de hardware, foi utilizado para montar um sistema V2I, através de OBUS e uma RSU. O sistema de hardware, foi implementado em placas perfuradas, e soldadas.

O trabalho, foi executado na parte de programação utilizando o IDE-Arduino.

2. DESENVOLVIMENTO

Um dos conceitos almejados, pelos gigantes da tecnologia é o *V2X*.

Uma forma de conectar o carro com tudo o que está ao seu redor, para tornar o trânsito mais confortável, eficiente e seguro.

Podemos subdividir o *V2x,* em *V2V*, Veículo para Veículo*, V2I*; Veículo para infraestrutura, *V2P*; Veículo para Pedestre, ou *C-V2P*; *V2N*, Veículo para rede *Network.* Uma demonstração de redes V2X, é indicada na Figura 1.

Figura 1; Rede *V2X*.

Fonte: NETO, Neri. V2x o mais novo conceito em Tecnolog. para carros conectados e inteligentes. 2019.

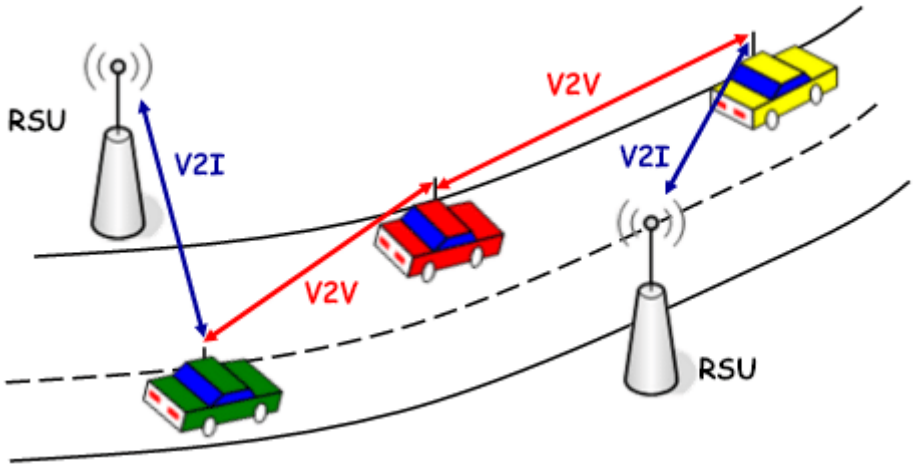
2.1. Como funciona e o que é o *V2X*

V2X é um estudo para carros autônomos, assistidos e conectados.
É uma forma de referência para a evolução tecnológica em veículos.
Ela deu origem ao DSRC *(Dedicate Short Range Communications)* e o C-V2X (Cellular Vehicle to x). As duas vem do mesmo princípio, sendo que o *C-V2X* é uma evolução da *DSRC*.

2.1.1. Comunicação *V2I*

A comunicação V2I, nada mais é do que a comunicação entre o veículo e a infraestrutura, três componentes principais fazem parte da comunicação *V2I*; são eles: OBU; Unidade *Onboard*, e a que vai dentro do veículo; *RSU; Roadside Unit*, e a unidade, que vai na via, na infraestrutura, normalmente, alimentado, por uma bateria, que é carregada, por uma célula solar; e o canal de transmissão disponível, a rede Broadcasting, ou via celular, ou *DSRC*; Comunicação Dedicada de Curto Alcance. Também temos o Centro de Controle Comunicação V2I e V2V. Como exemplo de V2I e V2V, temos a Figura 2.

Figura 2. Comunicação V2I.



Fonte: MELO Pablo, conhecendo o V2X, conectando veículos para tudo-Embarcados. 2017

Na Figura 3, é demonstrada a comunicação V2I para RSU com base em tráfego inteligente e comunicação com a infraestrutura.

Figura 3*: ITS*

Fonte: *ROSIN Zhongsham;2022*

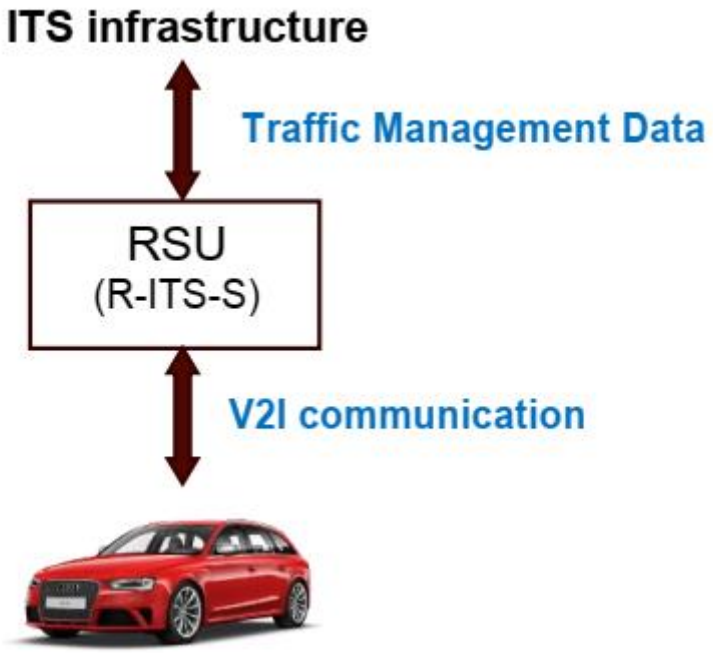
Uma estação *Roadside ITS (R-ITS-S)* representa a entidade que fornece:

1) Comunicação direta com veículos, via rádio-veículo-infraestrutura(V2I) transmissão.
2) Interface de comunicação para equipamentos de infraestrutura *ITS* central, como um Gestor de Tráfego ou Centro de Controle de Tráfego (TCC) para troca de dados de gerenciamento. Isto é ilustrado na figura 3.

Um dos termos comumente usados, como sinônimo para a estação ITS, à beira da estrada, é a expressão mais abreviada *Roadisite Unit (RSU).*

O termo *ITS Roadside.* Estação *(IRS) também* é utilizado para denotar um ITS

Figura 4. Funções de comunicação de uma *Roadside Unit (RSU)*.



Fonte: *Federal Ministry of Transport and Digital Infraestruture Europea Union.2020.*

Porém, *RSUs* geralmente são caros para instalar e manter, assim, há uma compensação entre cobertura total, em termos de conectividade, por meio de *RSUs* e seu custo de implementação. Os métodos de implantação *RSU* comumente baseados em posicionamento fixo, usando regras geométricas, tem limitações graves, como áreas sem cobertura ou carga de rede desequilibrada (ou seja, alguns *RSUs* exibem um uso de rede extremamente alto, enquanto outros não). Para superar tais limitações, propomos um algoritmo genético *(AGs)* para implantação de unidade de beira de estrada *(GARSUD)*, que é baseado na evolução de um ramo de computação da inteligência artificial.

2.2. Tratamento das unidades, e tecnologias correspondente, ao *V2X*

Trataremos das unidades e tecnologias correspondente, ao *V2X*, começaremos com a *DSRC*, depois, as outras partes, no contexto, prático da comunicação *V2X*, tais como *ITS, RSU, GARSUD* e *OBU*.

2.2.1 Comunicação dedicada de curto alcance *[DSRC].*

No sistema de comunicação *V2I*, o veículo pode ser conectado com dispositivos *RSU* instalados a beira de estrada. A *Roadside Unit* compartilha informações de todos os veículos conectados a ele e obtém informações dinâmicas de veículos conectados a ele. A comunicação *V2I*, pode ser também cliente-servidor, a *RSU* como cliente lê os dados do sensor e os envia, então o servidor armazena e processa os dados. Existem muitos obstáculos em redes sem fio com condições externas na forma de sinal de propagação, interferência e áreas de cobertura de sinal. Nosso intuito neste estudo, e proporcionar um dispositivo *RSU* robusto e dinâmico para que possa oferecer um sistema de comunicação *V2I* com bom desempenho e, posteriormente, o dispositivo *RSU*, será usado para aplicações de estacionamentos inteligentes.

2.2.2. Sistema de transporte inteligente (STI)

O sistema de transporte inteligente *(*STI*)* é um sistema de transporte inteligente, que melhora a eficiência, conforto e segurança para o motorista. O tipo de comunicação constituído entre eles são comunicação entre veículos *(V2V),* comunicação entre veículos e infraestrutura *(V2I)* e comunicação entre veículos e quaisquer objetos *(V2X).*

Genericamente a comunicação *ITS*, vem sendo usada em vários países, essa tecnologia é utilizada em várias áreas, por exemplo, sistema de navegação avançado. Neste sistema o veículo é levado a obter o percurso rodoviário mais curto e ideal.

2.2.2.1. Mapas Digitais

Mapas digitais de formato geral são criados, baseados em sistema de Informação Geográfica *(SIG).*

2.2.2.2. Sistema de gerenciamento de tráfego avançado

Ele fornece informações em tempo real sobre as condições do tráfego para este sistema. Engarrafamentos, acidentes e obstáculos.

2.2.2.3. Sistema de Gestão de Acidentes

Este sistema pode ser usado para detectar eventos de emergência, como acidentes, deslizamentos de terra / outros desastres.

Os sensores fornecerão informações às partes relacionadas para mitigação de desastres.

2.2.2.4. Cobrança Eletrônica de Pedágio

Este sistema visa economizar tempo nos pagamentos dos usuários das estradas pedagiadas para que não haja longas filas nas saídas / entradas de estradas com pedágio.

2.2.2.5. Sistema de Informação de Ônibus Avançado

Este sistema fornece informações de chegada e partida de ônibus, bem como controla a rota do ônibus sistema.

2.2.2.6. Energia Recebida

No processo de comunicação, é importante analisar o fator de potência recebido pelos dispositivos no sistema de comunicação V2I. O nível de potência determina o desempenho do dispositivo. Com base na pesquisa, a quantidade de potência (P) na condição de Linha de Visão *(LoS)* pode ser calculado com base na Equação 1.

Equação 1 Fórmula da potência .

$$\mathrm{Pr} = \frac{PtGtGr}{L(rd)} \left[Dd\left(\frac{\lambda}{4\pi rd}\right) + Dr\left(\frac{\lambda}{4\pi r}\right)\eta e - j\left(r_{d-\,r_r}\right) + \phi\right]$$

Fonte: *Sundawa Batik, Journal Mantik 2019.*

Pr = potência recebida.

Pt= potência enviada.

Gt e Gr = ganho de envio e recebimento de antenas.

λ = comprimento de onda de propagação do sinal.

Rd = comprimento de onda direto.

rr = reflexão do comprimento de onda do sinal.

Φ = fase durante a onda que experiencia reflexos do solo.

η = coeficiente da superfície do solo.

Dd e Dr = coeficiente de diretividade da antena.

Eu(rd) = fator de absorção.

2.2.3. Unidade *Roadside (RSU)*

No sistema *V2X*, o veículo, deverá, ser capaz de se comunicar, da OBU, unidade on-board, com a infraestrutura, pela *RSU Roadside Unity,* a *RSU*, comunica com todos os veículos, conectados a ele, e tem de obter informações dos veículos de forma dinâmica e em tempo real. A rede de comunicação *V2I*, também pode ser cliente servidor, *RSU* como cliente lê e envia os dados do sensor, então o servidor armazena e processa os dados. *OBU* pode ser *Gadget, Transceptor* de rádio *GPS* e vários aplicativos. O ponto *OBU*, pode ser importante o suficiente para se comunicar com a *RSU*. A *RSU*, tem como base uma estação transceptora de rádio, que pode se comunicar de ambas as maneiras com a *OBU* no veículo. A *RSU,* estabelece comunicação e recebe os dados, tais como hora velocidade e localização do veículo, para que as informações sejam enviadas de volta para o dispositivo *OBU* do veículo.

O *Local Host*, será colocado na Área de infraestrutura na altura especificada, para o equipamento. *RSU*, pode ajudar o motorista a tomar decisões ao dirigir.

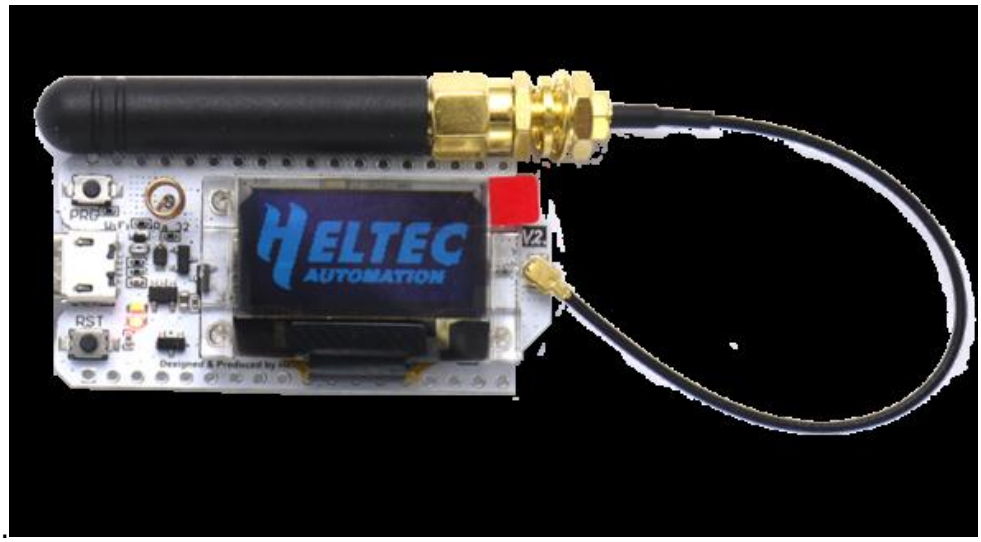
De acordo com a figura 5(RSU), temos uma foto de uma aplicação pratica de RSU.

Figura 5. Unidade *RSU (Fed.Min.)*.

