UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
BACHARELADO EM ENGENHARIA ELETRÔNICA

JOSÉ LUIZ PASCHOAL NETO

SISTEMA DE SEMÁFORO INTELIGENTE UTILIZANDO O PROTOCOLO ESP-NOW

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2020

JOSÉ LUIZ PASCHOAL NETO

SISTEMA DE SEMÁFORO INTELIGENTE UTILIZANDO O PROTOCOLO ESP-NOW

Smart Traffic System Using The ESP-NOW Protocol

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador: Prof. Dr. Márcio Rodrigues da Cunha.

CAMPO MOURÃO

2020

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação de Engenharia Eletrônica

TERMO DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

SISTEMA DE SEMÁFORO INTELIGENTE UTILIZANDO O PROTOCOLO ESP-NOW

DO DISCENTE

JOSÉ LUIZ PASCHOAL NETO

Trabalho de Conclusão de Curso apresentado no dia **04 de dezembro de 2020** ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O(A) discente foi arguido(a) pela Comissão Examinadora composta pelos professores abaixo assinados. Após deliberação, a comissão considerou o trabalho **aprovado com alterações**.

_____

Prof. Dr. Flávio Luiz Rossini
Avaliador(a) 1
UTFPR

_____

Prof. Me. Marcelo Nanni
Avaliador(a) 2
UTFPR

_____

Prof. Dr. Marcio Rodrigues da Cunha
Orientador(a)
UTFPR

RESUMO

Os semáforos são objetos comumente presentes nos cenários dos centros urbanos, sendo responsáveis por controlar o fluxo de veículos de forma organizada e evitar acidentes de trânsito. Este trabalho apresenta um estudo para o desenvolvimento de um sistema de semáforo que trabalha de forma inteligente, monitorando o qual por meio de sensores ultrassônicos o tráfego de veículos em um cruzamento para ajustar o tempo de sinalização de acordo com as informações obtidas, de forma a diminuir o tempo ocioso que os motoristas perdem diariamente. O protótipo desenvolvido utiliza a placa ESP32 e um protocolo de comunicação sem fio que foi criado especificamente para essa família de dispositivos, que é o protocolo ESP-NOW. Os testes realizados demonstraram o correto funcionamento do protótipo, sendo assim, é possível aplicar o sistema em um ambiente real.

Palavra-chave: ESP-NOW. Semáforos. HC-SR04. ESP32.

ABSTRACT

Traffic lights are objects commonly present in urban centers, being responsible for controlling the flow of vehicles in an organized way and preventing traffic accidents. This work presents a study for the development of a traffic light system that works intelligently, monitoring vehicle traffic at an intersection using ultrasonic sensors and adjusting the signaling time according to the information obtained, seeking to reduce the idle time that drivers lose daily. The prototype developed uses the ESP32 board and a wireless communication protocol that was created specifically for this family of devices, which is the ESP-NOW protocol. The tests carried out demonstrated the good functioning of the prototype, so it is possible to apply the system in a real environment.

Keywords: ESP-NOW. Traffic lights. HC-SR04. ESP32.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

Soc	*System on a Chip* (Sistema em um Chip)
UCP	*Unidade Central de Processamento*
E/S	Entrada/Saída
WLAN	*Wireless Local Area Network (Rede Local Sem Fio)*
WPAN	*Wireless Personal Area Network (Rede de Área Pessoal Sem Fio)*
WMAN	*Wireless Metropolitan Area Network* (Rede de Área Metropolitana Sem Fio)
WWAN	*Wireless Wide Area Network (Rede de Longa Distância Sem Fio)*
Wi-Fi	*Wireless Fidelity (Fidelidade Sem Fio)*
Hi-Fi	*High Fidelity (Alta Fidelidade)*
IEEE	*Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos)
CCMP	*Counter Mode Cipher Block Chaining Message Authentication Code Protocol* (Protocolo de Código de Autenticação de Mensagem de Encadeamento de Blocos de Codificação de Modo Contador)
P2P	*Peer-to-peer (Ponto a Ponto)*
BSA	*Basic Service Area (Área Básica de Serviço)*
BSS	*Basic Service Set (Conjunto Básico de Serviço)*
AP	*Acess Point (Ponto de Acesso)*
DFWMAC	*Distributed Foundation Wireless Medium Access Control* (Fundação Distribuída de Controle de Acesso ao Meio)
DCF	*Distributed Coordination Function (Função de Coordenação Distribuída)*
PCF	*Point Coordination Function (Função de Coordenação de Ponto)*
CSMA/CA	*Carrier-Sense Multiple Acess/Collision Avoidance* (Acesso Múltiplo com Verificação de Portadora com Anulação/Prevenção de Colisão.

SUMÁRIO

1. INTRODUÇÃO

Nos dias atuais é comum presenciar o congestionamento de carros e pessoas, principalmente, nas grandes cidades. No século 19, em Londres, onde o trânsito era formado majoritariamente por carruagens com cavalos, já havia preocupação com a organização do trânsito. Naquela época, 1865, surgiu o primeiro semáforo, desenvolvido pelo engenheiro John Peake Knight (ALBUQUERQUE, 2017).

A sinalização com semáforos é um subsistema da sinalização viária. É composto por 3 sinais luminosos que são acionados de forma alternada ou intermitente, por sistemas que podem ser eletromecânicos ou eletrônicos. Essa sinalização tem por objetivo transmitir diferentes mensagens aos usuários das vias públicas, para informar sobre o direito de passagem ou alertar sobre situações especiais nas vias (CONTRAN, 2014).

Existem dois tipos de semáforos (controladores de tráfego): (1) de tempo fixo e (2) por demanda de tráfego. O primeiro tipo é o mais comum no Brasil, e funciona com período fixo de tempo dos sinais luminosos, independente do volume de tráfego. Já o semáforo por demanda é mais complexo, e possui sensores de detecção de veículos e uma lógica de decisão baseada no volume de tráfego em cada instante, para ajustar o tempo dos sinais dinamicamente (CIRINEU, 2006).

Grande parte do tempo que os motoristas ficam parados no trânsito, se deve aos semáforos mal regulados, que obrigam os motoristas a ficarem parados, aguardando o sinal verde, enquanto a via transversal possui poucos ou nenhum veículo passando. Em vias de muito movimento, ocorre de motoristas só conseguirem atravessar o cruzamento depois que o sinal verde se repete 3 vezes (SABURO, 2008).

O semáforo é uma ferramenta importante na sinalização de trânsito, para controlar a passagem de veículos e pessoas nos cruzamentos. O número de infrações por avançar o sinal vermelho é alto, só no estado do Mato Grosso do Sul foram 21.482 multas em 2016 (DETRAN-MS, 2017).

Avançar o sinal vermelho coloca em risco a vida dos condutores de veículos e de pedestres. E um dos motivos que podem levar os motoristas a avançar o sinal

vermelho é a pressa de se chegar ao local de destino. Por isso, um sistema de semáforo inteligente, que diminua o tempo de espera quando for possível, pode reduzir esse tipo de infração de trânsito e consequentemente os números de acidentes nos cruzamentos.

Os semáforos por demanda de tráfego apresentam as seguintes vantagens em relação aos de tempo fixo: normalmente reduz o tempo de espera, adaptando-se as variações do fluxo de veículos e normalmente aumenta a capacidade do fluxo na via, devido ao continuo acrescimento de tempo de sinal verde. Por outro lado, apresenta as seguintes desvantagens: o custo de instalação é maior e os sensores que detectam o volume de tráfego precisam de manutenção especifica (SABURO, 2008).

Percebe-se que as desvantagens de um controlador por demanda de tráfego estão relacionadas a uma questão financeira. Por isso, o projeto desenvolvido nesse trabalho utiliza componentes de baixo custo, de forma a desenvolver um sistema mais barato caso venha a ser utilizado em um cruzamento real.

Na década de 90 surgiu o termo cidade inteligente, que se trata de aumentar o bem estar da população e melhorar o relacionamento que as grandes cidades têm com a natureza, utilizando como aliado às novas tecnologias. Os projetos de cidade inteligente podem ser aplicados em várias áreas, como: transporte, saúde e educação, buscando sempre o objetivo de melhorar a relação dos moradores com o ambiente em que estão inseridos (CASTRO, 2018).

De acordo com um levantamento feito pela ONU-HABITAT, em 1990 havia cerca de 43% da população mundial vivendo em áreas urbanas, em 2014 esse número atingiu 54% da população e a projeção para 2030 é que esse número aumente para 60%. Esse crescimento das áreas urbanas necessita de recursos para manter a organização nesses locais, por isso é importante fazer o uso das tecnologias ditas “inteligentes” como aliado nesse objetivo (SOUZA, 2017).

Devido ao crescimento acelerado da urbanização, a discussão sobre mobilidade urbana tornou-se indispensável na administração pública, principalmente em cidades que estão em desenvolvimento e possuíam seu planejamento urbano centrado no uso de automóveis particulares. Trazendo como consequência da

redução da mobilidade urbana, problemas na economia, qualidade de vida e meio ambiente (SÓ, 2017).

Grandes empresas de tecnologia vêm apresentando novas soluções para o problema da mobilidade urbana, como a *Google*, que lançou em 2016 uma solução que informa para o usuário em tempo real como está o trânsito em todas as regiões da cidade e indica o trajeto que levará um tempo menor para chegar ao local de destino. Esse exemplo mostra como esse tema se tornou relevante nos últimos anos e a importância que tem vida das pessoas (CARVALHO, 2019).

Dessa forma, este trabalho pode ser categorizado como um projeto para cidades inteligentes, atuando especificamente na categoria de transporte e mobilidade urbana.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver o protótipo de um semáforo inteligente (por demanda de tráfego), que analise em tempo real o fluxo em um cruzamento de trânsito, fazendo uso de sensores ultrassônicos para identificação da passagem dos veículos, e uma rede de comunicação sem fio, utilizando o protocolo ESP-Now, para comunicação entre os dispositivos, de modo a determinar a duração do tempo em que o semáforo estará emitindo os sinais, verde, amarelo e vermelho.

1.1.2 Objetivos Específicos

Para exequibilidade do tema e do objetivo geral, os objetivos específicos serão elencados a seguir:

1. Análise de parâmetros de operação e normas técnicas referentes aos semáforos;
2. Desenvolver a comunicação entre dispositivos da família ESP32, através do protocolo ESP-NOW;
3. Analisar o modo de funcionamento do sensor ultrassônico;
4. Desenvolver o algoritmo para controlar o tempo do semáforo, de forma eficiente, com base no fluxo de veículos;
5. Construir uma maquete para testar o protótipo de forma eficiente.

1.2 Justificativa

O número de veículos no Brasil está aumentando cada vez mais. Só em 2017, o crescimento foi de 1,2%, alcançando a marca de 43,4 milhões, que representa uma proporção de 4,8 habitantes por veículo (CORREIO BRAZILIENSE, 2017).

Esse grande número de automóveis nas ruas, implica na necessidade de um aumento do controle de tráfego nas cidades, pois pode causar efeitos negativos, como engarrafamentos, crescimento do número de acidentes e estresse nos motoristas que ficam mais tempo no trânsito.

Por isso, se faz necessário diminuir o tempo de espera nos semáforos, para melhorar o fluxo de veículos, e consequentemente uma melhor qualidade de vida para os motoristas que ficarão menos tempo parados nos semáforos e irão chegar mais rápido ao seu destino.

A fim de se obter uma redução no tempo de espera, em relação aos semáforos de tempo fixo, que são comumente encontrados na maioria das grandes cidades.

Além disso, busca-se, o crescimento acadêmico e pessoal, com o desenvolvimento de projetos utilizando tecnologia de comunicação sem fio entre vários dispositivos para integrar um sistema de tempo real.

2. FUNDAMENTAÇÃO TEÓRICA

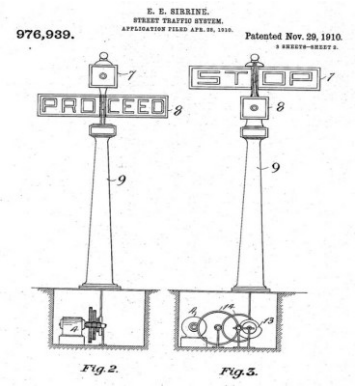
Neste capítulo são apresentadas as bases teóricas utilizadas para a compreensão do trabalho. Uma visão geral sobre a sinalização semafórica é explanada nos dois primeiros tópicos, em seguida é explicado o ESP32 que consiste na placa utilizada para construção do hardware do projeto e nos tópicos seguintes são explicados o funcionamento de redes sem fio e o protocolo ESP-NOW.

2.1 Semáforos

O congestionamento de veículos e pessoas existe a muito tempo nas grandes cidades. No século 19 em Londres, as carruagens com cavalos eram um dos principais meios de locomoção. Nesse período foi inventando o semáforo, em 1865, pelo engenheiro John Peake Knight, que pensou em criar uma invenção para evitar acidentes e organizar o trânsito. O semáforo de John era operado por guardas da cidade que se revezavam em turnos. Porém, após um mês de implantação do semáforo ouve um acidente, devido ao vazamento de gás de uma das lâmpadas, ocasionando a morte de um policial que gerenciava a invenção (ALBUQUERQUE, 2017).

Depois do acidente, os inventores da época desistiram da ideia de semáforo, que só retornou em 1910 com o americano Ernest Stirrine, que criou e patenteou outro modelo de semáforo, que passou a ser usado para organizar o trânsito na cidade de Chicago, nos Estados Unidos. O semáforo de Ernest consistia em duas torres com placas escritas “proceed” e “stop” (que significam continue e pare em português), que eram operadas manualmente girando-se uma manivela (ALBUQUERQUE, 2017), como pode ser visto na Figura 1.

Figura 1 - Diagrama do semáforo criado por Ernest Stirrine.



Fonte: Abuquerque, 2017.

No Brasil, se tem relato da primeira instalação de um semáforo em 1935, em São Paulo, no bairro do Brás, época em que a indústria automobilística se popularizou no país. De lá pra cá a invenção se popularizou, e hoje é comum encontrar semáforos até em cidades pequenas (ESTADAO, 2013).

2.1.1 Definição E Sinalização Semafórica

Existem dois tipos de semáforos: veicular e para pedestres. O semáforo veicular, que será tratado nesse trabalho, é definido como um dispositivo geralmente composto por 3 focos de luz, um de cor vermelha, um de cor amarela e outro de cor verde. Podem existir focos auxiliares, com funções especificas, como o verde e o vermelho seta para movimentos de conversão ou sempre livre (DENATRAN, 1984).