Fonte: *Federal Ministry of Transport and Digital Infraestruture Europea Union 2020.*

A unidade RSU, que Identifica e comunica, os veículos, com a Network, no sistema V2I, pode ser implementado, com o auxílio de um CI LoRa, baseado no ESP-32, que consiste, no conjunto do ESP-32, mais uma rede Wi-Fi com Antena, de 1 Ghz, e uma circuito GPS, tudo alimentado por 12V.De acordo com a figura 6.

Figura 6. LoRa ESP32 Espressif.



Fonte: LORA ALLIANCE. LoRa. 2019.

2.2.3.1. Algoritmo genérico para o sistema de implantação de *RSU(GARSUD)*

Redes de veículos fazem uso de *Roadside Units (RSUs).* Para melhorar a capacidade de comunicação dos veículos para encaminhar mensagens de controle e/ ou fornecer acesso à internet a *OBU*, existe um problema de o sinal *Wi-Fi*, a propagação é barrada por edifícios e outros obstáculos, considerando, em especial o padrão IEEE 802.11p.

As unidades *RSU*, distribuídas geograficamente, tem limitações graves, tais como áreas sem cobertura ou carga de rede desequilibrada. Para contornar este problema foi proposto uma solução utilizando *IA*. Em um ramo, no que concerne à Algoritmos Genéticos (*AG*), em homenagem ao processo de evolução biológica e sua base, como conduzir uma população a evoluir, submetendo tal população à processos aleatórios semelhantes aos presentes na evolução biológica, na forma, como mutações e recombinações, formam a seleção de quais indivíduos são mais adaptados a sobreviver. Os indivíduos mais aptos, são selecionados e avançados para as gerações subsequentes, e os indivíduos menos aptos, são descartados. Os

Algoritmos Genéticos requerem o estabelecimento de relacionar soluções para um problema, denominado fenótipo e um conjunto de indivíduos de uma população natural. Isso é realizado codificando as informações de cada solução em uma palavra, geralmente binária, chamada cromossomo ou genótipo. Os símbolos que compõe tal palavra são os genes. As populações de possíveis soluções codificadas à medida que os genótipos evoluem através de iterações chamadas gerações, obtendo assim, para cada genótipo os fenótipos correspondentes, que são avaliados usando uma medida de adequação. Para gerar a geração subsequente, o seguinte não deve ser usado um conjunto de operadores genéticos, seleção de pais, cruzamento, mutação e substituição.

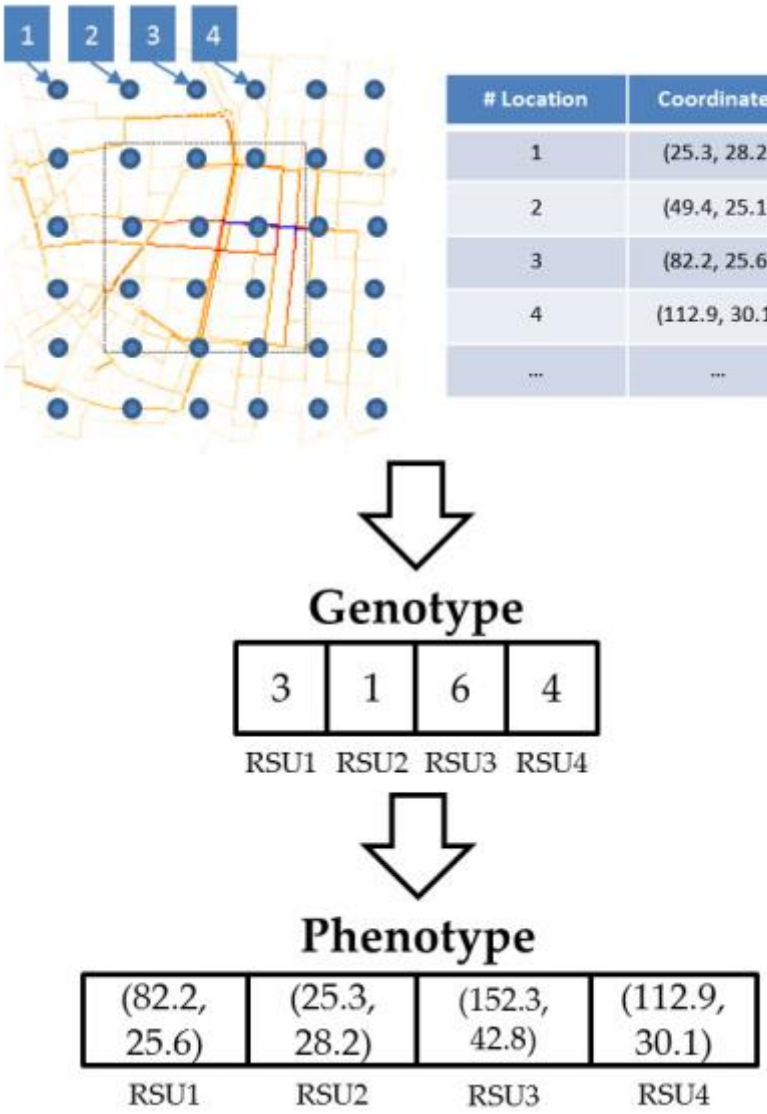
2.2.3.2. Representação de indivíduos na *GARSUD*

Cada solução plausível para o problema requer a codificação em um genótipo antes de aplicar os operadores genéticos. O fenótipo nada mais é do que as soluções possíveis para a localização das *RSUs* disponíveis. Um genótipo apropriado teria correspondência clara com um fenótipo específico. Certo de que os genes devem ser selecionados a partir de um alfabeto finito, o número de locais possíveis para as *RSUs* é limitado e eles podem ser configurados. *GARSUD* atribui um número único a cada localização possível para as *RSUs*, e o genótipo de solução contém uma sequência de números que representa o local selecionado para cada *RSU,* conforme figura 7.

Uma estratégia de medida é aplicada para evitar a atribuição de mais de uma RSU para cada local possível, melhorando a performance do algoritmo.

Representação de indivíduos no algoritmo genético para RSU, desdobramento GARSUD, cada possível localização para *RSUs*, aparece com círculos azuis no layout do mapa. (FOGUE Manuel).

Figura 7.GARSUD seleção.



Fonte: FOGUE Manuel, SAGUESA Julio, MARTINEZ Franscisco, BORGES Johann M. M. Aplyed Science. 2017.

2.2.3.3. Seleção dos pais

Nesta fase, seleciona os indivíduos mais aptos na população que existe naquele momento, para transmitir suas informações para a próxima geração. O *GARSUD*, faz seleção no grupo, no qual n indivíduos, aleatórios são comparados e aquele com maior valor de aptidão, torna-se um pai em potencial de novos indivíduos.

2.2.3.4. Recombinação ou *Crossover*

O *crossover*, une dois pais para gerar um terceiro novo individual. O *GARSUD* usa um operador de recombinação padrão em algoritmos genéticos, ou, melhor, gera um novo ponto crossover.

2.2.3.5. Mutação

É o efeito, que os genes podem mudar seu valor usando uma probabilidade constante.

2.2.3.6. Substituição

Também conhecida como seleção de sobreviventes, é feita de forma reposição geracional, onde os últimos indivíduos, são inteiramente repostos pela nova geração.

2.2.3.7. Fitness

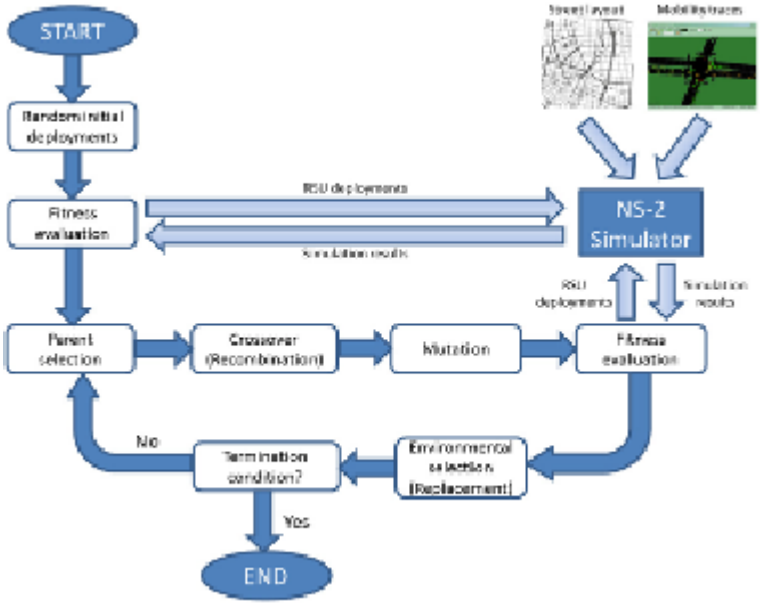
Atua como uma guia no processo de pesquisa. A função de *Fitness* que orienta o *GARSUD*, onde temos wnt (i) representa o tempo médio de aviso de notificação medida no cenário i e N é o número de cenários testados.

$$GARSUD\_Fitness = \frac{100}{\frac{\sum_{i=1}^{N} wnt(i)}{N} + 1.0}$$

Equação 2: Fórmula *Fitness* (Applsci)

O esquema de funcional do GARSUD, estudado, é mostrado na Figura 8.

Figura 8. Algoritmo *GARSUD*.



Fonte: FOGUE Manuel, SAGUESA Julio, MARTINEZ Franscisco, BORGES Johann M. M. Aplyed Science. 2017.

2.3. O que é LoRa?

A LoRa é uma tecnologia de modulação de rádio frequência para redes de distância longa (LPWANs) de baixa potência. Seu nome vem de uma similaridade aos links de dados de alcance extremamente longo, que se faz com esta tecnologia. Foi criado pela Semtech para padronizar LPWANs. O alcance da comunicação LoRa é de até 5 km em áreas urbanas e até 15 km ou mais em áreas rurais.

A LoRa tem caracteristica de requisito de energia ultrabaixa, que permitem dispositivos operador por bateria, que podem durar até 10 anos, implementado em técnologia estrela. É ideal para comunicação interna profunda, entre muitos dispositivos que tem requisitos de baixa energia e que coletam pequenas quantidades de dados.

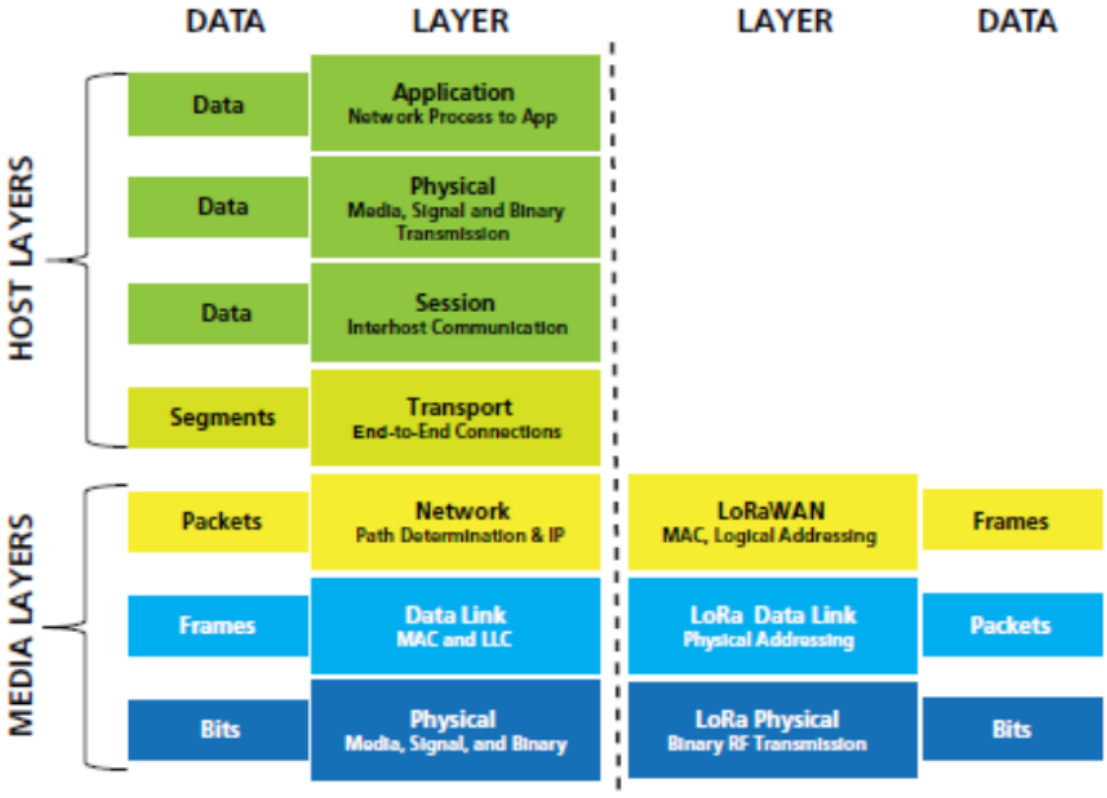
2.3.1. Modulação de rádio e LoRa

Temos uma técnica proprietária de modulação de espectro espalhado, derivado do chirp Spread Sprectum (Css) tecnologia existente LoRa oferece um compromisso entre a sensibilidade e taxa de dados enquanto opera em um canal de largura de banda fixa de 125khz ou 500khz (para canais de uplink) e 500 khz (para canais downlink).

A LoRa, usa fatores de espalhamento ortogonais. Isso permite que a rede preserve a vida útil das baterias de nós finais conectados, fazendo otimizações adaptativas de energia em nó individual, em níveis e taxas de dados. Por exemplo, um dispositivo final localizado próximo de um gateway, deve transmitir dados em um baixo fator de espalhamento, uma vez que pouco nível de link é necessário. No entanto, um dispositivo final localizado a vários kms de um gateway precisara transmitir com um fator de espalhamento maior. Esse fator de maior difusão, fornece maior ganho de processamento e maior sensibilidade de recepção, embora a taxa de dados necessariamente seja inferior.

LoRa é puramente uma implementação de camada física (PHY) ou “bits” conforme definido pela OSI de sete camadas. O modelo de rede, representado na Figura 9.