A sinalização semafórica de regulamentação tem o objetivo de transmitir informações aos usuários sobre o direito de passagem em interseções onde o espaço é disputado por dois ou mais movimentos conflitantes, alternando o direito de passagem através de indicações luminosas (CONTRAN, 2014). As diferentes combinações de cores transmitem aos condutores informações distintas, como mostrado no Quadro 1.

Quadro 1 - Significado das cores emitidas pelos semáforos.

Cor	Significado
Vermelho	Indica a proibição do direito de passagem
Amarelo	Indica o término do direito de passagem.
Verde	Indica a permissão do direito de passagem
Amarelo (Intermitente)	Adverte da existência de situação perigosa ou obstáculo.

Fonte: Baseado em CONTRAN, 2014.

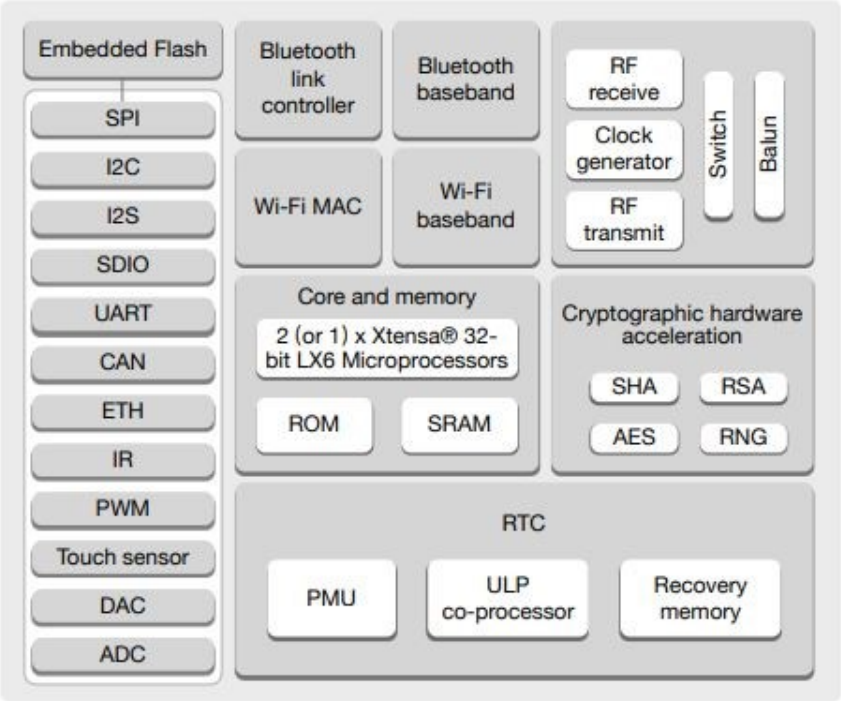
2.2 ESP32

Microcontroladores são sistemas computacionais em miniatura. Possuem UCP (Unidade Central de Processamento), memória e portas de E/S (LOBUR, 2010).

SoCs (Sistemas em um Chip), são outra classe de dispositivos que se distinguem dos microcontroladores, por ter uma complexidade maior e mais recursos no chip. Muitas vezes, os microcontroladores necessitam de circuitos de suporte, como processadores de sinal, decodificadores e conversores de sinal. Ao contrário dos SoCs, que são formados de uma peça única contendo todos os circuitos necessários para realizar suas funções específicas (LOBUR, 2010).

O ESP32 é um SoC desenvolvido pela Espressif, que suporta tecnologia Wi-Fi e Bluetooth, dentre outros recursos, e possui um processador de 32 bits (ESP32, 2018). A Figura 2 mostra o diagrama de bloco com os principais recursos do chip.

Figura 2 - Diagrama em blocos do ESP32.



Fonte: ESP32 SERIES, 2018.

2.3 Rede Wireless

O termo wireless, que em tradução livre significa sem fio, é definido como qualquer tipo de conexão para transmissão de informações sem o uso de cabos ou fios. As redes wireless funcionam com equipamentos que utilizam radiofrequência, como a comunicação via satélite, a comunicação via infravermelho, entre outros. Essas redes são divididas em quatro tipos: A rede local (WLAN), de curta distância (WPAN), metropolitana (WMAN) e de longa distância (WWAN) (POZZEBOM, 2012).

Quase na mesma época em que surgiram os notebooks, as pessoas desejavam poder chegar no escritório e ter seu computador conectado a internet como se fosse mágica. Em decorrência disso, muitos grupos começaram a trabalhar para alcançar esse objetivo. A forma mais prática encontrada foi equipar os notebooks com transmissores e receptores de rádio de ondas curtas para conseguir uma comunicação entre eles. Com isso, muitas empresas começaram a comercializar redes sem fio (TANENBAUM, 2011).

Houve um problema quando começaram a ser comercializadas essas redes sem fio, elas não eram compatíveis. Por exemplo, um computador equipado com a marca A não conseguia se comunicar com o computador da marca B. Devido a esse problema, a indústria decidiu que seria melhor adotar um padrão para essas redes sem fio e passou a tarefa para um comitê da IEEE, que em 1990 criou o padrão que foi denominado 802.11 (TANEMBAUM, 2011).

Uma expressão que é comumente utilizado quando se fala de redes sem fio é o Wi-Fi, que nada mais é que uma marca registrada pela Wi-Fi Alliance, que acabou se tornando um sinônimo de wireless. A expressão Wi-Fi é uma contração das palavras Wireless Fidelity, e surgiu como referência ao termo Hi-Fi (High Fidelity), utilizado pela indústria de equipamentos de som (ESPERIDIÃO, 2018).

2.4 Protocolo ESP-NOW

ESP-Now é um protocolo de comunicação sem fio desenvolvido pela Espressif, que permite a comunicação entre vários dispositivos sem precisar de uma rede Wifi utilizando um roteador. A rede de comunicação sem fio criada com esse protocolo trabalha com a mesma frequência e os mesmos canais utilizados pelos roteadores Wifi (KOYANAGI, 2018).

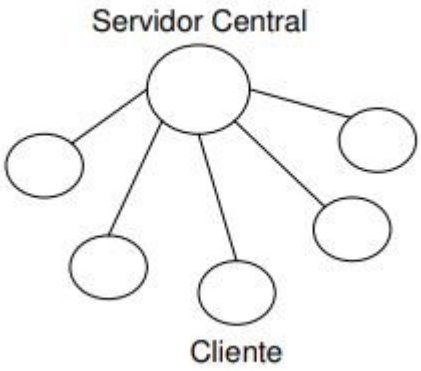
O ESP-Now aplica o padrão IEEE802.11 juntamente com funções desenvolvidas pela Espressif, que permitem enviar e receber de uma forma simples, sem precisar fazer muitas configurações. Além disso, o protocolo possui uma tecnologia de criptografia conhecida como CCMP. A topologia de rede utilizada é *peer to peer.* O protocolo permite a comunicação de dispositivos com criptografia e sem criptografia, e suporta até 20 pares na mesma rede, desses 20 pares, no máximo 10 podem contar com a tecnologia de criptografia. *(ESP-NOW, 2016).*

Em um teste realizado em campo aberto, a distância de comunicação alcançada entre dois dispositivos da família ESP32*,* utilizando o protocolo ESP-Now foi de 165,4 metros (KOYANAGI, 2018).

2.5 Topologias De Rede P2P E Cliente/Servidor

A maioria dos serviços de internet utilizam o modelo cliente/servidor, mostrado na Figura 3. Nesse modelo o cliente tem um papel passivo, pois precisa fazer requisições ao servidor para acessar os serviços, e não pode se comunicar diretamente com os outros clientes da rede. Como o servidor é a parte central da rede, uma desvantagem dessa topologia é que caso ocorra alguma falha nele, toda a rede irá parar de funcionar. Outra desvantagem desse modelo é que pode ocorrer sobrecarga no servidor, se a rede possuir um grande número de clientes (SANTOS, 2007).

Figura 3 - Topologia cliente/servidor.

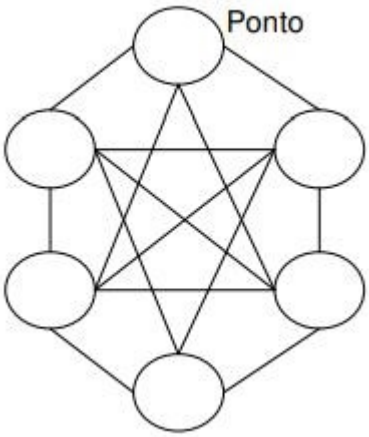


Fonte: Santos, 2007.

Na topologia P2P todos os pontos da rede são capazes de se comunicar diretamente com os outros pontos, sem precisar passar por um ponto central, como acontece no modelo cliente/servidor. E por isso não existe uma hierarquia nesse modelo, pois todos os pontos podem realizar as mesmas funções. Cada ponto é responsável por manter as informações de seus próprios dados, e quando recebe uma solicitação de consulta, pode responder e/ou reescrever essa solicitação e repassar aos pontos vizinhos. Uma desvantagem nessa topologia está na escalabilidade do sistema, devido a necessidade de uma grande largura de banda caso a rede seja

constituída de muitos pontos, isso ocorre pois é necessário saber o caminho de ligação entre todos os pontos (SANTOS, 2007). A Figura 4 ilustra essa topologia.

Figura 4 - Topologia P2P.



Fonte: Santos, 2007.

2.6 Padrão IEEE 802.11

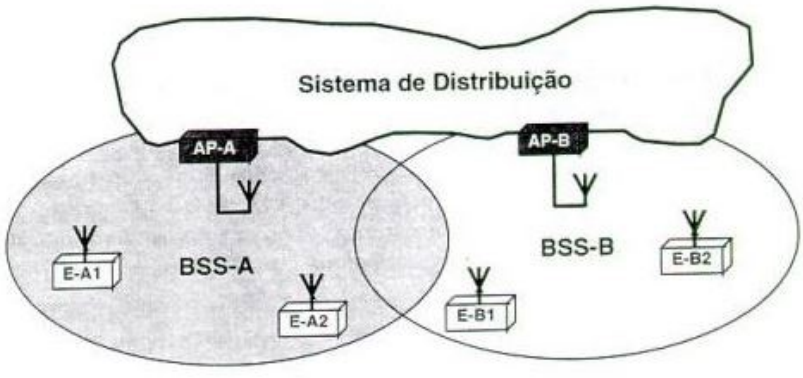
O IEEE formou um grupo conhecido como “Wireless Loca-Area Networks Standard Working Group, IEEE Projetc 802.11”. Com objetivo de definir um nível físico para redes com transmissão feitas na frequência de rádio ou infravermelho, e um protocolo de acesso ao meio (SOARES; LEMOS; COLCHER, 2004).

Diversas técnicas de transmissão são utilizadas nesse padrão: espectro espalhado, saltos de frequência espalhada e modulação de posição de pulso (no caso da transmissão por infravermelho). A velocidade de dados varia de 1 Mbps a 2 Mbps, nas bandas de frequência de 2,4 – 2,4835 GHz (BRAGA, 2018).

2.6.1 Arquitetura

A arquitetura implementada pelo projeto IEEE 802.11, para redes sem fio, se baseia na divisão da área coberta pela rede em células. Essas áreas são chamadas de BSA (Basic Service Area), e seu tamanho depende das características dos transmissores e receptores utilizados e do ambiente. Um grupo de estações se comunicando em uma BSA, constitui um BSS (Basic Service Set). Existem também os APs (Acess Points), que são estações especiais responsáveis pela captura das transmissões realizadas pelas estações de sua BSA, com destino a estações localizadas em outras BSAs, retransmitindo a informação recebida pelo sistema de distribuição (que pode ser formado por uma rede com outro meio de transmissão, como fios metálicos ou fibra ótica). E um conjunto formado pela união de vários BSSs conectados por um sistema de distribuição define um ESS (Extended Service Set), que forma uma de rede sem fio IEEE 802.11, como pode ser visto na Figura 5 (SOARES; LEMOS; COLCHER, 2004).

Figura 5 - Rede sem fio com infraestrutura (ESS).



Fonte: Soares; Lemos; Colcher, 2004.

Existe um caso especifico nessa arquitetura, onde o ESS é formado por um único BSS. Esse tipo de rede é conhecida como Ad Hoc e permite a comunicação entre estações próximas sem nenhum tipo de infra-estrutura, ou seja, sem a necessidade dos APs para estabelecer as comunicações entre as estações (UZEDA, S/D). Esse é o caso do ESP-Now, que realiza comunicação entre os dispositivos

próximos, sem precisar de um roteador (KOYANAGI, 2018), que seria o ponto de acesso (AP) na arquitetura de rede mostrada na Figura 5.

2.6.2 Protocolo MAC Do Padrão IEEE 802.11

Além de definir uma forma de transmissão física utilizando radiofrequência ou infravermelho, o IEEE também definiu um protocolo de acesso ao meio, que foi denominado DFWMAC (Distributed Foundation Wireless Medium Access Control). Esse protocolo suporta dois métodos: o método distribuído e o método centralizado. Esses métodos também são chamados de “funções de coordenação”, e tem a função de determinar quando uma estação específica tem permissão para transmitir (UZEDA, 2018).

Se a função de coordenação for distribuída (Distributed Coordination Function-DCF), é tomada individualmente pelos nós, a decisão de quando transmitir, podendo ocorrer transmissões simultâneas. Porém, se a função de coordenação for pontual (Point Coordination Function-PCF), a decisão de qual estação irá transmitir é centralizada em um ponto, evitando a ocorrência de colisões. No entanto, a função de coordenação pontual não é aplicada em redes sem infraestrutura (SOARES; LEMOS; COLCHER, 2004) como o ESP-Now, por isso não será aprofundada nesse trabalho.

2.6.3 Função De Coordenação Distribuida (CSMA/CA)

O método de acesso ao meio básico do DFWMAC é uma função de coordenação distribuída (DCF) conhecida como CSMA/CA (Carrier-Sense Multiple Acess/Collision Avoidance). É obrigatório a utilização desse método pelas estações e pontos de acesso, com ou sem infraestrutura (SOARES; LEMOS; COLCHER, 2004).