Figura 9 Camadas de comunicação LoRa, ESP32.



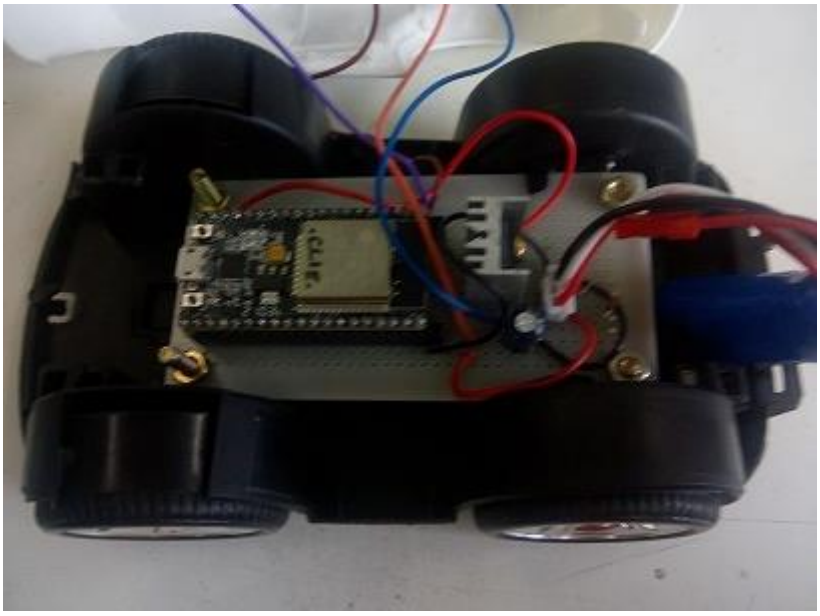
Fonte: LORA ALLIANCE. A Technical Overview. 2019.

2.4. OBU Unidade On-Board

A unidade On-Board, que torna o veículo identificado no sistema *V2X*, pode ser implementada utilizando o *ESP-32*, é um microcontrolador de 32 bits, 160 MHz, 2,4 GHz de *Wi-Fi* e *Bluetooth,* que pode ser alimentado com 12V, que tem um circuito redutor de 3,3V, para funcionamento do microcontrolador, sua característica é DIP, sendo que facilita configuração de placa , que pode ser programada em C, mas admite programação similar ao do Arduino, tem algumas bibliotecas disponíveis, o custo do CI não é alto, e está disponível no mercado nacional.

Para efeito prático devemos adotar um módulo ESP32 nodeMCU, como na figura 10.

Figura 10. OBU prática.



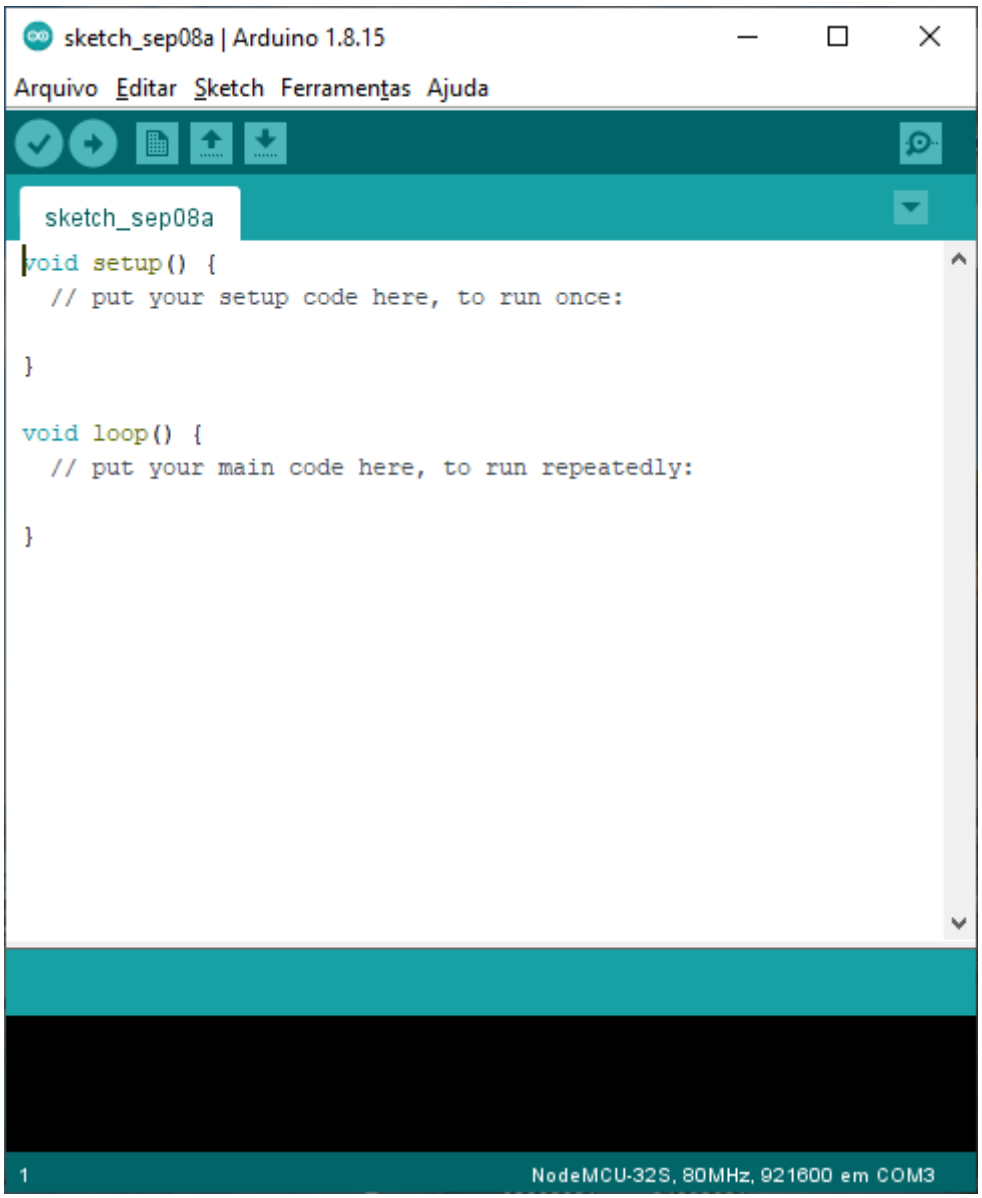
Fonte: Datasheet ESP32 Espressif OBU 2022

2.5. Instalando o IDE-Arduino, para programação

Para o desenvolvimento do projeto será necessário instalar a IDE Arduino. Na Figura 11, é demonstrado a IDE Arduino. Todo o processo de instalação da IDE está descrito no Anexo 1.

.

Figura 11,



Fonte: BERTOLETI Pedro. 2019.

Inicialmente foi desenvolvido um programa teste, para a leitura dos sinais adquiridos por WI-FI.

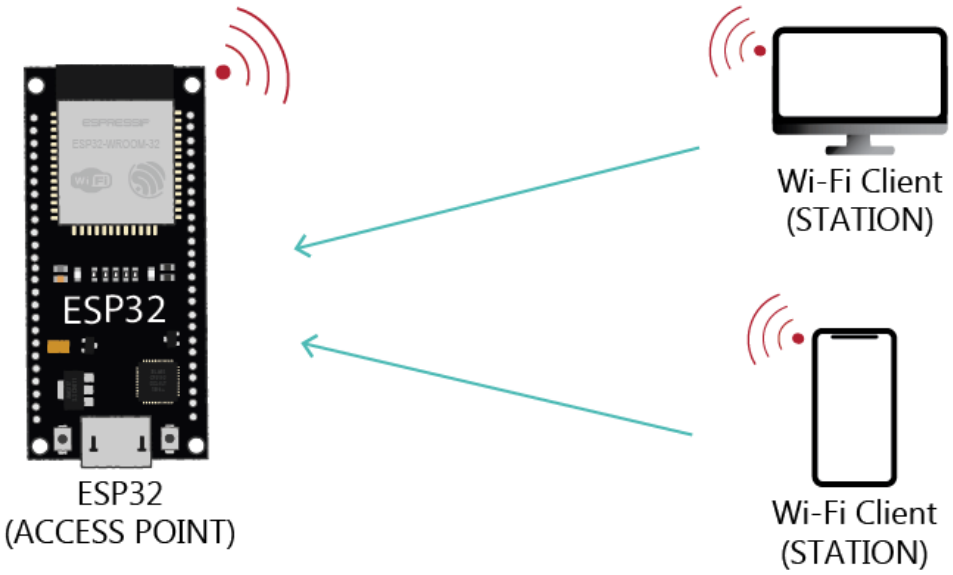
2.5.1. Modo ESP32 Wi-Fi:

A placa ESP32 pode atuar como estação Wi-Fi, ponto de acesso ou ambos. Para definir o modo Wi-Fi, foi utilizando a função Wi-Fi.mode () e definir o modo desejado como argumento:

- Wi-Fi.mode(WI-FI_STA); modo de estação, o ESP32 se conecta a um ponto de acesso.
- Wi-Fi.mode(WI-FI_AP); modo de ponto de acesso: as estações podem se conectar ao ESP32.
- Wi-Fi.mode(WI-FI_STA_AP); ponto de acesso e uma estação conectada a outro ponto de acesso.

A placa OBU será configurada como: Wi-Fi.mode(WI-FI_STA) modo de ponto de estação. A placa RSU será configurada como: Wi-Fi.mode(WI-FI_AP) modo de ponto de acesso; as estações podem se conectar ao ESP32. Quando definimos nossa ESP32 como ponto de acesso, podemos conectar qualquer dispositivo com recursos de WI-FI, sem se conectar ao roteador, nesse caso você cria sua própria rede WI-FI e dispositivos próximos, podem se conectar com ela, como um computador ou no nosso caso outra, ESP32, sem a necessidade de um roteador para controlá-lo. Como na Figura 12.

Figura 12 modo de ponto de acesso ESP32 Wi-Fi.



Fonte: Random Nerd Tuturails 2019.

2.5.2. Configuração do Wi-Fi ESP32

Definindo o modo Wi-Fi como ponto de acesso: Wi-Fi.mode (WI-FI_AP), em seguida, foi empregado a função softAP ():

- ssid: nome do ponto de acesso – máximo de 63 caracteres;
- Senha: mínimo de 8 caracteres; definido como NULO se você desejar o ponto de acesso aberto:
- Canal; Número do canal Wi-Fi (1-13)
- ssid hidden;(0 = transmitir SSID, 1= ocultar SSID)
- Max_connection: máximo de clientes simultâneos (1-4) [15].

2.5.3. Comunicação ESP-NOW

Para o projeto será utilizado o protocolo *ESP-NOW*, que é de comunicação sem conexão desenvolvida pela Espressif que apresenta transmissão em pacotes curtos. Este protocolo permite que vários dispositivos ESP32 possam trocar informações de forma simplificada[31]

ESP-NOW é um protocolo desenvolvido pela Espressif que vários dispositivos se comunicam entre si sem usar o Wi-Fi. O protocolo é semelhante à conectividade sem fio de 2,4 GHz de baixa potência, o emparelhamento entre dispositivos é necessário antes de sua comunicação. Após o emparelhamento a conexão é segura e ponto a ponto, sem a necessidade de *handshaking*. Com isso, uma vez realizado o processo de emparelhamento de um dispositivo com o outro, a conexão é persistente.

2.5.3.1. Recursos do ESP_NOW

- Não suporta Broadcast
- Pares criptografados limitados
- No máximo 10 pares criptografados, são suportados no modo Station; 6 no máximo no modo SoftAP ou SoftAP + Station, seu número deve ser menor que 20.
- A carga útil é limitada a 250 bytes.

2.5.3.2. Como funciona o protocolo ESP-NOW

Ele funciona como um protocolo de comunicação Wi-Fi sem conexão definido pela Espressif. No ESP-NOW, os dados do aplicativo são encapsulados em um quadro de ação específico do fornecedor e depois transmitidos de um dispositivo Wi-Fi para outro sem conexão. A CTR com o protocolo CBC-MAC(CCMP) é usado para proteger o quadro de ação por segurança. Precisara ser utilizada a biblioteca esp_now.h, para executar o programa.

3. Programação

Aqui vão ser demostrados, os programas de cliente e servidor, demonstrativos, para provar comunicação entre os dois, a ser aplicado, na comunicação V2I, junto com seus fluxogramas.

O cliente no caso, recebe os valores pré-configurados em uma matriz de dados que será transmitido, em pares de comunicação de protocolo ESP_NOW.

O programa cliente, faz uso de duas bibliotecas principais, Esp_now.h e Wi-Fi.h, inicialmente serão definidas as variáveis, que serão tabuladas de maneira simples, dentro da função struct, sera criada uma mensagem struct, Sera utilizada o comando myData, para enviar os dados, que serão enviados, e depois sera criada uma interface de pares, e posteriormente enviados, depois terá um retorno a chamada, pela void OnDataSent dentro do protocolo ESP_NOW,

No nosso caso de plataforma IDE-Arduino, é comum dividirmos em duas partes de execução, void setup e void loop.