As regras do CSMA/CA funcionam basicamente da seguinte forma: A estação que quer transmitir sente o meio, para saber se há outras estações transmitindo, se o meio estiver ocupado, então ela irá esperar um tempo aleatório para poder transmitir novamente seu quadro (pacote de dados). Se o meio estiver livre, então a estação envia o quadro e espera um tempo de resposta do receptor, se a resposta chegar

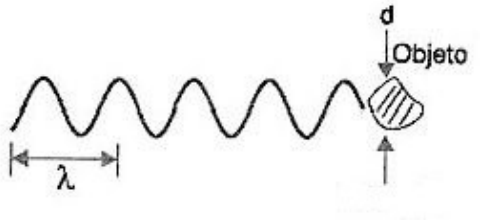
dentro do tempo limite significa que o quadro foi recebido. Se não chegar, a estação irá dobrar o tempo de espera e tentar enviar novamente (TANEMBAUM, 2011).

2.7 Sensores Ultrassônicos

Sensores ultrassônicos são dispositivos utilizados para detecção e medição de objetos. Com esses dispositivos é possível detectar objetos que estão em uma determinada área ou região, independente de seu estado físico. Eles são utilizados em diversas áreas, como: medicina, instrumentação, automação industrial, entre outras.

Os sensores trabalham com ondas sonoras que possuem uma frequência maior que 20kHz, e por isso não são perceptíveis aos ouvidos humanos. Através de estímulos elétricos, um transdutor irá emitir ondas sonoras que se propagam pelo ar até encontrarem um objeto (SOUSA, 2014). Se a área do objeto for maior que o comprimento de onda λ da onda emitida, ela será refletida de volta para o receptor do sensor, como pode ser visto na Figura 6. De acordo com o tempo decorrido entre a emissão e o retorno da onda ao sensor é possível calcular a distância que o objeto se encontra, sabendo que ondas sonoras se propagam no ar com uma velocidade de aproximadamente 340 metros por segundo (BRAGA, 2012).

Figura 6 - Ilustração de uma onda sonora atingindo um objeto.



Fonte: Adaptado de Braga, 2012.

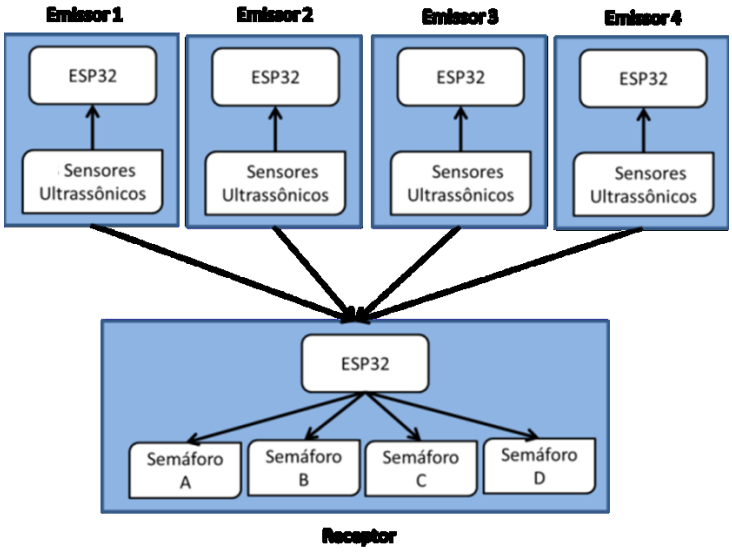
3. METODOLOGIA

Para implementar o sistema é preciso identificar através de sensores, o número de veículos que trafegam por cada uma das 4 vias que formam o cruzamento. Sendo assim, é necessário que os dados lidos pelos sensores sejam enviados para o receptor, dentro de uma estreita faixa de tempo. O receptor é responsável pelo processamento dos dados enviados pelos 4 emissores, e o acionamento dos semáforos.

A decisão sobre o período de tempo em que a sinalização ficará em verde para uma via e vermelho para a outra será feita com 3 intervalos de tempo distintos, sendo o menor para quando houver um fluxo baixo de veículos. O intermediário para quando existir um fluxo médio, e um tempo maior quando o fluxo ultrapassar o valor normal em uma das vias.

Os dados com o número de veículos em cada uma das vias são enviados para o receptor, que irá ajustar o tempo de sinalização na próxima troca de sinal com base nesses dados recebidos. A Figura 7 mostra o diagrama em blocos do hardware do sistema.

Figura 7 - Diagrama em blocos do sistema.



Fonte: Autoria Própria, 2018.

3.1 Localização De Cada Dispositivo Do Sistema

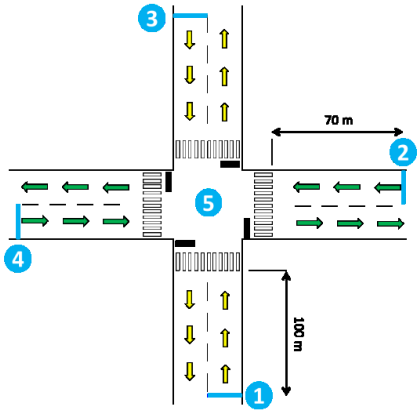
Com o auxilio da ferramenta de mapas e imagens de satélite *Google Maps*, observou-se um cruzamento com semáforo na cidade de Campo Mourão, para se obter informações sobre a distância entre o ponto central do cruzamento (ponto E em vermelho) e as ruas adjacentes, que dão acesso a ele os pontos A, B, C e D em amarelo, como mostrado na Figura 8.

Figura 8 - Imagem de satélite de um cruzamento de trânsito.

Fonte: Adaptado de Google Maps, 2018.

As distâncias entre os pontos, obtidas através da ferramenta de medição do Google Maps foi: 171 metros entre os pontos A e E, 88 metros entre B e E, 105 metros entre C e E, e por último, 96 metros entre os pontos D e E. Baseando-se nesses dados foi criado um modelo de cruzamento de trânsito para a aplicação do sistema de semáforo inteligente a ser desenvolvido, como mostrado na Figura 9.

Figura 9 - Modelo de cruzamento de trânsito.

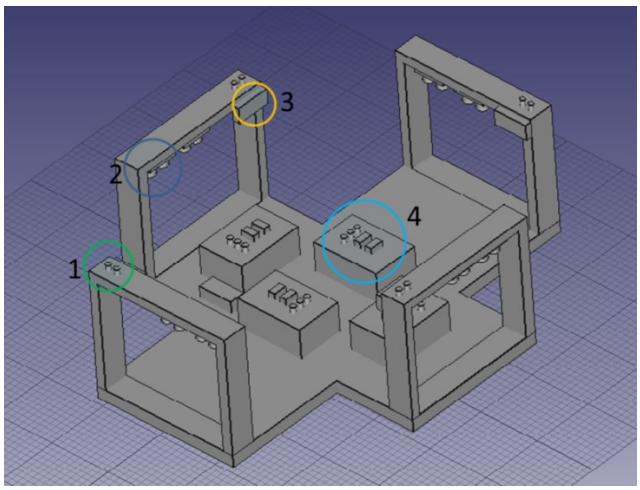


Fonte: Autoria própria, 2018.

As setas em verde representam o sentido do fluxo de veículos na via secundária (de menor fluxo) e as amarelas o sentido na via principal (de maior fluxo). Os círculos em azul numerados de 1 a 4 representam os dispositivos emissores, e barra em azul ao lado de cada circulo representa a área onde os sensores ultrassônicos irão identificar a passagem dos veículos. O círculo em azul com o número 5 representa o dispositivo receptor, e os 4 retângulos pretos próximos a ele são os semáforos.

Na Figura 10 é mostrado um desenho em 3D da maquete que foi construída.

Figura 10 – Desenho 3D da maquete.



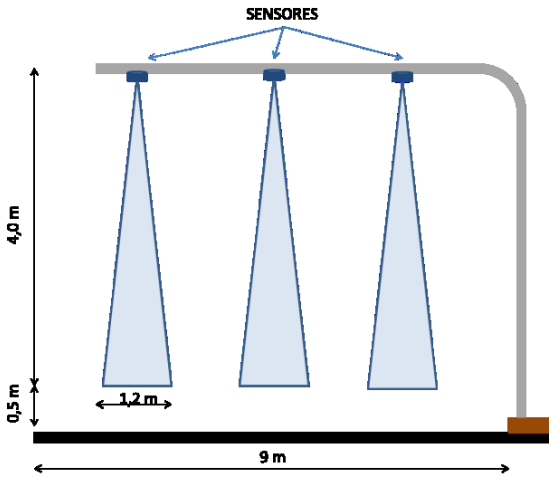
Fonte: Autoria própria, 2020.

O item 1 destacado em verde representa os LEDs de indicação do hardware do emissor que será explicado em uma seção especifica desse capítulo. O item 2 se refere aos sensores ultrassônicos. No item 3 é identificado o ESP32, que está presente do hardware dos dispositivos emissores e no receptor. E o item 4 representa as placas do circuito do semáforo.

3.2 Contagem Do Número De Veículos Usando Os Sensores Ultrassônicos

Usando a ferramenta de medição do *Google Maps*, pode-se medir a largura das ruas que formam o cruzamento da Figura 8, que possui aproximadamente 9 metros cada pista. Sendo assim, seria possível o tráfego de até 3 veículos, simultaneamente, um ao lado do outro. Então seria preciso 3 sensores ultrassônicos em cada via para que se consiga fazer de forma satisfatória a contagem do número de veículos, como pode ser visto na Figura 11.

Figura 11 - Esquema de posicionamento dos sensores ultrassônicos.



Fonte: Autoria própria, 2018.

A altura máxima permitida para veículos que trafegam em vias urbanas é de 4,40 metros (CONTRAN, 2006). Então seria necessário que os sensores estivessem posicionados a 4,5 metros de altura para conseguir detectar todos os veículos, dado

que os sensores HC-SR04, que foram escolhidos para o projeto tem um alcance de até 4 metros.

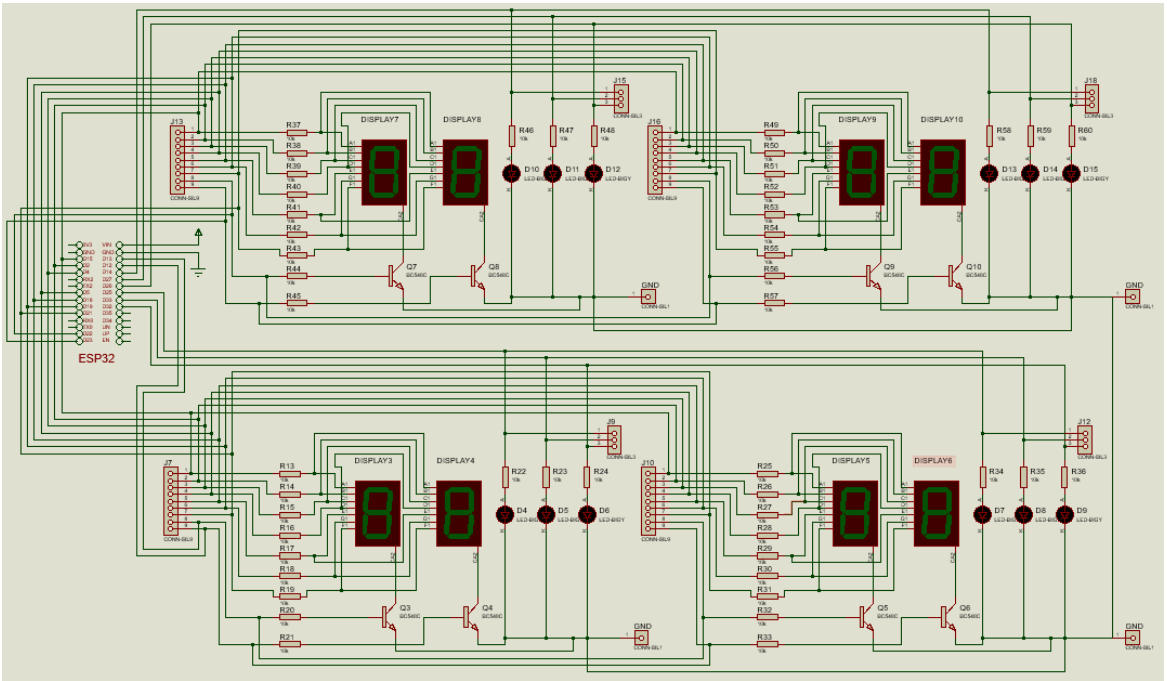
3.3 Hardware

O sistema possui um *hardware* que é utilizado nos 4 dispositivos que enviam os dados detectados pelos sensores (emissores), e o *hardware* que fica no centro do cruzamento (receptor) para exibir nos *displays* o tempo em cada via. A seguir será detalhado cada um deles.

3.3.1 Receptor

O *hardware* do receptor é composto por 8 *displays* de 7 segmentos e 12 LEDs, que são controlados por um ESP32, como pode ser visto na Figura 12.

Figura 12 - Hardware do Receptor.



Fonte: Autoria própria, 2020.

Os oito *displays* de 7 segmentos do circuito contêm quatro placas do semáforo, que podem exibir um número entre 0 e 99. Os displays estão ligados em paralelo, ou seja, mostram o mesmo número quando o sinal de nível lógico alto é enviado do ESP32. Para mostrar números diferentes nos *displays* da via principal e secundária, mesmo com eles ligados em paralelo é feito a multiplexação, ligando e desligando cada *display* através dos transistores. O software utilizado será explicado em detalhes em outro capítulo.

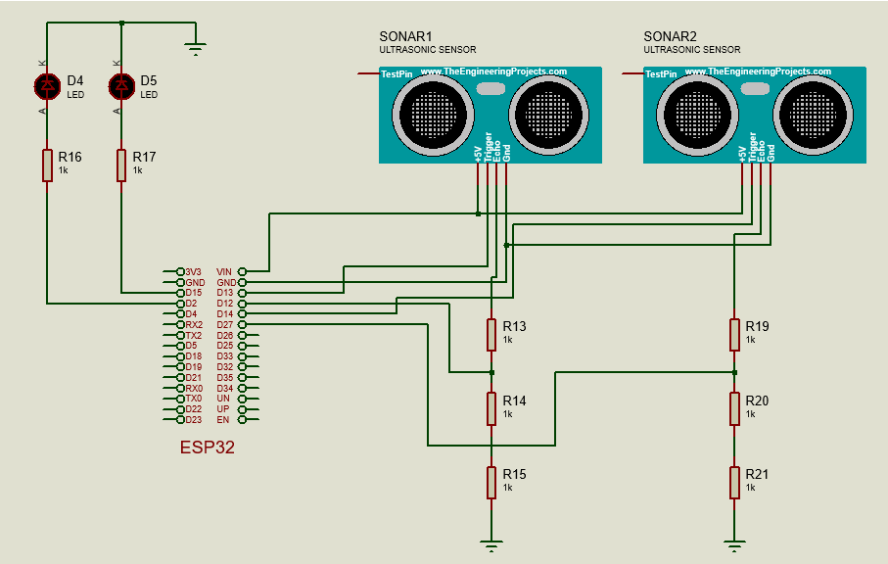
Cada placa possui 3 LEDs, nas cores: verde, vermelho e amarelo, que representam a sinalização padrão de um semáforo. Os LEDs das duas placas de cima do circuito estão em paralelo, pois representação a via principal e devem mostrar a mesma cor. Assim como acontece nas duas placas de baixo, que representam a via secundária do cruzamento. Dessa forma, são utilizados 6 pinos do ESP32 para fazer o acionamento dos LEDs.

Todos os resistores que foram utilizados no circuito possuem o mesmo valor de 1 KΩ.

3.3.2 Emissor

Responsável por detectar os carros que atravessam cada entrada do cruzamento, o hardware do receptor é composto por 2 sensores ultrassônicos HC-SR04, 2 LEDs para indicar quando o sensor detectar uma distância menor que o limiar definido e um ESP32 que irá enviar os dados até o dispositivo receptor. Foi colocado um sensor a menos no protótipo, pois a lógica do sistema é mesma utilizando 2 ou 3 sensores e a maquete teria que ter um tamanho maior caso fosse utilizado os 3 que foram mencionados anteriormente. Na Figura 13 é mostrado o circuito utilizado no protótipo.

Figura 13 - Esquema de ligação do Emissor.



Fonte: Autoria própria, 2020.

O sensor ultrassônico possui quatro pinos: TRIGGER, ECHO, +5V e GND. O primeiro tem função de disparar o envio da onda, o ECHO é responsável por detectar a onda que foi disparada pelo TRIGGER. Os outros dois pinos são para a alimentação do sensor.

Como pode ser observado na Figura 13, existe um divisor de tensão ligado ao pino ECHO, essa ligação foi necessária porque esse pino trabalha com uma tensão de 5V, porém os pinos do ESP32 suportam uma tensão de 3,3 V.

3.4 Software

Nesta seção será explicado o funcionamento do *software* do dispositivo emissor e receptor. Foram utilizadas as bibliotecas “esp_now.h”, “Wifi.h” e “Ultrasonic.h” no desenvolvimento dos códigos, escritos na linguagem de programação C para Arduino.

O Quadro 2 descreve a função das principais variáveis que fazem parte do software dos dispositivos.

Quadro 2 – Descrição das variáveis principais do software.

Nome da variável	Dispositivo	Descrição
flag1	Emissor	Habilita a leitura do sensor 1.
sensor1	Emissor	Distância lida pelo sensor 1 em cm.
value	Receptor	Identifica de qual via foi enviado os dados.
estagio	Receptor	Se refere aos 4 estágios que existem no ciclo semafórico.
valor_display	Receptor	Valor que aparece no display de 7 segmentos e é atualizado a cada um segundo.
carros_p	Receptor	Número de carros detectados na via principal.
carros_s	Receptor	Número de carros detectados na via secundária.
principal	Receptor	Informa se o display da via principal está ligado.
secundario	Receptor	Informa se o display da via secundária está ligado.

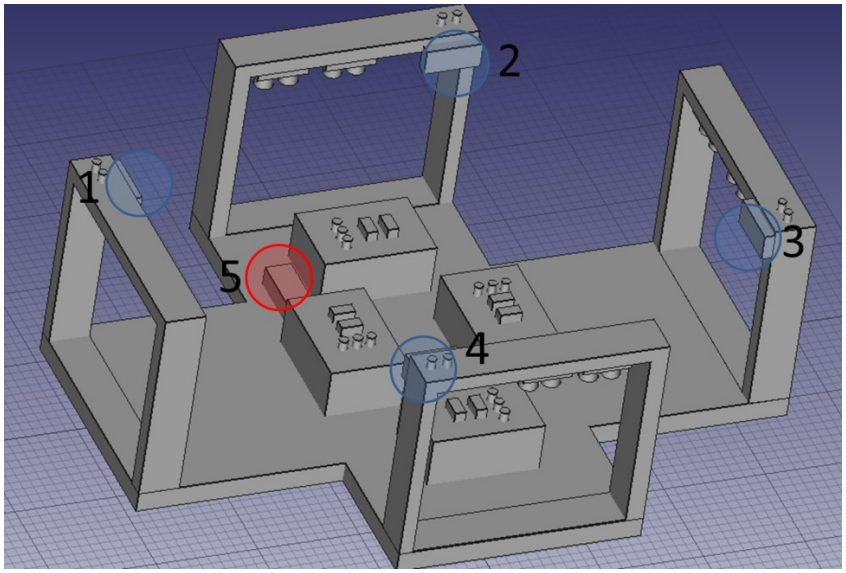
Fonte: Autoria própria, 2020.

3.4.1 Software Do Emissor

O código foi estruturado para fazer a leitura dos sensores ultrassônicos, em loop a cada 250 ms (valor escolhido devido ao bom tempo de resposta obtido nos testes realizados), e enviar um determinado valor para o receptor caso a distância que foi lida pelo sensor seja menor que 10 cm (esse valor foi definido para fazer testes na maquete), isso representa que um carro passou e foi detectado pelo sensor. O valor enviado ao receptor depende da posição de cada dispositivo emissor na maquete, se estiver posicionado na via principal (dispositivo 1 e 3) o valor enviado é 10 e se estiver na via secundária (2 e 4) o valor será 5. Dessa forma é possível saber qual emissor

enviou o dado, e consequentemente o número de carros em cada via. O posicionamento de cada dispositivo na maquete pode ser visto na Figura 14.

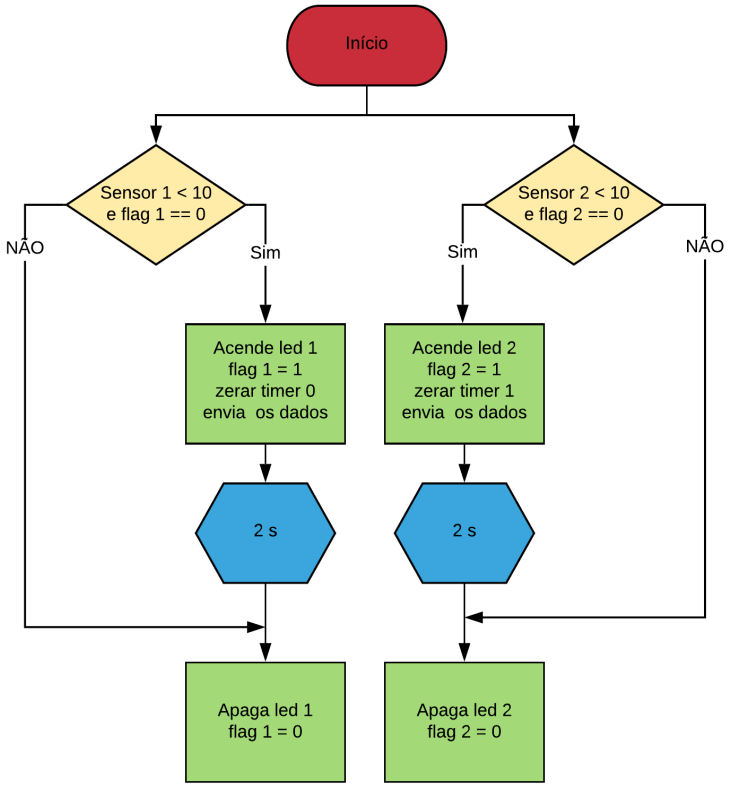
Figura 14 - Posicionamento dos dispositivos na maquete.



Fonte: Autoria própria, 2020.

No *software* (disponível no Apêndice 1), primeiramente foi utilizada a função “modeStation()”, para configurar o ESP32 como uma estação na topologia de rede P2P que foi explicada nos capítulos anteriores. Através da função “addPeer()” é feito o pareamento de cada emissor com o receptor, passando como parâmetro para a função o endereço MAC do ESP32 que irá receber os dados. Em seguida é configurada e habilitada a interrupção do TIMER 0 e TIMER 1, para dois segundos. Logo após, é chamada a função principal “readAndSend()”, que obtém os dados lidos dos sensores ultrassônicos para enviá-los ao receptor. O diagrama em blocos da lógica aplica na função “readAndSend()” é mostrado na Figura 15.

Figura 15 - Diagrama em blocos da função “readAndSend()”.



Fonte: Autoria própria, 2020.

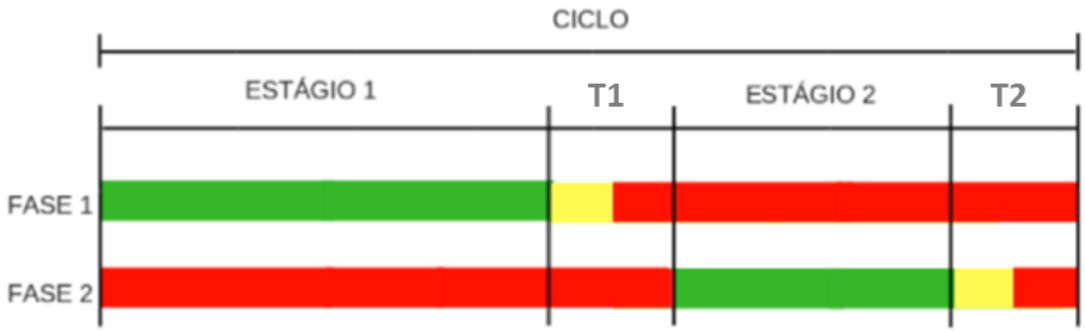
Analisando o diagrama em blocos e o código disponível no anexo 1, nota-se que foi utilizado uma *flag* dentro das condições lógicas que verificam se a distância obtida pelos sensores é menor que 10, essa *flag* é utilizada para permitir que a instrução *if* seja verdadeira e execute o bloco de comandos apenas quando a *flag* for igual a 0 e a distância lida pelo sensor menor que 10.

Quando essa expressão lógica é verdadeira, o LED é ligado para indicar que foi detectado um veiculo, o timer é zerado para garantir que a interrupção irá acontecer 2 segundos após essa instrução. A *flag* é definida com o valor 1, para que a mesma instrução *if* não seja verdadeira e bloco de código seja executado novamente antes de 2 segundos. Isso acontece porque é na interrupção do timer que o valor da flag volta a ser 0 e o LED é desligado.

3.4.2 Software Do Receptor

Este *software* é responsável por receber os dados enviados pelos emissores, interpretar os valores recebidos e exibir o tempo e a sinalização correta nos 4 painéis que compõe o semáforo. O ciclo semafórico que foi implementado no *software* é mostrado na Figura 16.

Figura 16 - Ciclo semafórico.



Fonte: Adaptado de Viana Dandara, 2019.

O estágio é definido pelo tempo que o sinal fica verde para cada uma das vias, no estágio 1 o sinal é verde para a via principal e vermelho para a secundária. T1 se refere ao estágio de transição, em que o sinal fica amarelo por 2 segundos e em seguida fica vermelho para a via principal, no segundo estágio os sinais são invertidos em relação ao primeiro estágio e no T2 ocorre o mesmo sinal de T1 mas agora na via secundária. A fase 1 se refere ao ciclo semafórico da via principal e a fase 2 ao ciclo da via secundária.

Na lógica desenvolvida no software, a duração do tempo de verde de cada estágio é alterada de acordo com os dados recebidos dos emissores, como pode ser visto no Quadro 3.

Quadro 3 – Relação entre o fluxo de veículos e o tempo de sinalização.

	Condição normal	Fluxo maior na via principal	Fluxo maior Na via secundária
Número de carros em cada via	P >= S e P <= S + 3	P > S + 4	P < S
Tempo de verde no estágio 1	40 s	60 s	27 s
Tempo de verde no estágio 2	30 s	20 s	40 s

Fonte: Autoria própria, 2020.

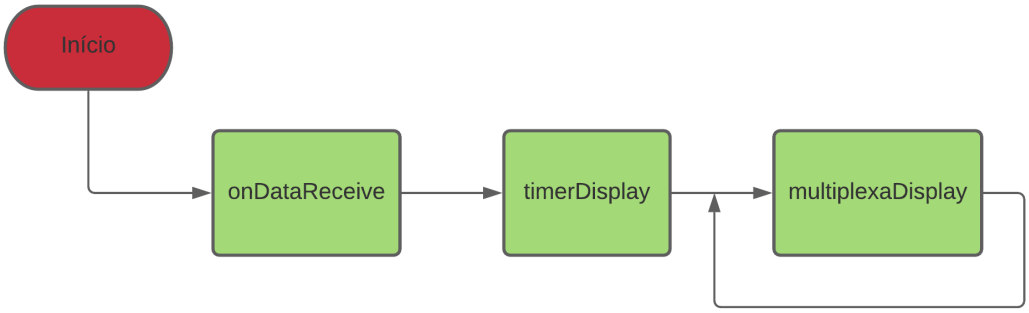
No Quadro 3, o “P” significa o número de carros na via principal e “S” na via secundária. A segunda coluna da esquerda para direita, representa a condição de fluxo normal de veículos esperado no cruzamento, havendo até 3 carros a mais na via principal em relação a via secundária, nessa condição o valor do tempo de sinal verde é de 40 s caso o próximo estágio seja o primeiro, se for o segundo estágio o tempo de verde é de 30 s.

A terceira coluna representa o caso onde há mais carros na via principal em relação ao que é esperado em condições normais. Nesse caso o algoritmo precisa aumentar o tempo de verde se o próximo for o estágio 1, ou diminuir caso for o estágio 2, para dar prioridade a via principal, que possui um maior fluxo de veículos no momento.

Na última coluna ocorre o caso em que há mais carros que o normal na via secundária, por isso o tempo de verde normal é diminuído se o próximo estágio for o 1 ou aumentado se for o estágio 2. Em ambos os casos da 3 e 4, o tempo de sinal verde é aumentado ou diminuído em cerca de 30% comparado ao tempo normal que foi definido na coluna 2.

A Figura 17 mostra um diagrama em blocos que mostra em um maior nível de abstração, as partes principais que compõem o *softwar*e do dispositivo.