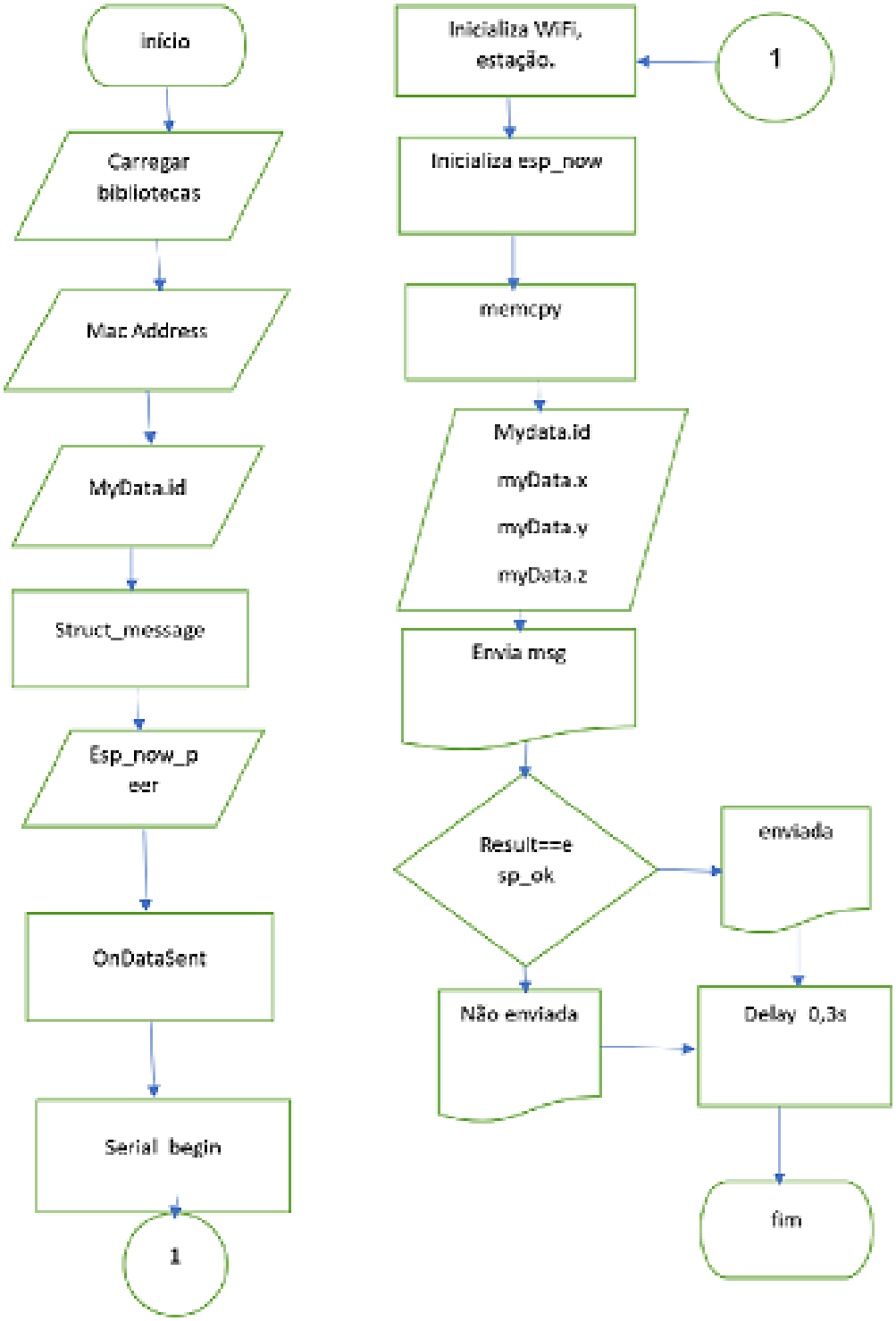
No void setup, ligaremos um led, para indicar que os dados foram enviados, iniciaremos o monitor serial, com sua velocidade de comunicação e definiremos que tipo de configuração de comunicação usaremos o ESP32, no nosso caso sera STA, station(estação). Iniciaremos o ESP_NOW, obteremos o status dos pares transmitidos, esp_now_register_send_cb (OnDataSent). Registraremos os pares, definiremos o canal de transmissão e se é ou não criptografada,

No void loop, daremos valores as variáveis, com extensão myData, ex; myData.id = 1, que no nosso caso, indicara o número do veículo. E finalmente envia a mensagem via ESP_NOW, se for enviada acendera o led builtin. para finalizar é dado um delay de 0,3 segundos, apenas para fins de contagem de tempo da rotina.

Para exemplificar e esquematizar o programa do cliente, que é o veículo 1 e 2, primeiro demonstramos com o fluxograma do programa (Figura 13) que serve genericamente, para os dois programas dos veículos.

3.1. Fluxograma do cliente(Automóvel)

Figura 13: Fluxograma do cliente



Fonte: Própria.

3.2. Programação do ESP32 Mestre. Veículo 1

No nosso programa mestre, esp32 NODEMCU, indicados no apêndice 1. Daremos início ao programa da OBU 1, que fara o papel de STA (Station, estação). Que enviara o sinal para a RSU, que ao mesmo tempo, recebera de outras OBUS.

Este programa sera carregado no veículo 1, que sera um carrinho, com bateria lipo, um redutor de tensão para 5,0V, chaves de liga/desliga e Enable, chaves de um polo duas posições. Este enviara um sinal, de programação via ESP_NOW, que enviara pares, que conforme a distância, comutara a Roadside Unit,

3.2.1. Programação do ESP32 Mestre. Veículo 2

No nosso programa mestre, esp32 NODEMCU, indicados no apêndice 2. Daremos início ao programa da OBU 2, que fara o papel de STA (Station, estação). Que enviara o sinal para a RSU, que ao mesmo tempo, recebera de outras OBUS.

Este programa sera carregado no veículo 2, que sera um carrinho, com bateria lipo, um redutor de tensão para 5,0v, e chaves de liga e desliga e Enable, chaves de um polo duas posições. Este enviara um sinal, de programação via ESP_NOW, que enviara pares, que conforme a distância, comutara a Roadside Unit.

3.2.2. Versões anteriores, do programa Mestre

Fica aqui demostradas versões anteriores de desenvolvimento dos programas ESP32 Mestre, veículos (13/11/2021). Primeira versão, apêndice 3, onde foram desenvolvidas as bases de programação para funcionamento da comunicação, entre mestre servidor. Na sequência, foi estipulado uma versão utilizando peer info, instrução de comunicação por pares, de 25/02/2022, colocado no apêndice 4. Logo depois foi criado um programa que enviaria os dados, um por um, via ESP_NOW, que não funcionou adequadamente, em 21/04/2022, descrito no apêndice 5. E por último a versão, que envia um dado, pelo ESP_NOW de 03/05/2022, conforme apêndice 6.

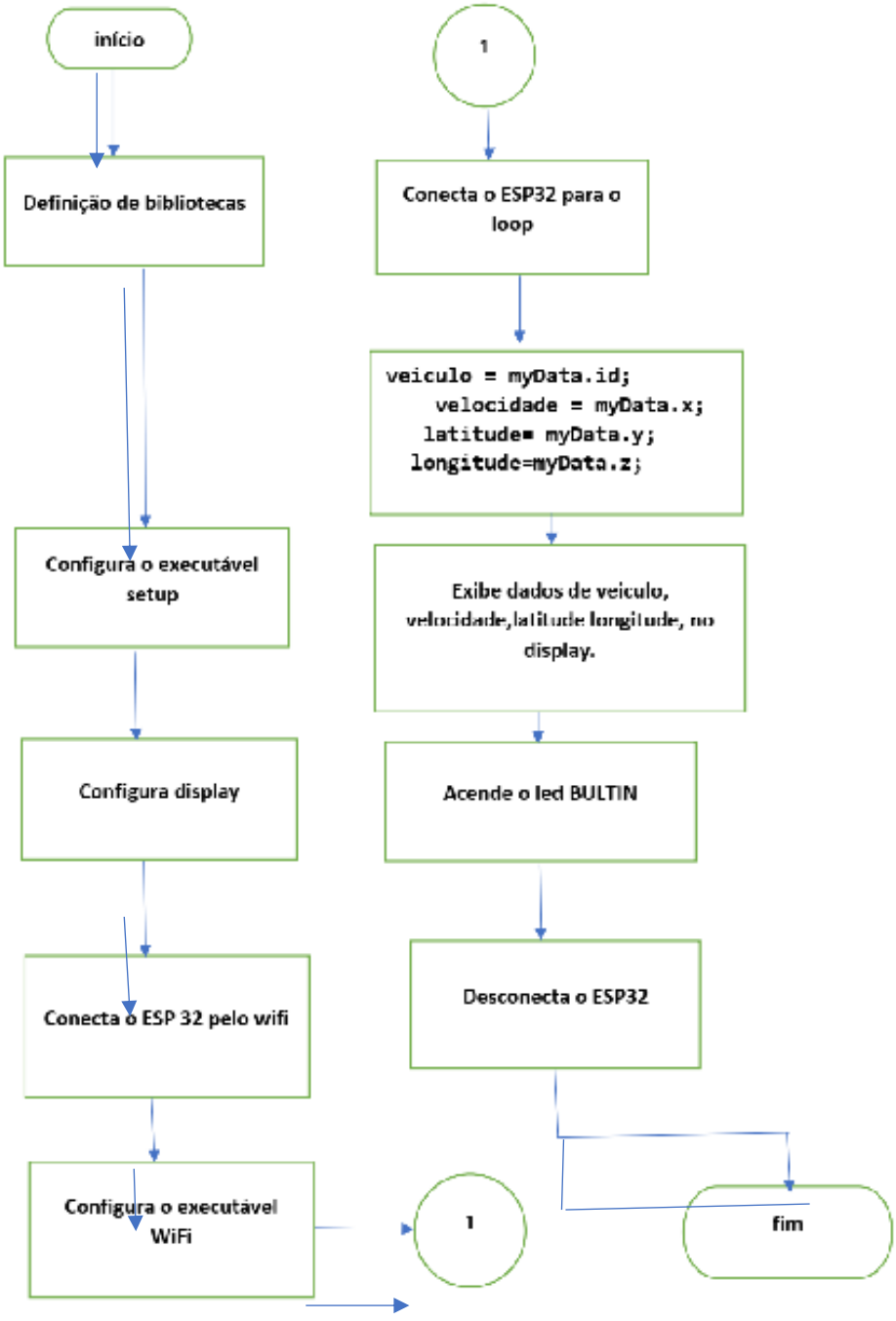
3.3. PROGRAMAÇÃO DA ROADSIDE UNIT, COMO SERVIDOR STA

Para demonstramos a funcionabilidade do programa da RSU, que sera genericamente chamada de servidor, por sua função de recebimento de sinais e redistribuição. Vamos esquematizar através de fluxograma e posteriormente o

programa de funcionamento, que deverá ser gravado, via cabo USB mini, através da plataforma IDE-Arduíno, que deverá ser configurada de acordo com o Anexo1 e 2.

3.3.1. Fluxograma do Servidor RSU

Figura 14: Fluxograma da RSU.



Fonte: Própria.

3.3.2. Programas do ESP32 SERVIDOR_ROADSIDE_UNIT

No nosso programa Servidor, *Roadside* Unit, que recebera os dados das OBUS, com base no esp32 LoRa Wi-Fi, conforme Apêndice 7.

3.3.3. Versões Anteriores dos Programas RSU, servidores

A primeira versão de programa RSU, foi de 13/11/2021, cujo programa e mostrado no apêndice 8. Programa este que recebera os dados enviados pelas OBUS e mostrara no display, foi utilizada biblioteca Heltec, para o display. Corresponde ao Programa do Apêndice 3.

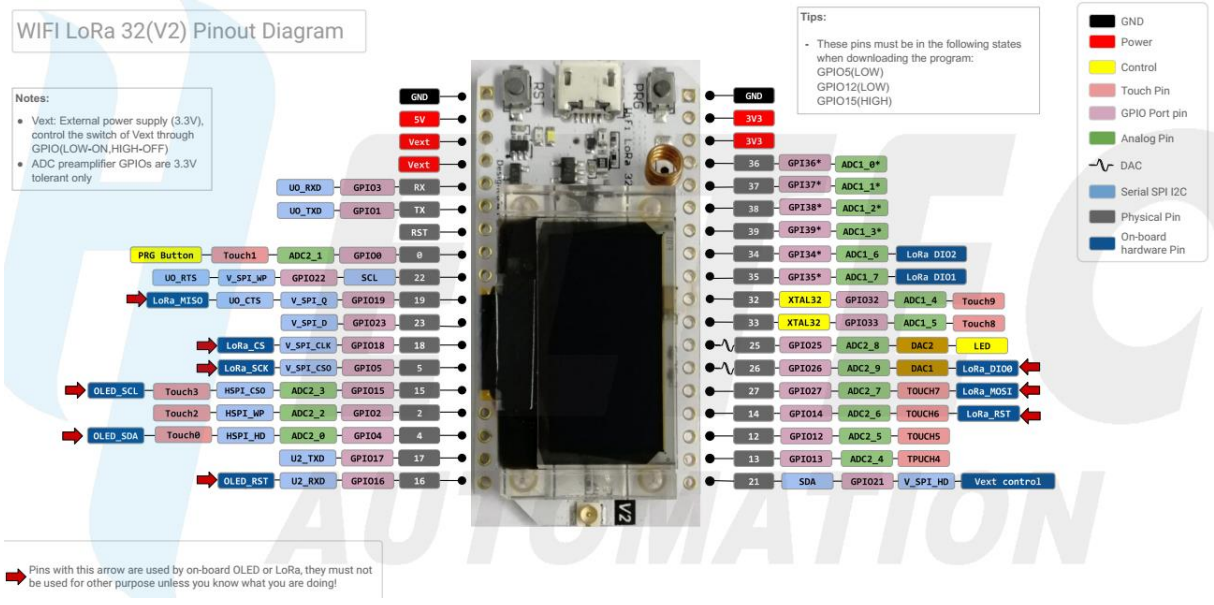
Uma nova versão do programa foi construída em 10/03/22, cujo programa é mostrado no apêndice 9. Que funciona em conjunto com o programa no Apêndice 4.

Em 22/04/2022, foi criada uma versão do programa (Apêndice 10), que corresponde ao programa mestre do apêndice 5. Na sequência temos a versão RSU_0505, que corresponde ao apêndice 11, que corresponderá ao par do apêndice 6.

A placa da RSU vai ser utilizada como Base eletrônica para recepção e comunicação de sinais. Como na Figura 15.

3.4 Pinagem LoRa ESP32 (pin-out)

Figura 15 descrição física de pinos do ESP-32 LoRa.