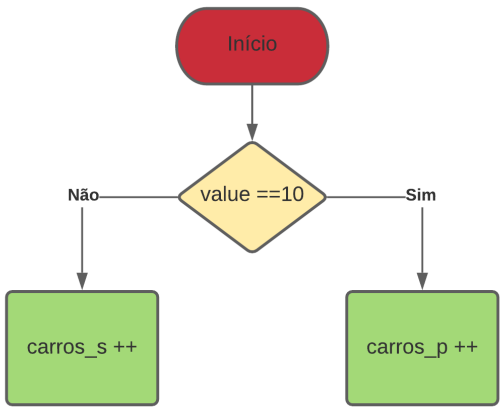
Figura 17 - Diagrama em blocos do software do dispositivo receptor.



Fonte: Autoria própria, 2020.

A função “onDataReceive” é responsável por receber e interpretar os dados enviados pelos emissores, “timerDisplay” é a função principal do *software*, ela é executada a cada 1 segundo via interrupção do *timer* 0, tendo o propósito de atualizar os valores de tempo que serão exibidos nos *displays* e controlar os LEDs de sinalização. O último bloco do diagrama “multiplexaDisplay”, utiliza os dados gerados no “timerDisplay” para inserir os valores nos *displays* de 7 segmentos, através da técnica de multiplexação. Cada um dos blocos mostrados no diagrama acima será explicado de forma detalhada a seguir.

Figura 18 - Diagrama em blocos da função “onDataReceive”.



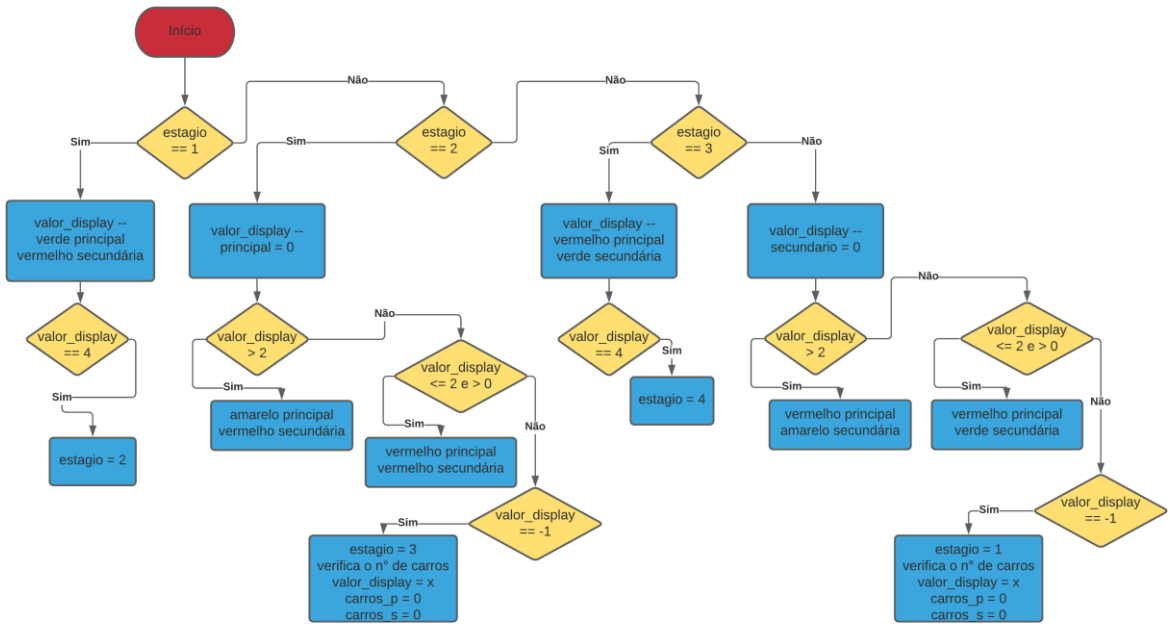
Fonte: Autoria própria, 2020.

Na figura 18 pode ser visto o diagrama em blocos da função “onDataReceive” . Essa é uma função nativa da biblioteca do protocolo ESP-NOW, quando um dado é recebido ela é chamada e passa como parâmetro o endereço MAC do dispositivo que enviou e o valor enviado que é um inteiro de 8 bits sem sinal.

No software (Apêndice 2) o valor recebido está na variável “value”, se o valor for igual a 10, significa que foi enviado por algum dos dois dispositivos da via principal, então é somado mais 1 no valor da variável global “carros_p “ . E se o valor for diferente 10, significa que foi enviado por algum dos dispositivos da via secundária, e seguindo a mesma lógica será somado mais 1 no valor da variável global “carros_s”.

A próxima função a ser explicada será “timerDisplay”, seu diagrama em blocos é mostrado na Figura 19.

Figura 19 - Diagrama em blocos da função “timerDisplay”.



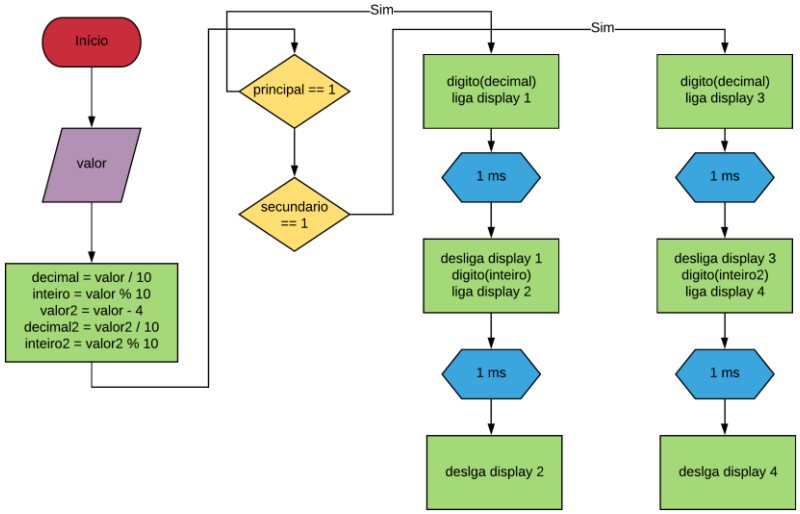
Fonte: Autoria própria, 2020.

Essa é a função chamada a cada um segundo via interrupção do TIMER 0. A variável “estagio” é inicializada com o valor 1 no inicio do código e por isso quando a função é chamada pela primeira vez executa essa condição, que decrementa o valor da variável “valor_display” que inicia com o valor 44, referente aos 40 s de verde no

estágio 1 mais 4 segundos do estágio de transição T1 e aciona o sinal vermelho na via principal e o verde na secundária, essa rotina vai sendo executada até a variável “valor_display” chegar no valor igual a 4, quando isso ocorre o valor do estágio é alterado para 2. No segundo estágio o valor do *display* também é decrementado e a variável “principal” é colocada em 0 para fazer com que o *display* da via principal seja desligado durante a fase de transição do ciclo semafórico, que mantém o sinal amarelo na via principal e vermelho na secundária durante 2 segundos e depois vermelho nas duas vias pelo mesmo tempo. Após essa rotina o valor do *display* estará em 0 e nesse momento é verificado o valor das variáveis globais “carros_p” e “carros_s”, para saber o número de carros em cada via e ajustar o tempo que irá aparecer no *display* no próximo estagio do ciclo semafórico e a variável “estagio” é alterada para 3.

O terceiro estágio segue a mesma lógica aplicada no estágio 1 e o estágio 4 que ocorre quando a condição “estagio == 3” é negativa, segue a mesma lógica do estágio 2. Na Figura 20 é apresentado o digrama da última função que compõem o *software*.

Figura 20 - Diagrama em blocos da função “multiplexaDisplay”.



Fonte: Autoria própria, 2020.

A função “multiplexaDisplay” é executada em *loop* e é responsável por pegar os valor da variável “valor_display” e colocar os valores corretos nos *displays* de cada via, usando para isso a técnica de multiplexação.

O valor que deve ser exibido é recebido como parâmetro na função, a parte inteira e decimal é salva em outra variável. A variável “valor2” representa o número que irá aparecer na outra via do semáforo e é subtraído 4 desse número pois esse é o tempo de transição entre os estágios do ciclo semafórico.

Após ter os valores separados é verificado se o *display* da via principal será ligado, caso seja, a parte decimal do número é passada como parâmetro para a função “digito”, que vai ligar os segmentos corretos do *display* 1 para formar o número. Após isso é aplicado um *delay* de 1 ms e em seguida o *display* 1 é desligado, a parte inteira do número é passada para a função “digito” e o *display* 2 é ligado, depois dessa etapa é aplicado um *delay* de 1 ms novamente e por último o *display* 2 é desligado.

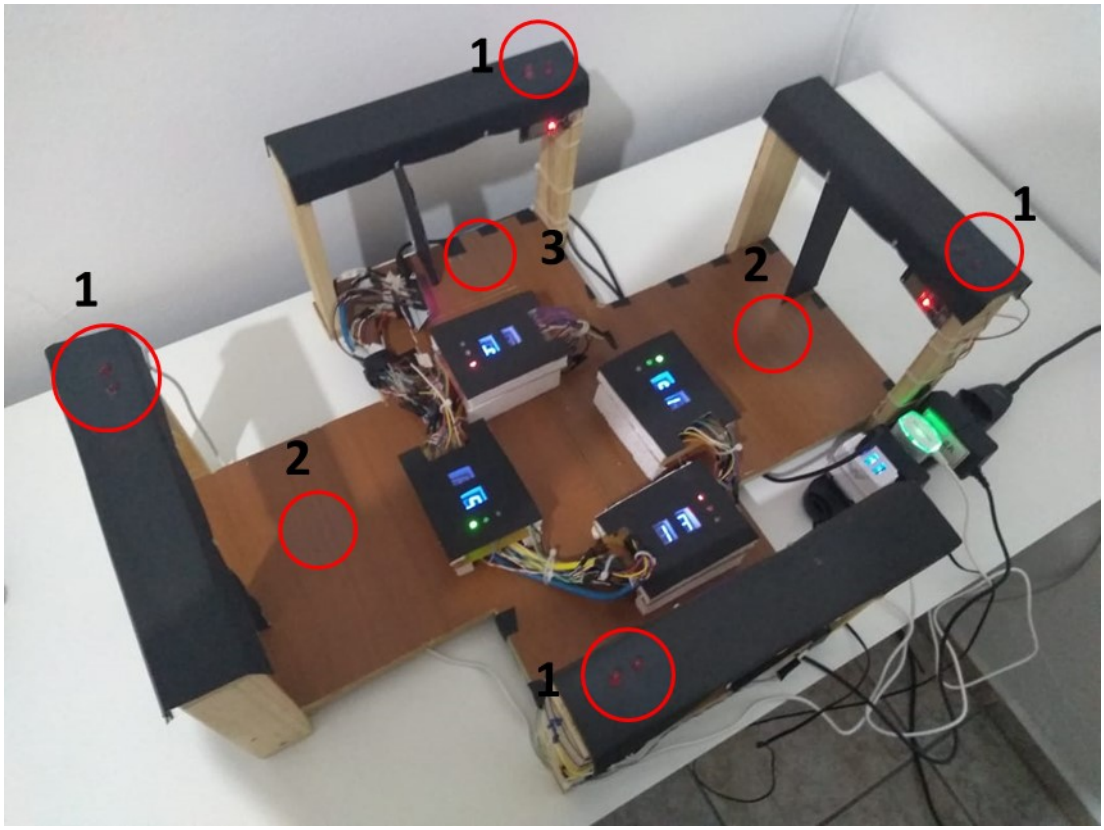
A lógica se repete para o *display* da via secundária, é dessa forma que é feita a multiplexação dos *displays*, que visualmente aparentam estarem todos ligados ao mesmo tempo.

4. RESULTADOS

Nesse capitulo será mostrado os resultados obtidos através dos testes realizados utilizando o protótipo que foi desenvolvido.

A Figura 21 mostra a maquete que foi construída para realizar os testes do protótipo. A via principal (2) tem um comprimento maior que a via secundária (3), como pode ser visto na imagem.

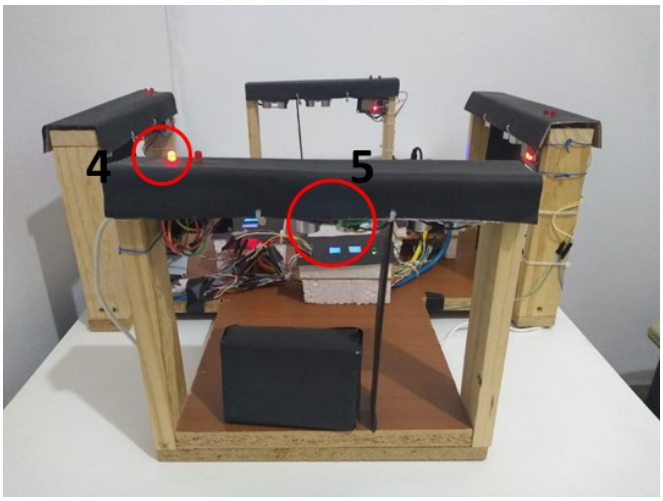
Figura 21 – Protótipo.



Fonte: Autoria própria, 2020.

Para realizar os testes foi colocado um objeto em baixo do sensor para simular que um veículo foi detectado, quando isso acontece o LED de indicação (1) é ligado durante 2 segundos. Esse processo é mostrado na Figura 22, onde o LED esquerdo (4) está aceso, sinalizando que o sensor esquerdo (5) detectou um objeto.

Figura 22 – Sensor detectando um objeto.

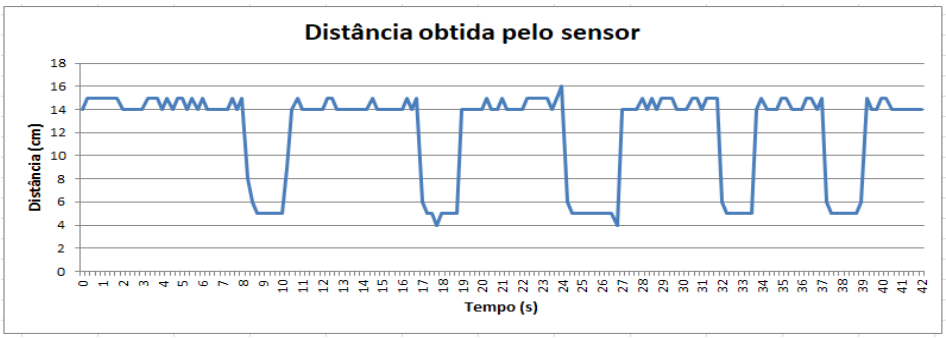


Fonte: Autoria própria, 2020.