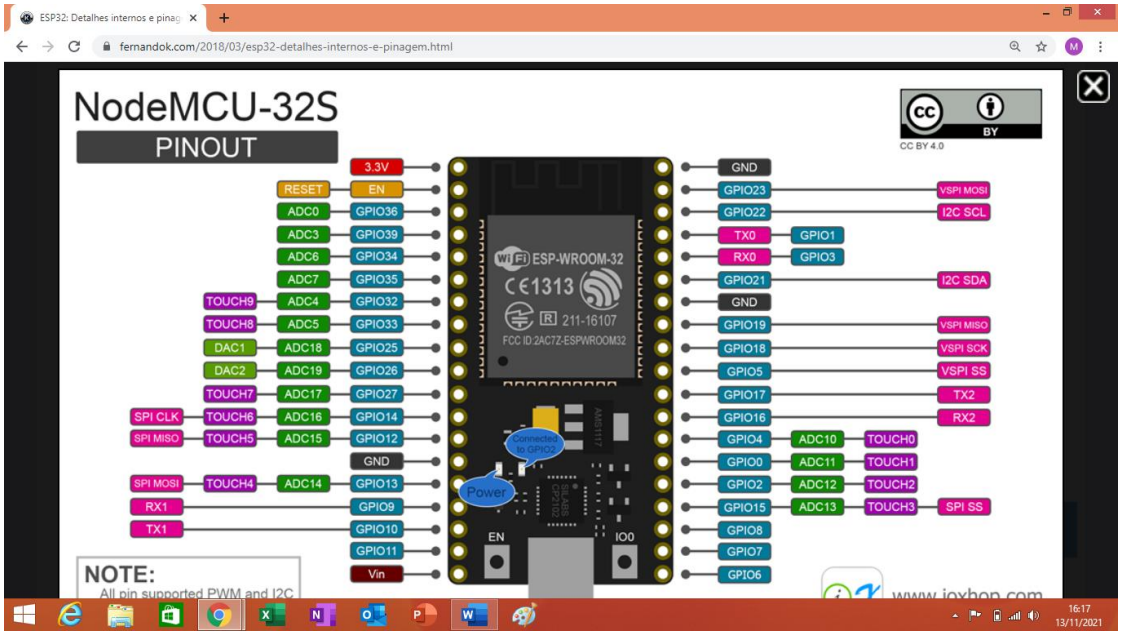


Fonte: FLOP Felipe. 2021.

Como Hardware da OBU, iremos utilizar o nodeMCU, como na Figura 16.

3.5. Pinagem (pin-out ESP32 NODEMCU) cliente.

. Figura 16. Descrição física dos pinos do ESP32 nodeMCU.

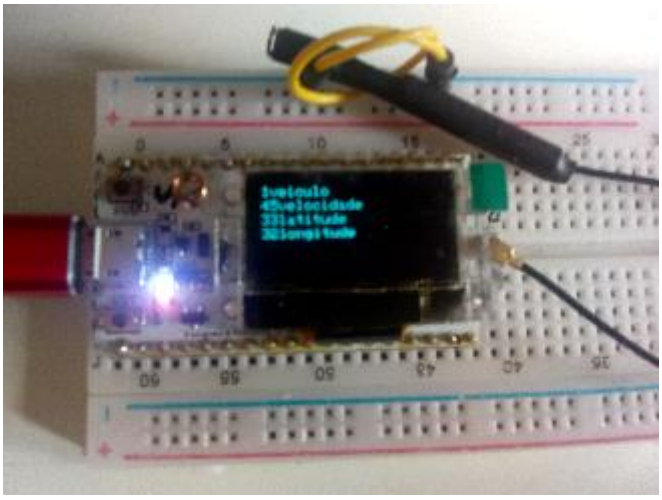


Fonte: FLOP Felipe. 2021.

4. TESTE FUNCIONAL

Para testar a funcionabilidade de envio de mensagem da OBU para a RSU, podemos monitorar a mensagem, escrevendo no ESP32 LoRa, a mensagem recebida pela OBU (Figura 17).

Figura 17 ESP32, LoRa, mostrando a mensagem de leitura de dados recebidos.



Fonte: própria.

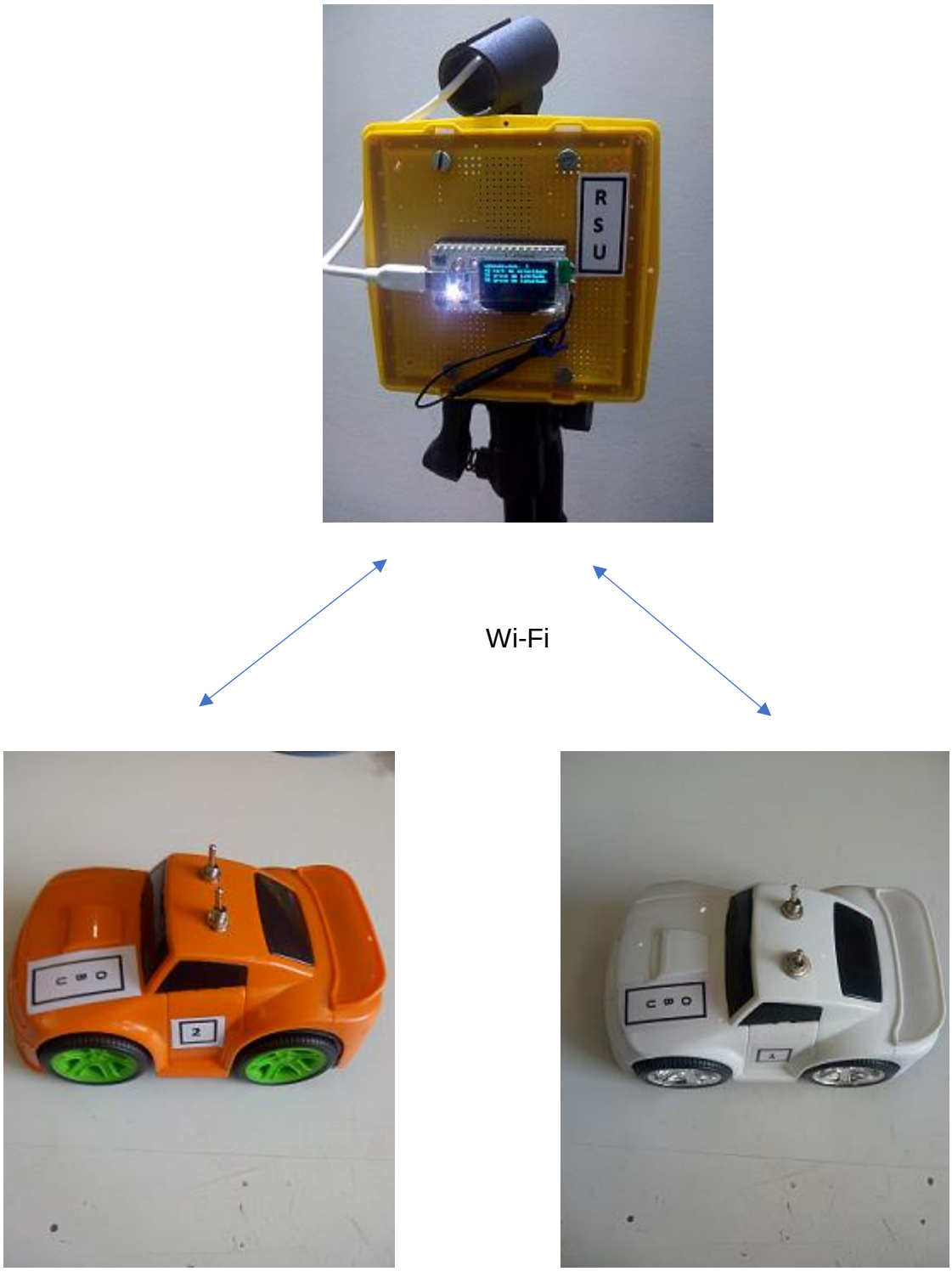
4.1. Experiência Prática

Como procedimento, utilizaremos, dois ESP32 e um nodeMCU, instalados em carrinhos de plástico, cuja função será a de emitir sinal, de Wi-Fi, foram feitos em base de placa perfurada universal, com um soquete soldado para instalação do ESP32, sua alimentação que terá de ser de 5V, que será alimentada por uma bateria lipo de 2 células de 7,7V. conforme figura 10.

Como se trata de um pequeno sistema de comunicação, a RSU, recebera e enviara dados às OBUS, que serão os veículos, que no programa são tratados como mestre, e a RSU como servidor. O sistema funciona por ondas emitidas via sinal de rádio ou Wi-Fi, e formam um par de entrelaçamento, que correspondera a comunicação, os veículos no caso, são maquetes, carrinhos de plástico, alimentados por bateria lipo, e a RSU neste caso é alimentada por um carregador, e plugado na rede elétrica, tudo para efeitos práticos.

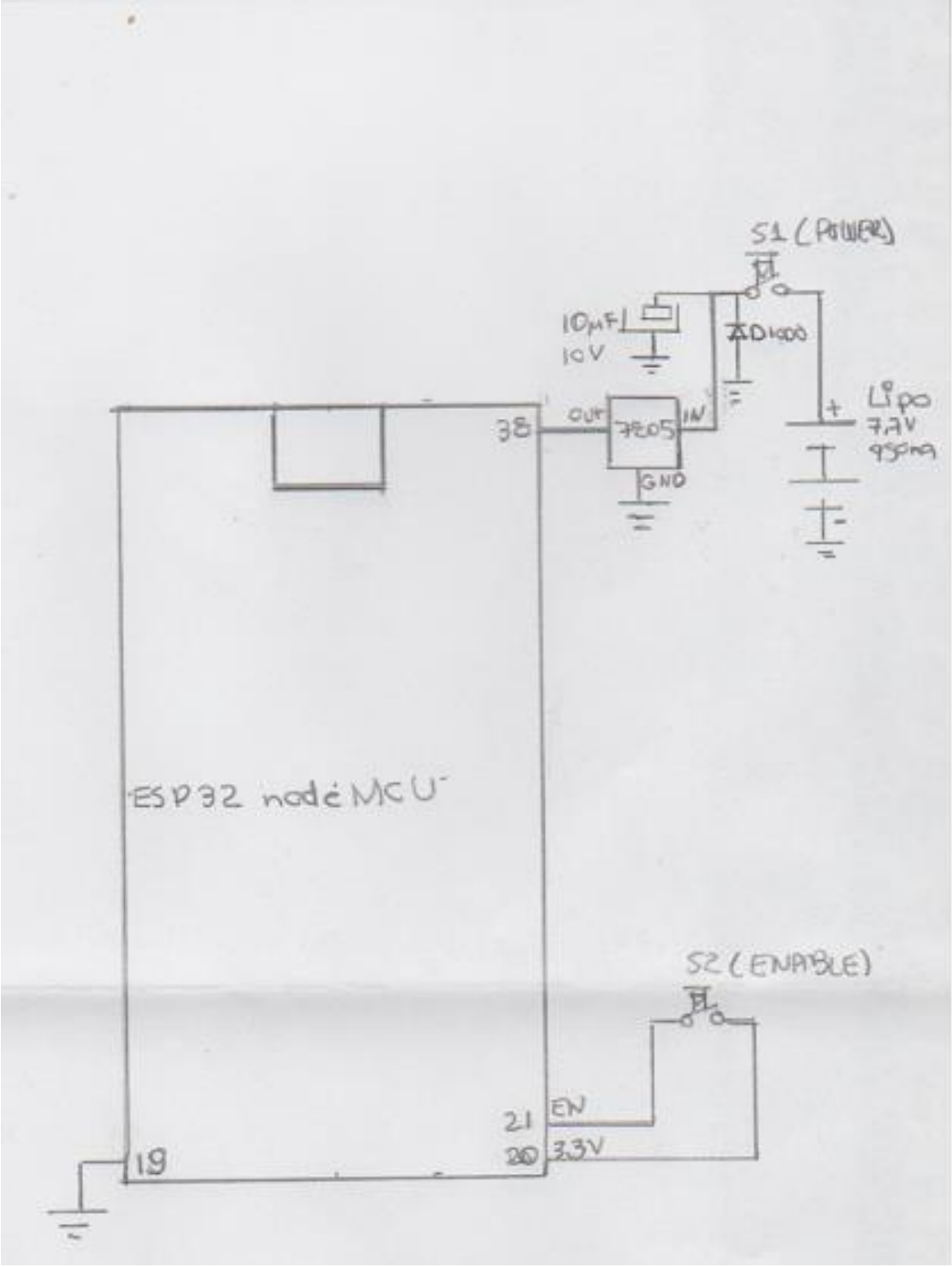
Foi feito o teste de distância de conectividade, que obteve 30 metros com obstáculo, e 70 metros sem barreiras. Foi feita o reset, a entrada da OBU 2 e depois a entrada da OBU 1 a distância, recebido o sinal na RSU, conforme figura 18.