Como os sensores ultrassônicos estão muito próximos e direcionados para a mesma superfície, ocorre interferência entre eles devido ao disparo do TRIGGER de um sensor ser detectado pelo outro, causando um erro de leitura na distância. Devido a isso foi necessário utilizar um separador entre os sensores para evitar a interferência. Se o sistema estivesse instalado em um cruzamento real, esse problema de interferência não aconteceria por que os sensores estariam instalados com um espaçamento muito maior entre eles.

Na figura 23 é mostrado como ocorre a detecção de um objeto pelo sensor ultrassônico.

Figura 23 – Gráfico do sensor ultrassônico.

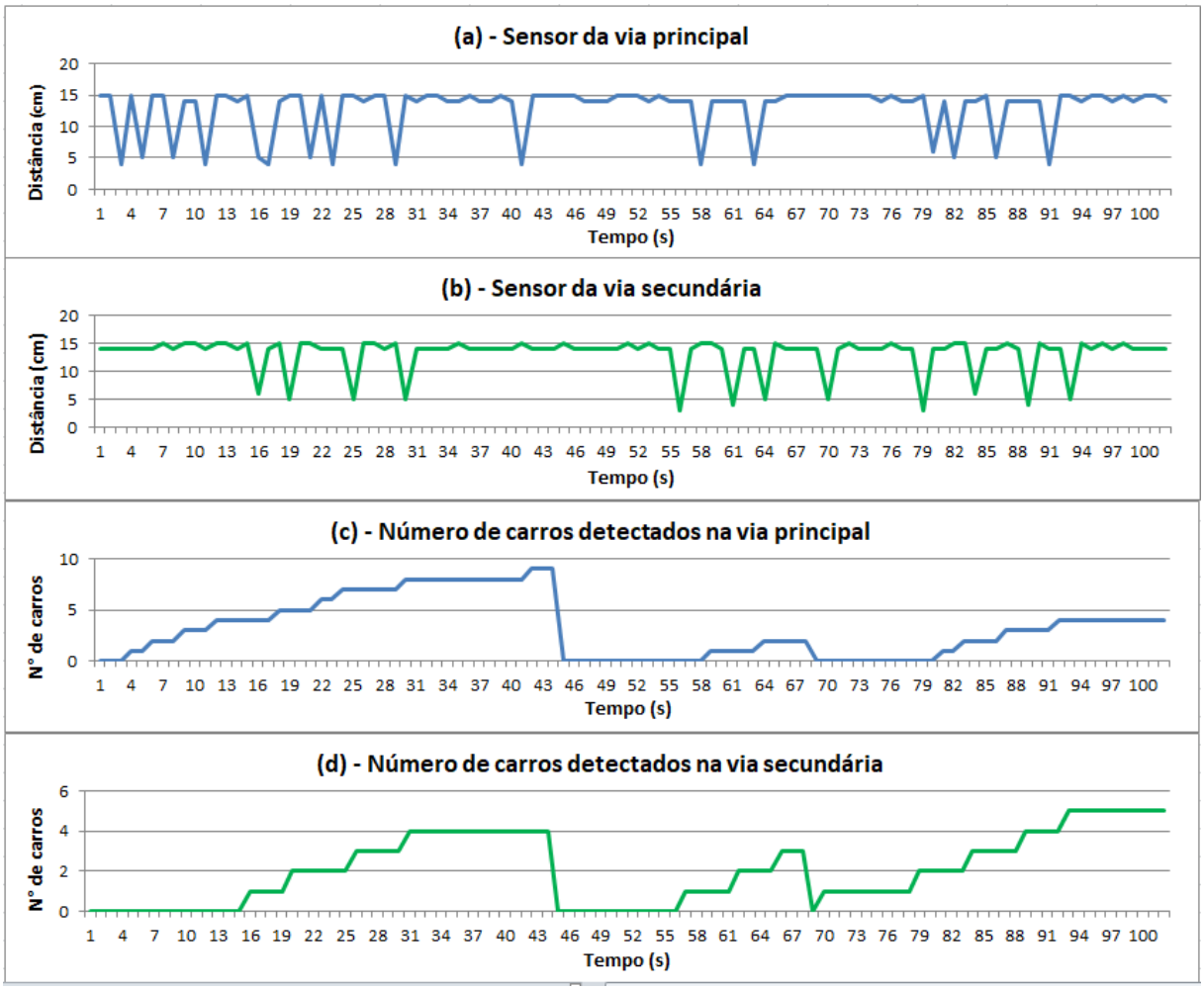


Fonte: Autoria própria, 2020.

Os dados foram obtidos através do monitor serial do IDE Arduino e o gráfico foi feito com o Microsoft Excel. No teste, um objeto foi colocado em baixo do sensor á uma distância de aproximadamente 5 cm, durante poucos segundos. Esse processo foi repetido 5 vezes, como pode ser percebido pelos vales formados no gráfico.

A relação entre o objeto detectado pelos sensores ultrassônicos, com o envio dessa informação para o receptor e o processamento desses dados é mostrado na Figura 24.

Figura 24 – Gráfico comparativo entre emissores e receptor.



Fonte: Autoria própria, 2020.

Os dois gráficos na parte de cima da imagem (a e b) mostram a detecção de objetos de uma simulação que foi feita na maquete. Foi utilizado um sensor da via principal e outro da via secundária e os dados foram obtidos da mesma forma que no teste anterior. Os gráficos c e d mostram o valor das variáveis “carros_p” e “carros_s” do dispositivo receptor, que representam o número de carros em cada via, esses dados foram são enviados pelos dispositivos emissores após a detecção dos objetos pelos sensores ultrassônicos.

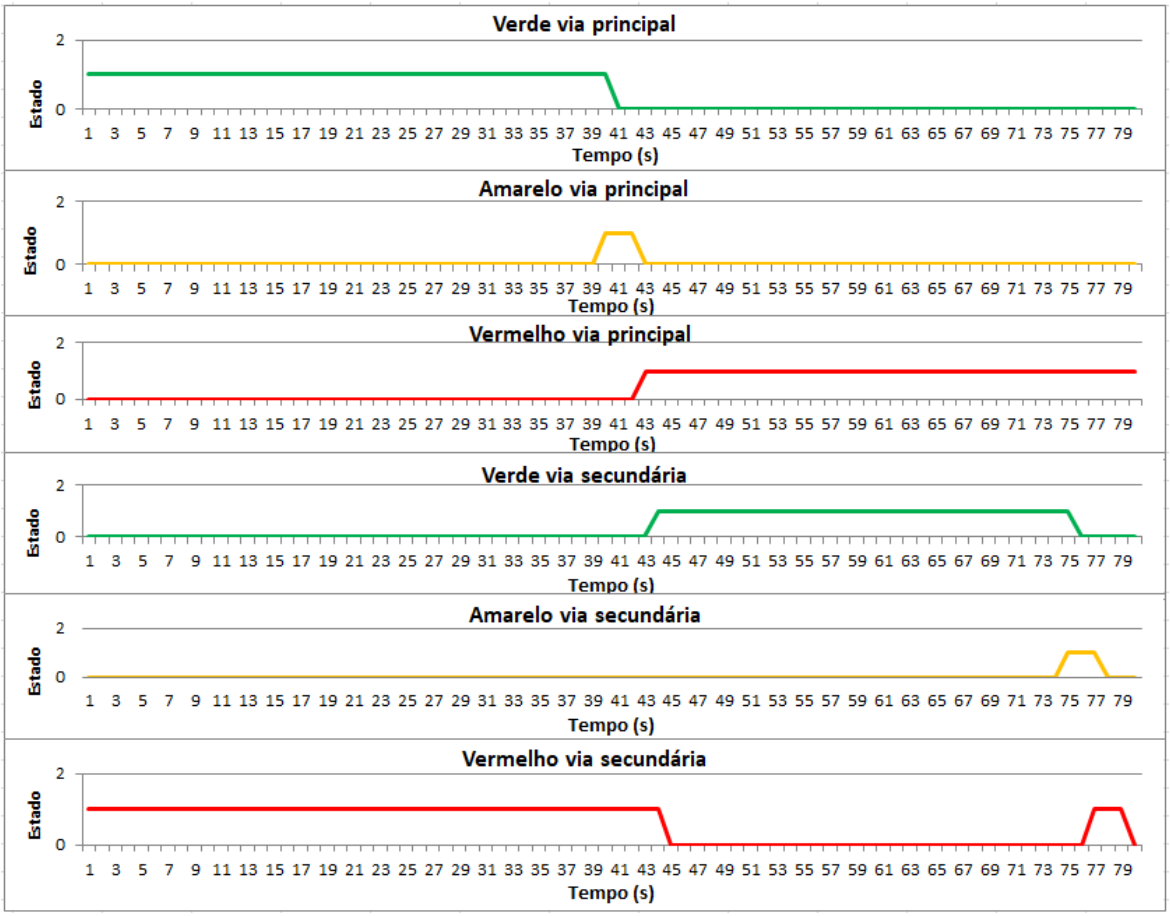
Quando o receptor é ligado, o primeiro estágio com tempo de verde de 40 segundos na via principal ocorre, e durante esse tempo e o tempo de transição de um estágio para o outro, que é de 4 segundos, os dados enviados pelos emissores são lidos e armazenados nas variaves “carros_p” e “carros_s”. Possuindo esses dados do número de carros em cada via, a lógica do sistema é aplicada para definir qual vai ser o tempo de sinal verde no próximo estágio, e então o valor dessas variáveis é zerado e processo se repete.

O momento em que o valor das variáveis são zeradas pode ser visto nos gráficos no tempo 44 segundos, até esse instante o número de carros detectados em cada via foi de 9 carros na via principal e 4 carros na via secundária, como pode ser visto nos gráficos a e b, contando o número de vales em cada gráfico até o tempo de 44 segundos.

Então é dessa forma que acontece a detecção dos objetos pelos emissores, que enviam o valor 1 para o dispositivo receptor quando o valor lido pelo sensor ultrassônico é menor que 10 cm, indicando que um carro foi detectado. Dessa forma o receptor consegue somar todos os valores que foram recebidos para comparar a quantidade de carros em cada via e ajustar o tempo no próximo estágio do ciclo semafórico.

A seguir, na Figura 25 é mostrado os sinais de verde, vermelho e amarelo dos semáforos de cada uma das vias.

Figura 25 – Gráfico dos sinais do semáforo.

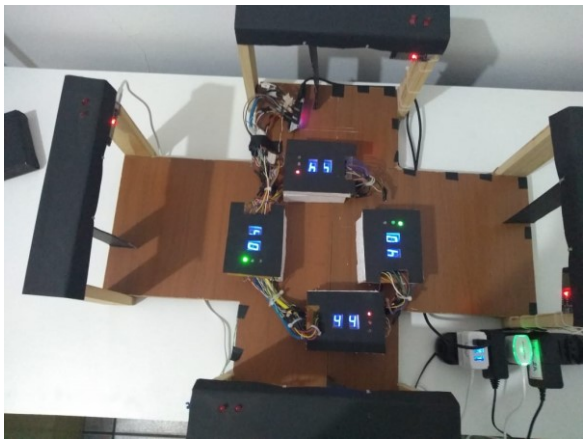


Fonte: Autoria própria, 2020.

Os dados foram coletados da mesma forma que nos dois gráficos anteriores. Como pode ser observado na imagem a cima, o primeiro estágio inicia com um tempo de verde igual a 40 segundos na via principal, em seguida ocorre a transição de 4 segundos com os sinais amarelo e vermelho, o segundo estágio tem 30 segundos de tempo de verde, pois nesse teste não foi colocado nenhum objeto para ser detectado pelos sensores, sendo essa a condição normal de fluxo de veículos do Quadro 3.

A seguir serão mostrados outros testes realizados no protótipo com o objetivo de demonstrar todas as condições de fluxo de veículos do Quadro 3.

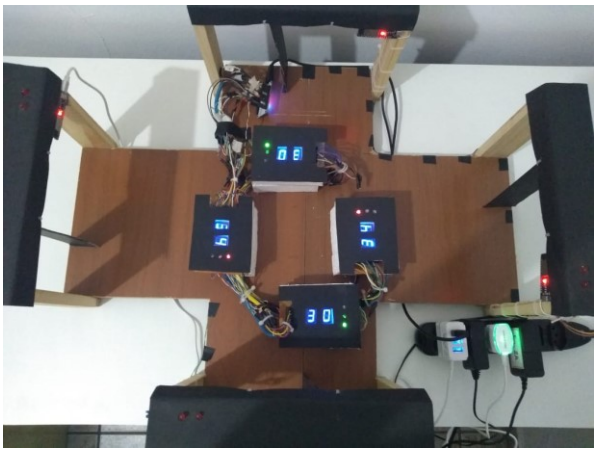
Figura 26 – Tempo de verde igual a 40 s na via principal.



Fonte: Autoria própria, 2020.

A Figura 26 mostra o valor inicial exibido no display quando o sistema é ligado, esse é o valor definido no Quadro 3, sendo 40 s de tempo de verde na via principal no caso em que o cruzamento apresenta uma condição normal de fluxo de veículos, lembrando que esse é o estágio 1 do ciclo semafórico. Durante esse estágio não foi colocado nenhum objeto para ser detectado pelo sensor, para testar a condição da segunda coluna do Quadro 3, que possui o valor de 30 s de tempo de verde na via secundária no segundo estágio, como é mostrado na Figura 27.

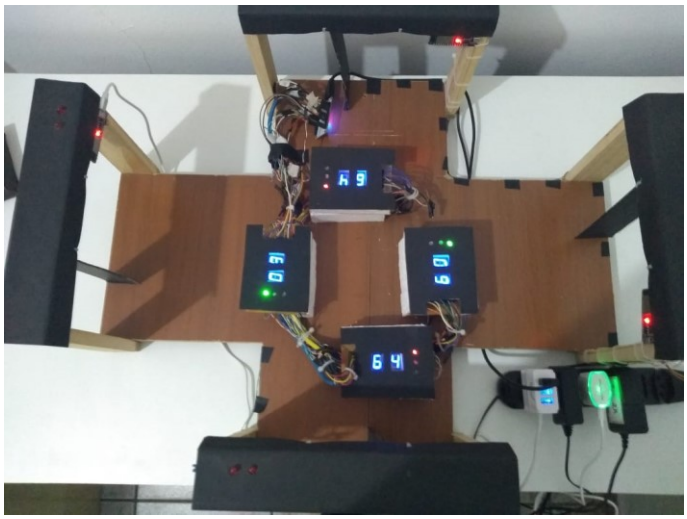
Figura 27 - Tempo de verde igual a 30 s na via secundária.