Figura 18



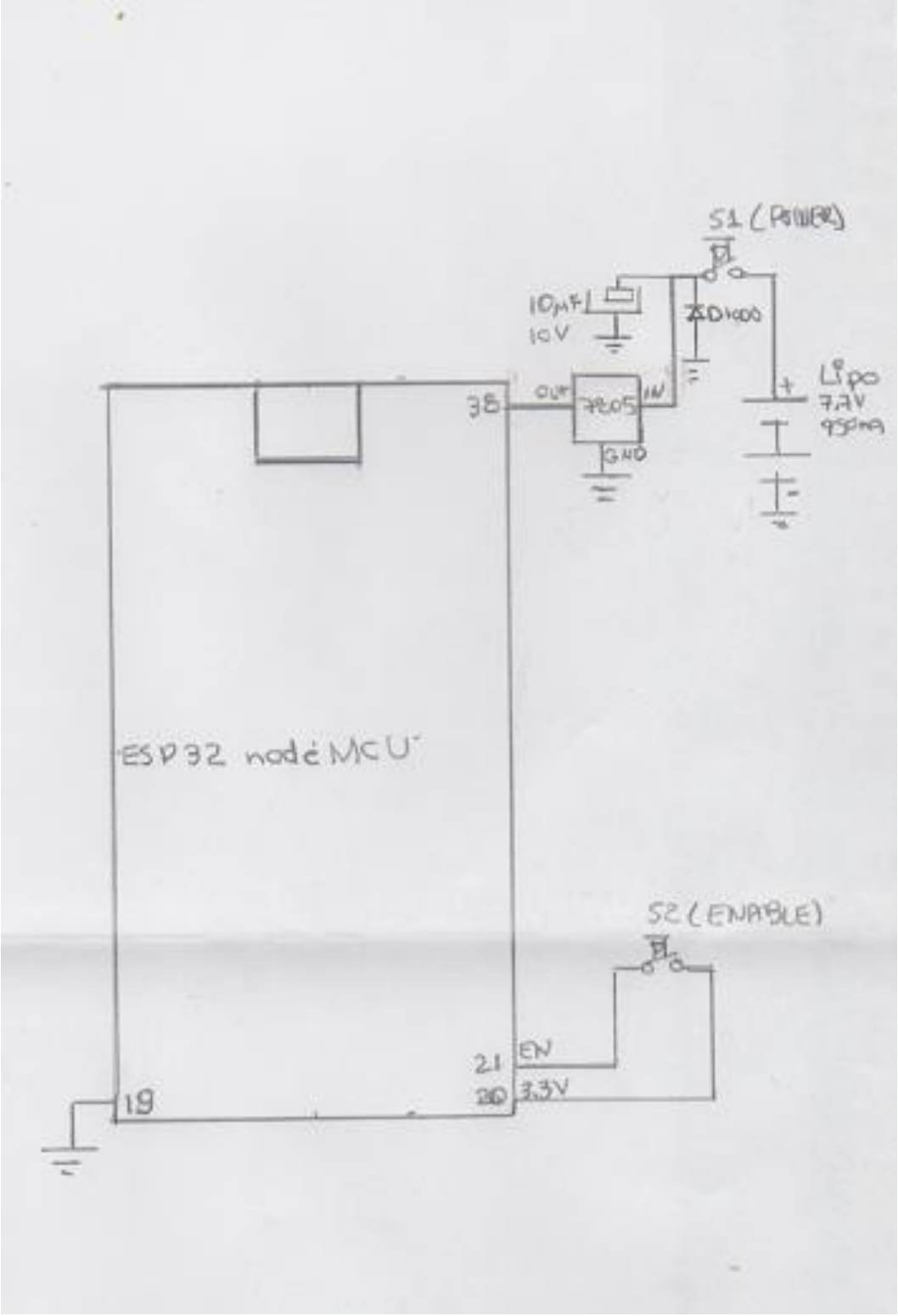
Fonte: Fotografia tirada em 30/06/2022

Figura 19: Circuito elétrico da OBU 1:



Fonte: Desenho do circuito, baseado na pinagem do CI ESP32.

Figura 20: Circuito elétrico da OBU 2



Fonte: Desenho do circuito, baseado na pinagem do CI ESP32

5.CONCLUSAO:

Devido ao grande número de dispositivos conectados nas redes 4G e 5G, e com a necessidade de tornar os carros conectados, entre os meios , seja *V2I* (Veiculo para infraestrutura), *V2V* (Veiculo para Veículo) , *V2N* (Veiculo para Network) e C-V2P (Veiculo para pedestre) via celular, existe a necessidade de instalar , unidades *Roadsides*, ao longo do percurso das vias, que no intuito deste estudo, poderá ser implementado a partir, de um dispositivo *LoRa*, que tem um *ESP 32*, mais Wi-Fi mais Gps. , conta com antena, e pode ser alimentado por uma fonte de 12V. e pode ser implementado, à beira de estrada, com dispositivos, que disponham de carregamento por fotocélula, e bateria, para o período noturno.

6. PROPOSTAS FUTURAS

A possibilidade de usar as entradas de rede CAN do ESP32, para fazer leituras no automóvel, e usá-los como dados a serem enviados à RSU, quanto à RSU, poderíamos pensar em um meio de usá-la, em estacionamentos inteligentes.

REFERENCIAS

[1] AUTOMATIZATION Heltec, novembro de 2021, disponível em: https://heltec.org/project/Wi-Fi-lora-32/. Acesso em: 11 maio.2022.

[2] BERTOLETI Pedro. Projetos com Esp32 e LoRa. Cap. 3. São Paulo: Instituto NCB:2019.São Paulo.

[3] BERTOLETI Pedro. Enviando e recebendo dados via MQTT com o ESP32, 15 de novembro 2019. Disponível em : https://www.newtoncbraga.com.br/index.php/microcontroladores/143-tecnologia/17070-enviando-e-recebendo-dados-via-mqtt-com-o-esp32-mic373.html#:~:text=Para%20enviar%20dados%20ao%20ESP32,conforme%20mostra%20a%20figura%209. Acesso em: 05 março. 2022.

[4] BERTOLETI Pedro. Módulo ESP32 Heltec, 15 de março 2019. Disponível em: https://www.newtoncbraga.com.br/index.php/microcontrolador/143-tecnologia/16326-moduloesp32-heltech-mec218. Acesso em: 22 abril. 2022.

[5] Camada Física PHY e OSI, disponível em: https://stringfixer.com/pt/Physical_layer. Acesso em: 04 julho. 2022.

[6] Cisco Annual Internet Report (2018–2023) White Paper. 2016. Disponível em: *https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.* Acesso em: 25 maio. 2021.

[7] Datasheet ESP32 Wroom. Disponível em: https://www.alldatasheet.com/view.jsp?Searchword=Esp-wroom-32%20datasheet&gclid=Cj0KCQjwn4qWBhCvARIsAFNAMigBgTX5W-7lM1kI6oQHCY0QEc2Dk7-BM-o-qLl_2q7TOtX5FrK0tu4aAuYAEALw_wcB. Acesso em: 04 julho. 2022.

[8] *Federal Ministry of Transport and Digital Infraestruture Europea Union. https://convex-project.de/onewebmedia/D4.1_RoadsIDE-ITS-Station-Specification_rev1.pdf* acesso em: 10 outubro. 2021.

[9] FLOP Felipe, 29/05/2021; https://www.filipeflop.com/produto/modulo-Wi-Fi-esp32-bluetooth/. Acesso em: 18 maio. 2022.

[10] FLOP Felipe, 29/05/2021 https://www.filipeflop.com/?s=LoRa&post_type=product. Acesso em: 18 maio. 2022.

[11] FMV learning. O que é ESP_NOW, Disponível em: https://www.fvml.com.br/2020/01/o-que-e-esp-now-e-como-funciona-codigo.html?showComment=1602025572334#google_vignette. Acesso em 05 julho.2022.

[12] FOGUE Manuel, SAGUESA Julio, MARTINEZ Franscisco, BORGES Johann M. M. Aplyed Science; 09/01/2018 https://www.mdpi.com/2076-3417/8/1/86. Acesso em: 10 outubro.2021.

[13] GOZALVEZ, J.; SEPULCRE, M.; BAUZA, R. IEEE 802.11p Vehicle to Infrastructure Communications in Urban Environments. IEEE Communications Magazine, [s. l.], n. May, p.176–183,2012. https://www.researchgate.net/publication/254058759_IEEE_80211p_Vehicle_to_Infrastructure_Communications_in_Urban_Environments. Acesso em: 10 outubro. 2021.

[14] IEEE 802. 11p. Disponível em: https://lume.ufrgs.br/handle/10183/78504. Acesso em: 04 julho. 2022.

[15] KOYANAGI Fernando, Arduino IDE com dual core, controle remoto,20 de novembro de 2018, Disponível em: https://www.fernandok.com/2018/01/esp32-controle-remoto-usando-sockets.html . Acesso em: 18 maio.2022.

[16] LORA ALLIANCE. LoRa® and LoRaWAN®:

A Technical Overview, December 2019. Disponível em: < https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf >. Acesso em: 24 maio. 2021.

[17] MARQUES Jemerson. FMV, o que é esp_now, exemplo e aplicações. Disponível em: https://www.fvml.com.br/2020/01/o-que-e-esp-now-e-como-funciona-codigo.html?showComment=1602025572334. Acesso em: 06 maio.2022.

[18] MELO Pablo: fotos comunicação V2I. Disponível em: https://www.embarcados.com.br/conhecendo-o-v2x/. Acesso em: 10 outubro. 2021

[19] MELO Pablo: Introdução ao LPWAN. Disponível em: https://www.embarcados.com.br/introducao-ao-lpwan/. Acesso em: 04 julho. 2022.

[20] NETO, Neri. V2x o mais novo conceito em Tecnologia para carros conectados e inteligentes, 2019. Disponível em: https://mundoconectado.com.br/artigos/v/8243/v2x-o-mais-novo-conceito-de-tecnologia-para-carros-conectados-e-inteligentes

[21] OMAR, Hassan Aboubakr; LU, Ning; ZHUANG, Weihua. Wireless access technologies for vehicular network safety applications. IEEE Network, [s. l.], v. 30, n. 4,p.22–26,2016.Disponivel,em: http://bbcr.uwaterloo.ca/~wzhuang/papers/VeMAC_magazine.pdf. Acesso em: 10 outubro.2021.

[22] Protocolo ESP_NOW, Espressif. Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html. Acesso em: 04 julho. 2022.

[23] ROSIN Zhongsham ITS Tecnolgy Co.Ltd. . Disponível em: https://sc01.alicdn.com/kf/HTB1ZfpWIXXXXXbnXFXXq6xXFXXXT/205734841/HTB1ZfpWIXXXXXbnXFXXq6xXFXXXT.jpg. Acesso em: 18 junho. 2022.

[24] *Sundawa Batik, Journal Mantik, vol.3 nº 3 novembro, Maranjemam Tecnology Informatika dom kumunikas ☹Mantik*) Disponível em: https://iocscience.org/ejournal/index.php/mantik/article/view/519. Acesso em: 10 outubro. 2021.

[25] SANTOS Rui, Blog Randon Nerd Tutoriais, 2013, Disponível em: https://randomnerdtutorials.com/esp32-useful-Wi-Fi-functions-arduino/ . Acesso em: 25 outubro. 2021.

[26] SANTOS Rui, Blog Randon Nerd Tutoriais, 2013, disponível em: https://randomnerdtutorials.com/esp32-useful-Wi-Fi-functions-arduino/#1. Acesso em 25 outubro. 2021.

[27] SANTOS Rui, Blog Randon Nerd Tutoriais, 2013, disponível em:

https://randomnerdtutorials.com/esp-now-many-to-one-esp32/. Acesso em 13 de outubro de 2022.

[28] SANTOS Rui, Blog Randon Nerd Tutoriais, 2013, disponível em:

https://randomnerdtutorials.com/esp-now-esp32-arduino-IDE/. Acesso em 17 de abril de 2022.

[29] TEIXEIRA Gustavo, ESP32 Comunicação Wi-Fi,2019, disponível em:

https://www.usinainfo.com.br/blog/esp32-Wi-Fi-comunicacao-com-a-internet/. Acesso em 01 de dezembro de 2021.

[30] TEIXEIRA Gustavo, ESP32 LoRa Wi-Fi SX1278,2019, disponível em: https://www.usinainfo.com.br/blog/esp32-lora-Wi-Fi-sx1278/, Acesso em 30 novembro. 2021.

[31] ESP-NOW User Guide, v1.0, 2016.07.20, disponível em: https://www.espressif.com/en/products/software/esp-now/resources

APÊNDICE 1: Programação do ESP32 Mestre. Veículo 1.

No nosso programa mestre, esp32 NODEMCU, começamos com a inclusão das bibliotecas necessárias ao funcionamento de HW e SW,

```
#include <esp_now.h>
#include <Wi-Fi.h>
```

Deverá ser inserido o endereço /Mac Address do receptor, como na linha a seguir.

```
//substitua pelo MAC Address do receptor.

uint8_t broadcastAddress [] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
//Exemplo de estrutura para enviar dados
//Deve corresponder à estrutura do receptor.
```

Depois, devemos criar uma estrutura que contenha os dados a enviar. Será chamada de struct_message, estrutura de mensagem, que terá a variável id. (mais as variáveis x, y, z).

```
typedef struct struct_message {

 int id; // must be unique for each sender board
int x;
  int y;
  int z;

 } struct_message;
//Cria uma mensagem struct, chamada myData.id.
struct_message myData;
```

crie uma variável do tipo esp_now_peer_info_t para armazenar informações sobre o par.

```
//Cria interface de pares.

esp_now_peer_info_t peerInfo;
//retorna a chamada quando os dados são enviados.
```

. Na sequência, defina OnDataSent () função. Esta deverá ser uma função de retorno de chamada que será executada quando a mensagem for enviada. Depois, esta função imprime se a mensagem foi entregue com sucesso ou não

```
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS? "Delivery Success" :
"Delivery Fail");
}


void setup() {
  //Inicia o monitor serial

  Serial.begin(115200);
```

```
  //Define dispositivo como uma estação Wi-Fi

  Wi-Fi.mode(Wi-Fi_STA);
  //Inicializa ESP_NOW.

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //Assim que o esp_now for iniciado com sucesso vamos nos registrar para enviar
CB para.
  //Obter o status do pacote transmitido.
  Após inicializar com sucesso o ESP_NOW, registre a função call-back que será
chamada quando uma mensagem for enviada. Neste caso, cadastre-se OnDataSent
() função já criada anteriormente.
  esp_now_register_send_cb(OnDataSent);
  //Registra par
 Para enviar dados para outra placa (o receptor). você precisa emparelhá-lo como
um par. As linhas a seguir registram e adicionam um novo par
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  //Adicionar par.

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
}

void loop() {
ciclo()
No ciclo(), enviaremos uma mensagem via ESP_NOW a cada 10 segundos, atribua
um valor a cada variável.
  //Define os valores para enviar.

  //strcpy (myData.id, "THIS IS A CHAR");

      myData.id = 1;
      myData.x=45;
      myData.y=33;
      myData.z=32;

  //Envia mensagens via esp_now
Por fim envie uma mensagem via ESP_NOW.
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
```

```
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  delay(300);
}
```