Fonte: Autoria própria, 2020.

A próxima condição a ser testada é a segunda coluna da tabela, que representa os casos onde existe maior fluxo de veículos na via principal. Para fazer o teste foi mantido um objeto a baixo do sensor da via principal durante 10 segundos, para contar como 5 veículos detectados nessa via e nenhum veiculo na via secundária. Esse teste foi feito durante o segundo estágio, então após o tempo de transição para o primeiro estágio o valor exibido no *display* foi de 60 s. O resultado é mostrado na Figura 28.

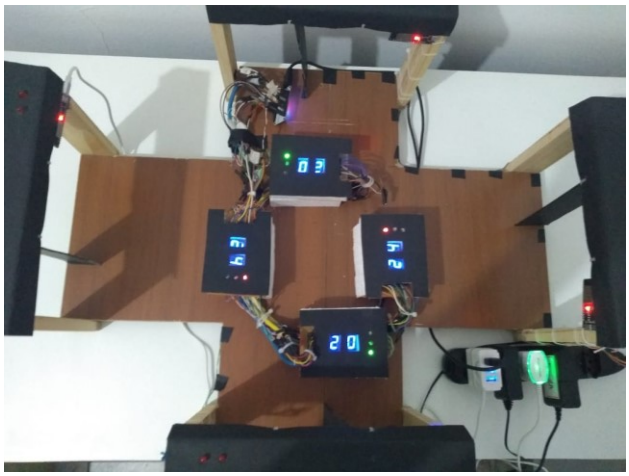
Figura 28 - Tempo de verde igual a 60 s na via principal.



Fonte: Autoria própria, 2020.

Durante o primeiro estágio que foi iniciado realizou-se um novo teste, mantendo o objeto na direção do sensor da via principal durante 10 s novamente. E após a transição para o estágio 2 o tempo de verde mostrado na via secundária diminui para 20 s, como esperado. O resultado é mostrado na Figura 29.

Figura 29 - Tempo de verde igual a 20 s na via secundária.



Fonte: Autoria própria, 2020.

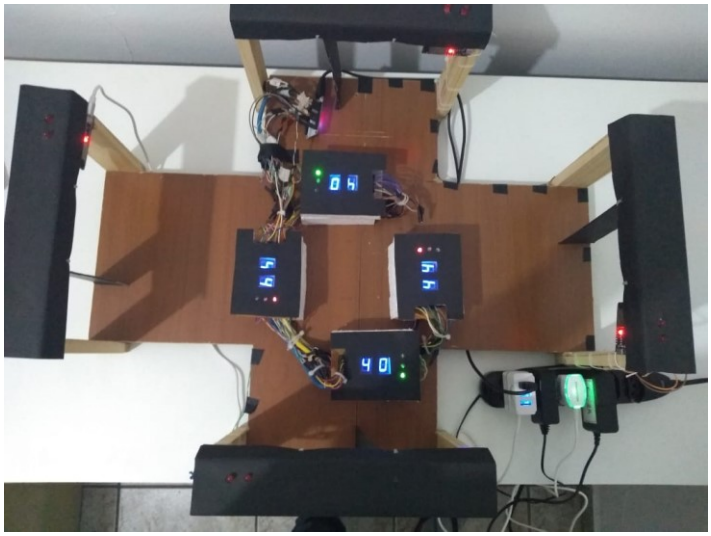
Os últimos testes a serem feitos são os da condição onde existem mais veículos na via secundária. Para fazer o primeiro teste, foi mantido um objeto para ser detectado pelo sensor da via secundária durante 6 segundos, o que representa 3 veículos passando por essa via e nenhum na pela outra. Esse teste foi feito durante o segundo estágio, após a transição para o primeiro, o tempo de verde mostrado no *display* foi de 27 s, como definido no Quadro 3. O resultado é mostrado na Figura 30.

Figura 30 - Tempo de verde igual a 27 s na via principal.

Fonte: Autoria própria, 2020.

Para finalizar os testes com todas as condições definidas no Quadro 3, manteve-se o objeto sendo detectado por um sensor da via secundária durante 6 segundos, como foi feito anteriormente, mas dessa vez durante o primeiro estágio. Após a transição para o estágio 2, o tempo de verde na via secundária aumentou para 40 segundos em relação ao tempo normal, pois existem mais veículos passando por essa via e por isso o tempo de verde deve ser aumentado. A Figura 31 mostra esse resultado exibido no display.

Figura 31 - Tempo de verde igual a 40 s na via secundária.



Fonte: Autoria própria, 2020.

5. CONCLUSÃO

Vem aumentando nos últimos anos a adesão de novas tecnologias para melhorar a qualidade de vida das pessoas nas cidades. Se tratando do trânsito onde as pessoas passam algumas horas por dia em grandes centros urbanos, é muito importante que existam ações para diminuir o tempo gasto para se locomover de local para outro.

Devido a isso, foi desenvolvido nesse trabalho uma ferramenta que pode ser utilizada para diminuir o tempo gasto nos cruzamentos de trânsito que possuem semáforos. Sendo uma solução que pode ser implementada facilmente, devido ao baixo custo dos componentes utilizados no projeto, em relação a sistemas que tem a mesma finalidade e utilizam câmeras para monitorar o tráfego de veículos.

O trabalho desenvolvido também serve de base para outros trabalhos que venham utilizar comunicação sem fio através do protocolo ESP-NOW, nas mais diversas áreas de aplicação, uma vez que, na data de criação deste trabalho existiam escassas fontes de informação disponíveis na internet sobre o protocolo e a sua implementação em projetos reais.

Durante o a construção do protótipo uma das etapas que levou mais tempo, foi o processo de montagem e soldagem das placas dos displays, devido a dificuldade de interligar as quatro placas em um único circuito eletrônico.

Os resultados obtidos nos testes foram satisfatórios e atenderam as expectativas do que foi planejado. Dessa forma pode-se concluir que o sistema de semáforo inteligente desenvolvido, foi capaz de fazer a leitura dos sensores, transmitir os dados e exibir os resultados como foi planejado.

REFERÊNCIAS

ALBUQUERQUE, R. **Qual a origem do primeiro semáforo de trânsito do mundo**. 2017. Disponível em: https://www.nexojornal.com.br/expresso/2017/10/20/Qual-a-origem-do-primeiro-sem%C3%A1foro-de-tr%C3%A2nsito-no-mundo. Acesso em: 20 set. 2018.

CONTRAN. **Manual brasileiro de sinalização de trânsito**. 2014.

CIRINEU, S.A. **Controlador de tráfego: semáforo inteligente**. 2006. 96 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) - Centro Universitário de Brasília - UNICEUB, Brasília, 2006.

SABURO, H.Y. **Projeto de controlador inteligente para semáforo**. 2008. 26 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Mecânica) - Universidade Estadual de Campinas – UNICAMP, Campinas, 2008.

DETRAN-MS. **Furar o sinal: um ato imprudente com sérias consequências**. 2017. Disponível em: http://www.detran.ms.gov.br/furar-o-sinal-um-ato-imprudente-com-serias-consequencias. Acesso em: 21 set. 2018.

CORREIO BRASILIENZE. **Frota brasileira cresce, mas idade média dos veículos ainda é alta**. 2017. Disponível em: https://www.correiobraziliense.com.br/app/noticia/economia/2018/04/13/internas_economia,673326/frota-brasileira-cresce-mas-idade-media-dos-veiculos-ainda-e-alta.shtm. Acessado em 21 set. 2018.

ESTADÃO. **Primeiro semáforo de SP foi instalado no Brás**. 2013. Disponível em: https://sao-paulo.estadao.com.br/noticias/geral,primeiro-semaforo-de-sp-foi-instalado-no-bras-imp-,1005848. Acesso em 25 set. 2018.

DENATRAN. **Manual de semáforos**. 2.ed. Brasília, 1984.

LOBUR, J.; NULL, L. **Arquitetura e organização de computadores**. 3. ed. Sudbury, 2010.

POZZEBOM, R. **O que é wireless e como funciona?**. 2012. Disponível em: https://www.oficinadanet.com.br/post/2961-o-que-e-wireless-e-como-funciona. Acesso em 2 out. 2018.

TANEMBAUM, A. S. **Redes de computadores**. 5. Ed. Pearson, 2011.

ESPERIDIÃO, H. **Redes sem fio**. 36 f. Notas de aula.

KOYANAGI, F. **ESP32 com protocolo ESP-Now**. 2018. Disponível em: https://www.fernandok.com/2018/03/esp32-com-protocolo-esp-now.html. Acesso em 10 out. 2018.

ESP-NOW. **User guide**. Versão 1.0. Estados Unidos. 2016. Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html. Acesso em: 30 out. 2018

SANTOS, C. E. P. **Um sistema P2P de gerenciamento de dados com conectividade baseada em semântica**. 2007. 123 f. Tese (Pós-Graduação em Ciências da Computação) - Universidade Federal de Pernambuco, Recife, 2007.

BRAGA, N.C. **Redes sem fio e o padrão IEEE 802.11 (TEL026)**. S/D. Disponível em: http://www.newtoncbraga.com.br/index.php/telecom/2364-tel026. Acesso em: 30 out. 2018.

SOARES, L.F.G.; LEMOS, G.; COLCHER, S. **Redes de computadores: das LANs MANs e WANs ás redes ATM**. 2. ed. Rio de Janeiro, 1995.

UZEDA, L.G.G. **Redes 802.11 (camada de enlace)**. S/D. Disponível em: https://www.gta.ufrj.br/grad/01_2/802-mac/index.html. Acesso em: 20 out. 2018.

SOUSA, J.B.C; Lugli, A. B. **Estudo de sensores ultrassônicos e suas aplicações**. *In*: SEMINÁRIO DE REDES E SISTEMAS DE TELECOMUNICAÇÕES INSTITUTO NACIONAL E TELECOMUNICAÇÕES, 1, 2014, Santa Rita do Sapucaí. I SRST - Seminário de redes e sistemas de telecomunicações instituto nacional e telecomunicações. Santa Rita do Sapucaí: INATEL, 2014. ISSN 2358-1913.

BRAGA, N.C. **Como funcionam os sensores ultrassônicos**. 2012. Disponível em: http://www.newtoncbraga.com.br/index.php/como-funciona/5273-art691. Acesso em: 20 out. 2018.

VIANA, D. **Programação semafórica: entenda como funciona**. Guia da Engenharia, 2019. Disponível em: https://www.guiadaengenharia.com/programacao-semaforo. Acesso em: 20 ago. 2020.

CASTRO, C.P.C.S. **Cidades inteligentes: definições, estudos de casos e modelagem de iniciativa inteligente em Volta Redonda**. 2018. 75 f. Trabalho de Conclusão de Curso – Engenharia de Telecomunicações, Universidade Federal Fluminense, Niterói. 2018. Disponível em: https://app.uff.br/riuff/bitstream/1/8066/1/TCC-%20Carolina%20Portela.pdf. Acesso em: 16 nov. 2020.

SOUZA, A. L. M. **Um estudo sobre o conceito de cidades inteligentes na região metropolitana do rio de janeiro**. 2017. 73 f. Trabalho de Conclusão de Curso – Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro. 2017. Disponível em: http://monografias.poli.ufrj.br/monografias/monopoli10021088.pdf. Acesso em: 18 nov. 2020.

SÓ, P. L. S. **Enfrentando os desafios da mobilidade urbana: um estudo de caso na região metropolitana da grande florianópolis**. 2017. 104 f. Trabalho de

Conclusão de Curso – Administração, Universidade Federal de Santa Catarina, Florianópolis. 2017. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/181672/TC_Paula%20Lunelli.pdf?sequence=1&isAllowed=y. Acesso em: 18 nov. 2020.

CARVALHO, J. R. **Agileasy: uma proposta de plataforma de agenda que auxilia a otimização do tempo dos usuários em função da mobilidade urbana**. 2019. 63 f. Trabalho de Conclusão de Curso – Sistemas e Mídias Digitais, Universidade Federal do Ceará, Fortaleza. 2019. Disponível em: http://repositorio.ufc.br/bitstream/riufc/45933/3/2019_tcc_jrcarvalho.pdf. Acesso em: 18 nov. 2020.