APÊNDICE 2: Programação do ESP32 Mestre. Veículo 2

```
//mestre_0505_A
#include <esp_now.h>
#include <Wi-Fi.h>
//substitua pelo MAC Address do receptor.
uint8_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
//Exemplo de estrutura para enviar dados
//Deve corresponder à estrutura do receptor.
 typedef struct struct_message {
  int id; // must be unique for each sender board
  int x;
  int y;
  int z;
} struct_message;
//Cria uma mensagem struct, chamada myData.id.
struct_message myData;
//Cria interface de pares.
esp_now_peer_info_t peerInfo;
//retorna a chamada quando os dados são enviados.
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS? "Delivery Success":
"Delivery Fail");}
void setup() {
// configura o led_builtin
  pinMode (LED_BUILTIN, OUTPUT);
  //Inicia o monitor serial
  Serial.begin(115200);
  //Define dispositivo como uma estação Wi-Fi
  Wi-Fi.mode(Wi-Fi_STA);
  //Inicializa ESP_NOW.
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;  }
  //Assim que o esp_now for iniciado com sucesso vamos nos registrar para enviar
CB para.
  //Obter o status do pacote transmitido.
  esp_now_register_send_cb(OnDataSent);
  //Registra par
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  //Adicionar par.
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;  }
}
```

```
void loop() {
  //Define os valores para enviar.
      //strcpy(myData.id, "THIS IS A CHAR");
      myData.id = 2;
      myData.x=54;
      myData.y=33;
      myData.z=31;
      //Envia mensagens via esp_now
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));
  if (result == ESP_OK) {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("Sent with success");  }
  else {
    Serial.println("Error sending the data");
  }
  delay(300);
}
```

APÊNDICE 3, VERSÃO ANTERIOR MESTRE_CLIENTE

```
//PRIMEIRA VERSAO DO PROGRAMA_MESTRE_CLIENTE.
#include <Wi-Fi.h>

//Dados da rede
//Deve ser igual no Server
#define SSID "ESP32Server"
#define PASSWORD "12345678"
#define SERVER_PORT 5000

//Objeto que vai fazer a conexão com o server
Wi-FiClient client;

//Struct que define os dados que vamos enviar (deve ser igual no server)
typedef struct{
    int number;
    int status;
}Pin;

//Quantidade de pinos que iremos ler e enviar o status
#define PIN_COUNT 2

//Array com os pinos definidos
//No caso vamos trabalhar com os 21 e 19, mas você pode alterar para os pinos que
desejar
Pin pins[PIN_COUNT] = {
    {.number = 21},
    {.number = 19}
};

void setup(){
    Serial.begin(115200);
```

```
    //Tempo para considerar a conexão como perdida
    client.setTimeout(5000);

    //Conectamos à rede Wi-Fi e conectamos ao server
    setupWi-Fi();
    connectClient();

    for(int i=0; i<PIN_COUNT; i++){
        pinMode(pins[i].number, INPUT);
        sendPinStatus(i); //Enviamos para o server o estado atual dos pinos
    }

    //Geramos uma nova tarefa
    xTaskCreatePinnedToCore(
        handleConnection,   //Função que será executada
        "handleConnection", //Nome da tarefa
        10000,      //Tamanho da pilha
        NULL,       //Parâmetro da tarefa (no caso não usamos)
        2,          //Prioridade da tarefa
        NULL,       //Caso queria manter uma referência para a tarefa que vai ser criada
(no caso não precisamos)
        0);         //Número do core que será executada a tarefa (usamos o core 0 para o
loop ficar livre com o core 1)
}

void setupWi-Fi()
{
    Serial.print("Connecting to " + String(SSID));
    //Conectamos à rede Wi-Fi criado pelo outro ESP
    Wi-Fi.begin(SSID, PASSWORD);

    //Esperamos conectar
    while (Wi-Fi.status() != WL_CONNECTED)
    {
```

```
        Serial.print(".");
        delay(550);
    }

    //Se chegou aqui está conectado à rede Wi-Fi
    Serial.println();
    Serial.println("Connected!");
}

void connectClient()
{
    Serial.println("Connecting client");
    //Esperamos conectar com o server
    while (!client.connect(Wi-Fi.gatewayIP(), SERVER_PORT))
    {
        Serial.print(".");
        delay(550);
    }

    //Se chegou aqui está conectado com o server
    Serial.println();
    Serial.println("Client connected!");
}

void loop()
{
    //Se não estiver conectado à rede Wi-Fi, mandamos conectar
    if(Wi-Fi.status() != WL_CONNECTED)
    {
        setupWi-Fi();
    }
}

void handleConnection(void* pvParameters)
```

```
{
    //IMPORTANTE: A tarefa não pode terminar, deve ficar presa em um loop infinito
    while(true)
    {
        //Se não estiver conectado com o server, mandamos conectar
        if(!client.connected())
        {
            connectClient();
        }

        //Para cada pino, verificamos se mudou o estado. Se mudou enviamos para o server o novo estado
        for(int i=0; i<PIN_COUNT; i++)
        {
            if(hasPinStatusChanged(i))
            {
                sendPinStatus(i);
            }
        }

        //Delay de 5ms da tarefa. É feita em ticks. Para executar em millis dividimos pela constante portTICK_PERIOD_MS
        TickType_t taskDelay = 5 / portTICK_PERIOD_MS;
        vTaskDelay(taskDelay);
    }
}

//Verifica se o estado do pino na posição 'i' do array mudou
//Retorna 'true' se mudou ou 'false' caso contrário
boolean hasPinStatusChanged(int i)
{
    //Faz a leitura do pino
    int pinStatus = digitalRead(pins[i].number);
```

```
    //Se o estado do pino for diferente
    if(pins[i].status != pinStatus)
    {
        //Guardamos o novo estado e retornamos true
        pins[i].status = pinStatus;
        return true;
    }

    //Só chegará aqui se o estado não foi alterado
    //Então retornamos falso
    return false;
}

//Envia para o server os dados do pino na posição 'i' do array
void sendPinStatus(int i)
{
    client.write((uint8_t*)&pins[i], sizeof(Pin));
    client.flush();
}
```

APÊNDICE 4: PROGRAMA MESTRE_CLIENTE USANDO PEER_INFO

```
//mestre_2502
#include <Wi-Fi.h>
#include <Wire.h>
#include<esp_now.h>
#include <esp_Wi-Fi.h>
WiFiServer server(80);
int *p;
int result = 0;
static char i = 1;

int c [3] = {11, 31, 55};

uint8_t RxMACaddress[]{0xE0 , 0x06 , 0xE6, 0xCF, 0x4A, 0x9B};
typedef struct Txstruct
{
  int c;
}Txstruct;
Txstruct sentData;
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status){//
callback fuction
Serial.print("\r\nlast packed send status:\t");
Serial.println(status==ESP_NOW_SEND_SUCCESS); "Delivery
Success";"Delivery_Fail";
}

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
Wi-Fi.mode (Wi-Fi_STA);
if(esp_now_init()!=ESP_OK){
  Serial.println ("Error Initializing ESP_NOW");
  return;
```

```
}
esp_now_register_send_cb(OnDataSent);
esp_now_peer_info_t peerInfo;
memcpy (peerInfo.peer_addr, RxMACaddress, 6);

peerInfo.channel=0;
peerInfo.encrypt = false;

if(esp_now_add_peer(&peerInfo)!=ESP_OK);
return;
}

void loop() {
  // put your main code here, to run repeatedly:
  p = & c[i];

  result = *p;
sentData.c = result;
for(i=0;i<4;i++){

esp_err_t result = esp_now_send(RxMACaddress, (uint8_t*)&sentData,
sizeof(sentData));
if(result==ESP_OK) Serial.println("Sent with sucesss");
else Serial.println ("Error sending the data");
delay(500);
}
}
```

APÊNDICE 5: PROGRAMA MESTRE_CLIENTE_2104.

```
//mestre_2104
#include <esp_now.h>
#include <Wi-Fi.h>
//substitua pelo MAC Address do receptor.

uint8_t broadcastAddress[] = {0x9c, 0x9c, 0x1f, 0xe8, 0x90, 0xe4};
//Exemplo de estrutura para enviar dados
//Deve corresponder à estrutura do receptor.

typedef struct struct_message {
  long int id; // must be unique for each sender board
 //long int velocidade ;
  //velocidade = 65832;
  //long int latitude;
  //latitude = 66081;
  //long int longitude ;
  //longitude = 66338;
 // int i ;
  //int x;
  //int y;
} struct_message;
//Cria uma mensagem struct, chamada myData.id.

struct_message myData;
//Cria interface de pares.

esp_now_peer_info_t peerInfo;
//retorna a chamada quando os dados são enviados.

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
```

```
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
"Delivery Fail");
}

void setup() {
//  configura o led_builtin
  pinMode (LED_BUILTIN, OUTPUT);

  //Inicia o monitor serial

  Serial.begin(115200);
  //Define dispositivo como uma estação Wi-Fi

  Wi-Fi.mode(Wi-Fi_STA);
  //Inicializa ESP_NOW.

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //Assim que o esp_now for iniciado com sucesso vamos nos registrar para enviar
CB para.
  //Obter o status do pacote transmitido.

  esp_now_register_send_cb(OnDataSent);
  //Registra par

  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  //Adicionar par.

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
```

```
    return;
  }
}

void loop() {
  //Define os valores para enviar.
  for( int i=1;i<=4;i++){
    switch(i){
      case1:myData.id =65832;
      case2:myData.id = 66081;
      case3:myData.id = 66338;
      delay(500);
    }
  }

  //myData.x = random(0, 50);
  //myData.y = random(0, 50);
  //Envia mensagens via esp_now
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  delay(10000);
}
```

APÊNDICE 6: PROGRAMA MESTRE_CLIENTE_DE_0305.

```
//MESTRE_0305.
#include <esp_now.h>
#include <Wi-Fi.h>
//substitua pelo MAC Address do receptor.

uint8_t broadcastAddress[] = {0x9c, 0x9c, 0x1f, 0xe8, 0x90, 0xe4};
//Exemplo de estrutura para enviar dados
//Deve corresponder à estrutura do receptor.

typedef struct struct_message {
  int id; // must be unique for each sender board
  int x;
  int y;
} struct_message;
//Cria uma mensagem struct, chamada myData.id.

struct_message myData;
//Cria interface de pares.

esp_now_peer_info_t peerInfo;
//retorna a chamada quando os dados são enviados.

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
"Delivery Fail");
}

void setup() {
// configura o led_builtin
  pinMode (LED_BUILTIN, OUTPUT);
```

```
  //Inicia o monitor serial

  Serial.begin(115200);
  //Define dispositivo como uma estação Wi-Fi

  Wi-Fi.mode(Wi-Fi_STA);
  //Inicializa ESP_NOW.

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //Assim que o esp_now for iniciado com sucesso vamos nos registrar para enviar
CB para.
  //Obter o status do pacote transmitido.

  esp_now_register_send_cb(OnDataSent);
  //Registra par

  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  //Adicionar par.

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
}

void loop() {
  //Define os valores para enviar.

  myData.id = 11;
```

```
  myData.x = random(0, 50);
  myData.y = random(0, 50);
  //Envia mensagens via esp_now

  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  delay(10000);
}
```

APÊNDICE 7: PROGRAMA SERVIDOR_ROADSIDE_UNIT

Conforme programa do Apêndice 1, é o programa da Roadside Unit, funcional de nosso trabalho.

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <esp_now.h>
#include <Wi-Fi.h>
//endereço i2c do display
#define OLED_ADDR 0x3C
/* distância, em pixels, de cada linha em relação ao topo do display */
#define OLED_LINE1 0
#define OLED_LINE2 10
#define OLED_LINE3 20
#define OLED_LINE4 30
#define OLED_LINE5 40
/* Configuração da resolução do display (este modulo possui display 128x64) */
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
 #define SSD1306_BLACK 0
/* Objeto do display */
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, 16);
/* Variáveis */
//Exemplo de estrutura para receber dados
//Deve comparar a estrutura do remetente.
  int veiculo;
    int velocidade;
   int latitude;
   int longitude;
//Cria uma estrutura de mensagem chamada
myData.id.(myData.x,myData.y,myData.z)
//Crie uma estrutura para receber os dados. Esta estrutura deve ser a mesma definida pelo sketch do emissor.
```

```
//Exemplo de estrutura para receber dados
//Deve comparar a estrutura do remetente.


typedef struct struct_message {
   int id;
 int x;
 int y;
 int z;
  } struct_message;

//Exemplo de estrutura para receber dados
//Deve comparar a estrutura do remetente.

//Cria uma estrutura de mensagem chamada myData.id.

struct_message myData;
//Cria uma estrutura para leitura da placa.
//Função de retorno da chamada que será executada quando os dados forem recebidos
//Crie uma função de retorno de chamada que é chamada quando o ESP32 recebe os dados via ESP_NOW. A função é chamada onDataRecv() e deve aceitar vários parâmetros como segue

void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
//obtenha o endereço da placa.
  char macStr[18];
  Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
  Serial.println(macStr);
//Copie o conteúdo dos dados de entrada variável de dados para os meus dados variáveis.
```

```
  memcpy(&myData, incomingData, sizeof(myData));
//Agora meus dados . A estrutura contém várias variáveis com os valores enviados por um dos remetentes do ESP32Podemos Identificar qual placa envia o pacote pelo seu id:myData.id.
  Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
  //Atualiza as estruturas com os novos dados de entrada

  Serial.println();
}

void setup() {
//inicializa display OLED
      Wire.begin(4, 15);
      //display. Begin(SSD1306_SWITCHAPVCC,0x3c);
     if(!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR))
        Serial.println("Display OLED: falha ao inicializar");
      else
        {
        Serial.println("Display OLED: inicialização ok");
/* Limpa display e configura tamanho de fonte */
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
      }


  //configura o led_builtin
  pinMode (LED_BUILTIN, OUTPUT);
  //Inicializa o serial monitor.

  Serial.begin(115200);
  //Define dispositivo como estação Wi-Fi.

  Wi-Fi.mode(Wi-Fi_STA);
```

```
  //Inicializa o ESP_NOW

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //Assim que o ESP_NOW for iniciado com sucesso nos registramos para recv CB
para.
  //Obtém informações do empacotamento do recebimento.
//Registra-se para uma função de retorno de chamada que sera chamada quando os
dados forem recebidos. Neste caos , registramos o OnDataRecv() função que foi
criada anteriormente.
    esp_now_register_recv_cb(OnDataRecv);
}
void loop() {  //void OnDataRecv();

  veiculo = myData.id  ;
    velocidade = myData.x ;
   latitude= myData.y;
  longitude=myData.z;
  //escreve no display os valores recebidos via esp_now.
 display.clearDisplay();
     display.setCursor(0,OLED_LINE1);


     display.print("veiculo, num.  ");
     display.println(veiculo);
      display.setCursor(0,OLED_LINE2);
     display.print(velocidade);
     display.println(" km/h de velocidade");
      display.setCursor(0,OLED_LINE3);
     display.print(latitude);
     display.println(" graus de latitude");
      display.setCursor(0,OLED_LINE4);
```

```
   display.print(longitude);
   display.println(" graus de longitude");

   display.display();

  digitalWrite(LED_BUILTIN, HIGH);
//acessa as variáveis de cada placa.
delay(1300);}
```

APÊNDICE 8: PROGRAMA INICIAL DE UMA RSU_0311.

```
//RSU_1311
#include <Wi-Fi.h>

#include "heltec.h"
#include "Arduino.h"


const char* ssid = "RSU";
const char* password = "12345678";
WiFiServer server(80);

 static const uint8_t LED_BUILTIN = 25;
unsigned char c ;
unsigned char conta;

void setup() {
  // put your setup code here, to run once:

  Heltec.begin(true, false, true);

 Heltec.display->setContrast(255);
 Heltec.display->clear();

 Heltec.display->setFont(ArialMT_Plain_10);
 Heltec.display->drawString(0, 0, "latitude=c");

 Heltec.display->setFont(ArialMT_Plain_10);
 Heltec.display->drawString(0, 25, "longitude=c");

 Heltec.display->setFont(ArialMT_Plain_10);
 Heltec.display->drawString(0, 45, "velocidade=c");


 Heltec.display->display();
  pinMode (LED_BUILTIN,OUTPUT);
  Serial.begin(115200);
  Wi-Fi.mode(Wi-Fi_AP);
  Wi-Fi.softAP("OBU,12345678");
  Serial.println();
  Serial.print("Conectando-se a");
  Serial.println("RSU");
  Wi-Fi.begin("RSU,12345678");

  while (Wi-Fi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("");
  Serial.println("Wi-Fi conectada.");
  Serial.println("Endereço de IP:");
  Serial.println(Wi-Fi.localIP());

  server.begin();

}




void loop() {
  // put your main code here, to run repeatedly:



  Wi-FiClient client = server. available();
  if (client) {
    Serial.println("New Client.");
    String currentline = "";
    while (client.connected()) {
      if (client.available()) {
       client.write(c);


        if (c == '\n') {
          if (currentline.length() == 0) {
            client.println();
            //ligar o led builtin
            if(c==11){
              digitalWrite(LED_BUILTIN,HIGH);
            }else{
              digitalWrite(LED_BUILTIN,LOW);}
              delay(100);

            }
            break;
          } client.stop();
          Serial.println("cliente desconectado");
          conta++;
        }
      }
    }
  }
```

APÊNDICE 9: PROGRAMA ROADSIDE UNIT DE 10/03:

```
//RSU_1003.
#include <Wi-Fi.h>
#include <Wire.h>
#include<esp_now.h>
#include <esp_Wi-Fi.h>
WiFiServer server(80);
int *p;
int result = 0;
static char i = 1;

int c [3] = {11, 31, 55};

uint8_t RxMACaddress[]{0x94 , 0xb9 , 0x7e, 0xc3, 0x9f, 0x50};
typedef struct Txstruct
{
  int c;
}Txstruct;
Txstruct sentData;
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status){//
callback fuction
Serial.print("\r\nlast packed send status:\t");
Serial.println(status==ESP_NOW_SEND_SUCCESS); "Delivery
Success";"Delivery_Fail";
}

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
Wi-Fi.mode (Wi-Fi_STA);
if(esp_now_init()!=ESP_OK){
  Serial.println ("Error Initializing ESP_NOW");
  return;
```

```
}
esp_now_register_send_cb(OnDataSent);
esp_now_peer_info_t peerInfo;
memcpy (peerInfo.peer_addr, RxMACaddress, 6);

peerInfo.channel=0;
peerInfo.encrypt = false;

if(esp_now_add_peer(&peerInfo)!=ESP_OK);
return;
pinMode (LED_BUILTIN, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  p = & c[i];

  result = *p;
  for (i = 0; i < 4; i++) {

sentData.c = result;
if(result==11){
  digitalWrite(LED_BUILTIN,HIGH);
}

esp_err_t result = esp_now_send(RxMACaddress, (uint8_t*)&sentData,
sizeof(sentData));
if(result==ESP_OK) Serial.println("Sent with sucesss");
else Serial.println ("Error sending the data");
delay(500);
}
}
```

APÊNDICE 10: PROGRAMA RSU DE 2204.

```
//RSU_2204
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
//#include <splash.h>


//#include <Adafruit_GrayOLED.h>
//#include <Adafruit_SPITFT.h>
//#include <Adafruit_SPITFT_Macros.h>
//#include <gfxfont.h>



#include <esp_now.h>
#include <Wi-Fi.h>
//#include "heltec.h"
//endereço i2c do display
#define OLED_ADDR 0x30
/* distância, em pixels, de cada linha em relacao ao topo do display */
#define OLED_LINE1 0
#define OLED_LINE2 10
#define OLED_LINE3 20
#define OLED_LINE4 30
#define OLED_LINE5 40
/* Configuração da resolucao do display (este modulo possui display 128x64) */
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDR  0X3C
// #define SSD1306_BLACK 0
/* Objeto do display */
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, 16);
/* Variaveis */
```

```
int contador = 0;
//Exemplo de estrutura para receber dados
//Deve comparar a estrutura do remetente.
 long int veiculo;
  long int item;
  long int dado;
typedef struct struct_message {
  long int id;

  // int x;
  //int y;
} struct_message;
//Cria uma estrutura de mensagem chamada myData.id.

struct_message myData;
//Cria uma estrutura para leitura da placa.
struct_message board1;
//struct_message board2;
//struct_message board3;
//Cria uma array com todas as estruturas

struct_message boardsStruct[1] = {board1};
//Função de retorno da chamada que sera executada quando os dados forem
recebidos


void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
  char macStr[18];
  Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
  Serial.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));
```

```
  Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
  //Atualiza as estruturas com os novos dados de entrada

  // boardsStruct[myData.id-1].x = myData.x;
  // boardsStruct[myData.id-1].y = myData.y;
  //Serial.printf("x value: %d \n", boardsStruct[myData.id-1].x);
  //Serial.printf("y value: %d \n", boardsStruct[myData.id-1].y);
  Serial.println();
}


void setup() {


  //configura o led_builtin
//  pinMode (LED_BUILTIN, OUTPUT);
  //configura display.
  // Heltec.begin(true, false, true);

 //Heltec.display->setContrast(255);
 //Heltec.display->clear();

 //Heltec.display->setFont(ArialMT_Plain_24);
 //Heltec.display->drawString(0, 0, "Fatec SA");

 //Heltec.display->setFont(ArialMT_Plain_10);
 //Heltec.display->drawString(0, 25, "Eletr. Autom.");

 //Heltec.display->setFont(ArialMT_Plain_24);
 //Heltec.display->drawString(0, 45, "TCC");


 //Heltec.display->display();
  //Inicializa o serial monitor.
```

```
  Serial.begin(115200);
  //Define dispositivo como estação Wi-Fi.

  Wi-Fi.mode(Wi-Fi_STA);

  //Inicializa o ESP_NOW

 if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //Assim que o ESP_NOW for iniciado com sucesso nos registramos para recv CB
para.
  //Obtém informações do empacotamento do recebimento.
  esp_now_register_recv_cb(OnDataRecv);
    //inicializa display OLED
      Wire.begin(4, 15);
      //display.begin(SSD1306_SWITCHAPVCC,0x3c);
     if(!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR))
         Serial.println("Display OLED: falha ao inicializar");
      else
         {
         Serial.println("Display OLED: inicializacao ok");

/* Limpa display e configura tamanho de fonte */
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
      }
}
void loop() {
  //void OnDataRecv();
  veiculo = myData.id  ;
```

```
 veiculo  >> 24;
// número do veiculo
 if(veiculo==1){

   display.clearDisplay();
   display.setCursor(0, OLED_LINE1);
   display.println("veiculo1");
    display.display();
    //delay(500);

 }if(veiculo==2){

   display.clearDisplay();
   display.setCursor(0,OLED_LINE1);
  display.println("veiculo2");

 }
 item = myData.id ;
 item>>16;
 item&0x000F;
 //número do item
 switch(item){
  case1:
   display.clearDisplay();
   display.setCursor(0,OLED_LINE2);
   display.println("velocidade");

  case2:
  display.clearDisplay();
  display.setCursor(0,OLED_LINE2);
  display.println("latitude");

  case3:
  display.clearDisplay();
```

```
    display.setCursor(0,OLED_LINE2);
    display.println("longitude");

     }
   display.display();
  dado = myData.id;
 dado &0x00FF;
 contador++;
// valor do dado.

  //compara se a variável recebida e 11
  if (myData.id == 11) {
   digitalWrite(LED_BUILTIN, HIGH);

  //acessa as variáveis de cada placa.

  delay(550);
  //contador++;

}
}
```

APÊNDICE 11: PROGRAMA RSU_0505:

```
//RSU_0505


#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <esp_now.h>
#include <Wi-Fi.h>
//endereço i2c do display
#define OLED_ADDR 0x30
/* distância, em pixels, de cada linha em relacao ao topo do display */
#define OLED_LINE1 0
#define OLED_LINE2 10
#define OLED_LINE3 20
#define OLED_LINE4 30
#define OLED_LINE5 40
/* Configuração da resolucao do display (este modulo possui display 128x64) */
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDR  0X3C
 #define SSD1306_BLACK 0
/* Objeto do display */
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, 16);
/* Variaveis */
//Exemplo de estrutura para receber dados
//Deve comparar a estrutura do remetente.
  int veiculo;
    int velocidade;
   int latitude;
   int longitude;
typedef struct struct_message {
   int id;
 int x;
```

```
 int y;
 int z;
  } struct_message;
//Cria uma estrutra de mensagem chamada myData.id.

struct_message myData;
//Função de retorno da chamada que sera executada quando os dados forem
recebidos
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
  //memcpy(&myData, incomingData, sizeof(myData));
  char macStr[18];
  Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
  Serial.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
  Serial.println();
}
void setup() {
  //inicializa display OLED
      Wire.begin(4, 15);
      //display.begin(SSD1306_SWITCHAPVCC,0x3c);
     if(!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR))
         Serial.println("Display OLED: falha ao inicializar");
      else
         {
         Serial.println("Display OLED: inicializacao ok");
/* Limpa display e configura tamanho de fonte */
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
      }
```

```
  //configura o led_builtin
  pinMode (LED_BUILTIN, OUTPUT);
  //Inicializa o serial monitor.
  Serial.begin(115200);
  //Define dispositivo como estação Wi-Fi.
  Wi-Fi.mode(Wi-Fi_STA);
  //Inicializa o ESP_NOW
 if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //Assim que o ESP_NOW for iniciado com sucesso nos registramos para recv CB
para.
  //Obtém informações do empacotamento do recebimento.
  esp_now_register_recv_cb(OnDataRecv);
}
void loop() {
  //void OnDataRecv();
  veiculo = myData.id  ;
    velocidade = myData.x ;
   latitude= myData.y;
  longitude=myData.z;
  //escreve no display os valores recebidos via esp_now.
     display.clearDisplay();
     display.setCursor(0,OLED_LINE1);
     display.print(veiculo);
     display.println("veiculo  ");
      display.setCursor(0,OLED_LINE2);
     display.print(velocidade);
     display.println("velocidade  ");
      display.setCursor(0,OLED_LINE3);
     display.print(latitude);
     display.println("latitude  ");
      display.setCursor(0,OLED_LINE4);
```

```
    display.print(longitude);
    display.println("longitude  ");
    display.display();

   digitalWrite(LED_BUILTIN, HIGH);

  //acessa as variáveis de cada placa.

  delay(1300);

}
```

ANEXO 1- Instalação da plataforma IDE-Arduino (ESP32)

1. Na Arduino -IDE, vá até a tela de preferencias clicando em *FILE>PREFERENCES*
2. Clique no botão ao lado do campo ***"****Additional Boards Manager URLs"*
3. Na nova janela que abrir, no campo" *Enter Additional URLs, one for each row*", insira o seguinte e endereço: https://dl.espressif.com/dl/package_esp32_index.json e clique em "ok"
4. Na janela preferencias clique em "ok"
5. Va até o gerenciador de placas, clicando em *TOOLS>BOARD>BOARDS MANAGER*...
6. Na janela do gerenciador de placas, insira no campo de busca o texto "esp32" (sem aspas)
7. Clique sobre a opção "*esp32 by Espressif Systems*" e depois no botão "*Install*"

Aguarde o download e instalação da biblioteca estará instalada.[1]