APENDICE I – CÓDIGO DO DISPOSITIVO EMISSOR

```
#include <esp_now.h>
#include <WiFi.h>
#include<Ultrasonic.h>
#define INTERVALO 250 //(ms)

//variável responsável por armazenar a distância lida pelo sensor ultrassônico
unsigned int distancia = 0;
unsigned int distancia2 = 0;

//flags do timer
unsigned int t0 = 0;  // se a flag for 0 será feita a leitura do sensor
unsigned int t1 = 0;

//conexão dos pinos para o sensor ultrasonico
#define PIN_TRIGGER   4
#define PIN_ECHO      5
#define PIN_TRIGGER2   22
#define PIN_ECHO2      23
Ultrasonic ultrasonic(PIN_TRIGGER, PIN_ECHO);
Ultrasonic ultrasonic2(PIN_TRIGGER2, PIN_ECHO2);

//Mac Address do peer para o qual serão enviado os dados
uint8_t peerMacAddress[] = {0x80, 0x7D, 0x3A, 0xF3, 0x35, 0xF8};  // Endereço
MAC do Master

hw_timer_t *timer = NULL;
hw_timer_t *timer1 = NULL;

//definindo que será usado o canal 1
#define CHANNEL 1
esp_now_peer_info_t peer;
//Inicializando o modo station
void modeStation(){
```

```
    WiFi.mode(WIFI_STA);
    Serial.print("Mac Address in Station: ");
    Serial.println(WiFi.macAddress());
}
//Função que inicia o ESPNow
void InitESPNow() {
    if (esp_now_init() == ESP_OK) {
        Serial.println("ESPNow Init Success");
    }
    else {
        Serial.println("ESPNow Init Failed");
        ESP.restart();
    }
}
//Função que faz o pareamento atraves do endereço MAC
void addPeer(uint8_t *peerMacAddress){
    peer.channel = CHANNEL;
    //0 para não usar criptografia ou 1 para usar
    peer.encrypt = 0;
    //Copia o endereço do array para a estrutura
    memcpy(peer.peer_addr, peerMacAddress, 6);
    esp_now_add_peer(&peer);
}

//Função que envia os dados
void send(const uint8_t *value, uint8_t *peerMacAddress){
    esp_err_t result = esp_now_send(peerMacAddress, value, sizeof(value));
    Serial.print("Send Status: ");
    //Se o envio foi bem sucedido
    if (result == ESP_OK) {
        Serial.println("Success");
    }
    //Se aconteceu algum erro no envio
```

```
    else {
        Serial.println("Error");
    }
}
void timerDisplay(){
    digitalWrite(13,LOW);
    t0 = 0;
}
void timerDisplay1(){
    digitalWrite(12,LOW);
    t1 = 0;
}
void setup() {
  Serial.begin(115200);
  modeStation();
  InitESPNow();

  //Faz o pareamento com o outro dispositivo
  addPeer(peerMacAddress);

  //Registra o callback sobre o status do envio
  esp_now_register_send_cb(OnDataSent);

  pinMode(PIN, INPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  digitalWrite(12,LOW);
  digitalWrite(13,LOW);
  timer = timerBegin(0, 80, true);                  // seta o timer0
  timerAttachInterrupt(timer, &timerDisplay, true);
  timerAlarmWrite(timer, 2000000, true);
  timerAlarmEnable(timer);
  timer1 = timerBegin(1, 80, true);                  // seta o timer1
```

```
  timerAttachInterrupt(timer1, &timerDisplay1, true);
  timerAlarmWrite(timer1, 2000000, true);
  timerAlarmEnable(timer1);

  readAndSend();
}
void readAndSend(){
  distancia = getDistance();
  distancia2 = getDistance2();
  if(distancia2 <= 1){
    distancia2 = 15;
  }
  if(distancia == 0){
    distancia = 15;
  }
   Serial.println("Distancia 1: ");
   Serial.println(distancia);
   Serial.println("Distancia 2: ");
   Serial.println(distancia2);
 delay(INTERVALO);
  uint8_t value2 = 0;
  if ((distancia <= 10 ) and (t0 == 0) ){
    value2 = 5;
    digitalWrite(13,HIGH);
    timerWrite(timer, 0);   // zera o timer
    t0 = 1;
  }
  if ((distancia2 <= 10 ) and (t1 == 0) ){
    value2 = 5; //mudar aqui
    digitalWrite(12,HIGH);
    timerWrite(timer1, 0);   // zera o timer1
    t1 = 1;
  }
```

```
  //Envia o valor para o master
  send(&value2, peerMacAddress);
}
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Success" : "Fail");
  readAndSend();
}
void loop() {
}
int getDistance()
{
    //faz a leitura das informacoes do sensor (em cm)
    int distanciaCM;
   long microsec = ultrasonic.timing();
  distanciaCM = ultrasonic.convert(microsec, Ultrasonic::CM);
    return distanciaCM;
}
int getDistance2()
{
    int distanciaCM2;
    long microsec2 = ultrasonic2.timing();
    distanciaCM2 = ultrasonic.convert(microsec2, Ultrasonic::CM);
    return distanciaCM2;
}
```

APENDICE II – CÓDIGO DO DISPOSITIVO RECEPTOR

```
#include <esp_now.h>
#include <WiFi.h>
#define CHANNEL 1

uint8_t teste_display = 0;
int decimal,unidade,decimal_2,unidade_2,valor_2;
int valor_display = 44;
int valor_led =0;
int principal = 1;
int secundario = 1;
int carros_p = 0;
int carros_s = 0;
int inverte_cruzamento = 0;
int estagio = 1;
hw_timer_t *timer = NULL;

void modeStation(){
    WiFi.mode(WIFI_STA);
    Serial.print("Mac Address in Station: ");
    Serial.println(WiFi.macAddress());
}
void InitESPNow() {
    if (esp_now_init() == ESP_OK) {
        Serial.println("ESPNow Init Success");
    }
    else {
        Serial.println("ESPNow Init Failed");
        ESP.restart();
    }
}
void timerDisplay(){
      if(estagio == 1){
      valor_display = valor_display - 1;
```

```
digitalWrite(22,LOW);  //VERMELHO          // PRINCIPAL
digitalWrite(23,LOW);   //AMARELO
digitalWrite(14,HIGH);   //VERDE
digitalWrite(32,LOW);  //VERDE             // SECUNDARIO
digitalWrite(33,LOW);   //AMARELO
digitalWrite(25,HIGH);   //VERMELHO
if(valor_display == 4){
  estagio = 2;
}
}
else if(estagio == 2){
  valor_display = valor_display - 1;
  principal = 0;
  if(valor_display > 2){
    digitalWrite(22,HIGH);  //VERMELHO          // PRINCIPAL
  digitalWrite(23,LOW);   //AMARELO
  digitalWrite(14,LOW);   //VERDE
  digitalWrite(32,LOW);  //VERDE             // SECUNDARIO
  digitalWrite(33,LOW);   //AMARELO
  digitalWrite(25,HIGH);   //VERMELHO

  }
  if(valor_display <=2 and valor_display > 0){
    digitalWrite(22,LOW);  //VERMELHO          // PRINCIPAL
  digitalWrite(23,HIGH);   //AMARELO
  digitalWrite(14,LOW);   //VERDE
  digitalWrite(32,LOW);  //VERDE             // SECUNDARIO
  digitalWrite(33,LOW);   //AMARELO
  digitalWrite(25,HIGH);   //VERMELHO

  }
  if(valor_display == -1){
    estagio = 3;
```

```
    //valor_display = 18;
    if( carros_p >= carros_s and carros_p <= carros_s + 3){
      valor_display = 34;
      digitalWrite(32,HIGH);  //VERDE            // SECUNDARIO
  digitalWrite(33,LOW);   //AMARELO
  digitalWrite(25,LOW);   //VERMELHO
    }
    else if (carros_p > carros_s + 4){
      valor_display = 24;
      digitalWrite(32,HIGH);  //VERDE            // SECUNDARIO
  digitalWrite(33,LOW);   //AMARELO
  digitalWrite(25,LOW);   //VERMELHO
    }
    else {
      valor_display = 44;
      digitalWrite(32,HIGH);  //VERDE            // SECUNDARIO
  digitalWrite(33,LOW);   //AMARELO
  digitalWrite(25,LOW);   //VERMELHO
    }
    carros_p = 0;
    carros_s = 0;
    principal = 1;
    inverte_cruzamento = 1;
   }
  }
else if(estagio ==3 ){
  valor_display = valor_display - 1;
  digitalWrite(22,LOW);  //VERMELHO         // PRINCIPAL
    digitalWrite(23,HIGH);   //AMARELO
    digitalWrite(14,LOW);   //VERDE

  digitalWrite(32,HIGH);  //VERDE            // SECUNDARIO
  digitalWrite(33,LOW);   //AMARELO
```

```
  digitalWrite(25,LOW);   //VERMELHO
  if(valor_display == 4){       // era 4
    estagio = 4;
    }
}
else {
  valor_display = valor_display - 1;
  secundario = 0;
   if(valor_display > 2){
      digitalWrite(22,LOW);  //VERMELHO          // PRINCIPAL
    digitalWrite(23,HIGH);   //AMARELO
    digitalWrite(14,LOW);   //VERDE
  digitalWrite(32,LOW);  //VERDE             // SECUNDARIO
  digitalWrite(33,HIGH);   //AMARELO
  digitalWrite(25,LOW);   //VERMELHO
    }
    if(valor_display <=2 and valor_display > 0){
      digitalWrite(22,LOW);  //VERMELHO          // PRINCIPAL
    digitalWrite(23,HIGH);   //AMARELO
    digitalWrite(14,LOW);   //VERDE

  digitalWrite(32,LOW);  //VERDE             // SECUNDARIO
  digitalWrite(33,LOW);   //AMARELO
  digitalWrite(25,HIGH);   //VERMELHO

    }
  if(valor_display == -1){                                  //era 0
      estagio = 1;
     // valor_display = 23;
      if( carros_p >= carros_s and carros_p <= carros_s + 3){
        digitalWrite(22,LOW);  //VERMELHO          // PRINCIPAL
    digitalWrite(23,LOW);   //AMARELO
    digitalWrite(14,HIGH);   //VERDE
```

```
          valor_display = 44;
         }
         else if (carros_p > carros_s + 4){
          digitalWrite(22,LOW);  //VERMELHO          // PRINCIPAL
        digitalWrite(23,LOW);   //AMARELO
        digitalWrite(14,HIGH);   //VERDE
          valor_display = 64;
         }
         else {
          digitalWrite(22,LOW);  //VERMELHO          // PRINCIPAL
        digitalWrite(23,LOW);   //AMARELO
        digitalWrite(14,HIGH);   //VERDE
          valor_display = 31;
         }
         carros_p = 0;
         carros_s = 0;
         secundario = 1;
         inverte_cruzamento = 0;
    }
}
}
void setup() {
  Serial.begin(115200);
  modeStation();
  InitESPNow();
  peer.channel = CHANNEL;
  peer.encrypt = 0;
  esp_now_register_recv_cb(onDataRecv);

   pinMode(15, OUTPUT);
   pinMode(2, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(19, OUTPUT);
```

```
    pinMode(21, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(18, OUTPUT);
    pinMode(12, OUTPUT);       //display 1
    pinMode(13, OUTPUT);       //display 2
    pinMode(26, OUTPUT);       //display 3
    pinMode(27, OUTPUT);       //display 4
    pinMode(22, OUTPUT);       //led vermelho
    pinMode(23, OUTPUT);       //led amarelo
    pinMode(14, OUTPUT);       //led verde
    pinMode(25, OUTPUT);       //led vermelho  painel 2
    pinMode(32, OUTPUT);       //led amarelo   painel 2
    pinMode(33, OUTPUT);       //led verde     painel 2
    digitalWrite(15,LOW);
    digitalWrite(2,LOW);
    digitalWrite(4,LOW);
    digitalWrite(19,LOW);
    digitalWrite(21,LOW);
    digitalWrite(5,LOW);
    digitalWrite(18,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
    digitalWrite(26,LOW);
    digitalWrite(27,LOW);
    digitalWrite(22,LOW);
    digitalWrite(23,LOW);
    digitalWrite(14,LOW);
    digitalWrite(25,LOW);
    digitalWrite(32,LOW);
    digitalWrite(33,LOW);

timer = timerBegin(0, 80, true);
    timerAttachInterrupt(timer, &timerDisplay, true);
```

```
    timerAlarmWrite(timer, 1000000, true);
    timerAlarmEnable(timer);
}
void onDataRecv(const uint8_t *mac_addr, const uint8_t *value, int len) {
  if(*value == 10){ // se for 10 é a principal e 5 é a secundaria
    carros_p = carros_p + 1;
  }
  if(*value == 5){
    carros_s = carros_s + 1;
  }
}
void onDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Success" : "Fail");
}
void loop() {
  multiplexa_display(valor_display);
}
void multiplexa_display(int valor){
  if(inverte_cruzamento == 0){
    decimal = valor / 10;
  unidade = valor%10;
  valor_2 = valor - 4;
  decimal_2 = valor_2 / 10;
  unidade_2 = valor_2 % 10;
  }
  else {
    decimal_2 = valor / 10;
  unidade_2 = valor%10;
  valor_2 = valor - 4;
  decimal = valor_2 / 10;
  unidade = valor_2 % 10;
  }
 if(principal == 1){
```

```
      digito(unidade_2);
  digitalWrite(12,HIGH);  //12 e 13 display 1
  delay(1);
  digitalWrite(12,LOW);
  digito(decimal_2);
  digitalWrite(13,HIGH);
  delay(1);
  digitalWrite(13,LOW);
    }
    if(secundario == 1) {
  digito(unidade);
  digitalWrite(26,HIGH);
  delay(1);
  digitalWrite(26,LOW);
  digito(decimal);
  digitalWrite(27,HIGH);
  delay(1);
  digitalWrite(27,LOW);
    }
}
void digito(int numero){
  if(numero == 0){
    digitalWrite(15,HIGH);
    digitalWrite(2,HIGH);
    digitalWrite(4,HIGH);
    digitalWrite(19,HIGH);
    digitalWrite(21,HIGH);
    digitalWrite(5,HIGH);
    digitalWrite(18,LOW);
  }
 else if(numero == 1){
    digitalWrite(15,LOW);
    digitalWrite(2,LOW);
```

```
  digitalWrite(4,HIGH);
  digitalWrite(19,LOW);
  digitalWrite(21,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(18,LOW);
 }
else if(numero == 2){
  digitalWrite(15,HIGH);
  digitalWrite(2,HIGH);
  digitalWrite(4,LOW);
  digitalWrite(19,HIGH);
  digitalWrite(21,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(18,HIGH);
 }
else if(numero == 3){
  digitalWrite(15,LOW);
  digitalWrite(2,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(19,HIGH);
  digitalWrite(21,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(18,HIGH);
 }
else if(numero == 4){
  digitalWrite(15,LOW);
  digitalWrite(2,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(19,LOW);
  digitalWrite(21,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(18,HIGH);
 }
```

```
else if(numero == 5){
   digitalWrite(15,LOW);
   digitalWrite(2,HIGH);
   digitalWrite(4,HIGH);
   digitalWrite(19,HIGH);
   digitalWrite(21,LOW);
   digitalWrite(5,HIGH);
   digitalWrite(18,HIGH);
 }
else if(numero == 6){
   digitalWrite(15,HIGH);
   digitalWrite(2,HIGH);
   digitalWrite(4,HIGH);
   digitalWrite(19,HIGH);
   digitalWrite(21,LOW);
   digitalWrite(5,HIGH);
   digitalWrite(18,HIGH);
 }
else if(numero == 7){
   digitalWrite(15,LOW);
   digitalWrite(2,LOW);
   digitalWrite(4,HIGH);
   digitalWrite(19,HIGH);
   digitalWrite(21,HIGH);
   digitalWrite(5,LOW);
   digitalWrite(18,LOW);
 }
else if(numero == 8){
   digitalWrite(15,HIGH);
   digitalWrite(2,HIGH);
   digitalWrite(4,HIGH);
   digitalWrite(19,HIGH);
   digitalWrite(21,HIGH);
```

```
    digitalWrite(5,HIGH);
    digitalWrite(18,HIGH);
  }
 else{
    digitalWrite(15,LOW);
    digitalWrite(2,LOW);
    digitalWrite(4,HIGH);
    digitalWrite(19,HIGH);
    digitalWrite(21,HIGH);
    digitalWrite(5,HIGH);
    digitalWrite(18,HIGH);
  }
}
```