FACULDADE DE TECNOLOGIA

FATEC SANTO ANDRÉ

Tecnologia em Eletrônica Automotiva

BRUNO DOMINGOS TEIXEIRA

VINÍCIUS CAVALCANTE PAZZINI

SISTEMA DE COMPARTILHAMENTO VEICULAR

Santo André

2020

Bruno Domingos Teixeira

Vinícius Cavalcante Pazzini

SISTEMA DE COMPARTILHAMENTO VEICULAR

Trabalho de Conclusão de Curso entregue à Fatec Santo André como requisito parcial para obtenção do título de Tecnólogo em Eletrônica Automotiva.

Orientador: Prof. Weslley Medeiros Torres

Santo André

2020

FICHA CATALOGRÁFICA

T266s

Teixeira, Bruno Domingos

Sistema de compartilhamento veicular / Bruno Teixeira Domingos, Vinicius Cavalcante Pazzini. - Santo André, 2020. – 73f: il.

Trabalho de Conclusão de Curso – FATEC Santo André. Curso de Tecnologia em Eletrônica Automotiva, 2020.

Orientador: Prof. Weslley Medeiros Torres

1. Eletrônica. 2. Veículos 3. Carsharing. 4. Mobilidade. 5. Bluetooth. 6. Sistema embarcado. 7. Dispositivo mobile. 8. Tecnologia. I. Pazzini, Vinicius Cavalcante. II. Sistema de compartilhamento veicular.

629.2

Bruno Domingos Teixeira

Vinicius Cavalcante Pazzini

SISTEMA DE COMPARTILHAMENTO VEICULAR

Trabalho de Conclusão de Curso apresentado a FATEC SANTO ANDRÉ como requisito parcial à obtenção de título de Tecnólogo em Eletrônica Automotiva.

BANCA EXAMINADORA

Local: Fatec Santo André
Horário: 20:00
Data: 17/07/2020

Prof. Weslley Medeiros Torres
Presidente da Banca
Fatec Santo André

Prof. Fernando Garup Dalbo
Primeiro membro da Banca
Fatec Santo André

Prof. Murilo Zanini de Carvalho
Segundo Membro da Banca
Fatec Santo André

Santo André

2020

AGRADECIMENTOS

Bruno Teixeira:

Gostaria primeiramente de agradecer a Deus, por todos os momentos que passei durante todo o meu curso, sem ele, jamais teria conseguido qualquer resultado durante todos estes anos. Em segundo lugar, quero agradecer aos meus pais que sempre me apoiaram nos dias difíceis e incentivaram a sempre entregar o meu melhor em tudo que fizesse.

Agradeço aos meus colegas da FATEC Santo André, que sempre me ajudaram em momentos e que direta, ou indiretamente colaboraram com o desenvolvimento desse trabalho. Por fim, agradeço aos professores da universidade, que sempre se dedicam em ensinar novos conhecimentos, em especial, os meus orientadores Weslley Medeiros Torres e Fernando Garup Dalbo, que me direcionaram neste trabalho e sempre acreditaram no meu potencial.

Vinicius Pazzini:

Agradeço ao meu pai, Luiz Carlos Pazzini, pela confiança no meu progresso e paciência durante todo o decorrer da elaboração deste trabalho.

A minha namorada Jéssica Baltazar por se manter firme ao meu lado, agradecê – la também como amiga e conselheira nos momentos difíceis, incentivando – me sempre que possível.

Aos meus orientadores Weslley Medeiros Torres e Fernando Garup Dalbo, pela atenção e ajuda que fizeram total diferença e foram essenciais para que o trabalho fosse concluído.

Aos meus colegas do curso de Eletrônica Automotiva e da FATEC Santo André pelo convívio e cooperação durante estes anos.

RESUMO

Este trabalho consiste em realizar um estudo sobre o compartilhamento de veículos com base no contexto atual da mobilidade urbana, conceitos, tipos e a forma pela qual este sistema é implementado ao redor do mundo e desenvolver um protótipo utilizando como base o projeto feito por Granja e Yamawaki (2019), que tem por objetivo criar um aplicativo que realize a substituição das chaves mecânicas dos carros, por chaves eletrônicas. Com isso queremos criar um aplicativo que respeite as metodologias de aplicação do *Carsharing* e que permita a administração de usuários através de um banco de dados e utilizando um microcontrolador ESP32 que possui comunicação via *Bluetooth* e Wifi, acessar o barramento de comunicação do carro e controlar o sistema de partida e travamento de portas para oferecer aos usuários um método prático e simples, com comodidade e agilidade no processo de aluguel de veículos na cidade.

Palavras-chave: Compartilhamento veicular. Mobilidade. ESP32. Chaves. *Bluetooth*. Wifi.

ABSTRACT

This work consists of conducting a study on vehicle sharing based on the current context of urban mobility, concepts, types and the way in which this system is implemented around the world and developing a prototype using as a basis the project made by Granja e Yamawaki (2019), which aims to create an application that switches the mechanical keys of cars, in electronic keys. With this we want to create an application that respects the application methodologies of *Carsharing* and that allows the administration of users through a database and using a microcontroller called ESP32 that has communication via *Bluetooth* and Wifi, access the communication bus of the car and control the starting and locking system of doors to offer users a practical and simple method, with convenience and agility in the process of renting vehicles in the city.

Keywords: Vehicle sharing. Mobility. ESP32. Keys. *Bluetooth*. Wifi.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES

µC	Microcontrolador
ACL	*Assyncronous Connection – Less*
ACT	*Activity*
API	*Application Programming Interface*
B2C	*Business-To-Consumer*
BC	Bicicleta Compartilhada
BLE	*Bluetooth Low Energy*
BPS	Bits por Segundo
BR	*Basic Rate*
BT	*Bluetooth*
CSS	*Carsharing Service*
EDR	*Enhanced Data Rate*
EDT	*EditText*
EUA	Estados Unidos da América
FF	*Free-Floating*
FH/TDD	*Frequency Hopping/ Time Division Duplex*
FH-CDMA	*Frequency Hopping – Code Division Multiple Access*
GHz	Giga Hertz
HCI	*Host Controller Interface*
HS	*High Speed*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISM	*Industrial, Scientific, Medical*

IT	*Intent*
IU	Interface de Usuário
L2CAP	*Logical Link Control and Adaptation Prootocol*
led	Light – Emitting Diode
LMP	*Link Manager Protocol*
m	Metros
MHz	Mega Hertz
OW	*One-Way*
P2P	*Peer-To-Peer*
PSS	*Product-Service System*
QoS	*Quality of Service*
RF	*Radio Frequency*
RT	*Round Trip*
SCO	*Syncronous Connection-Oriented*
SO	Sistema Operacional
THR	*Thread*
TX	Taxi
TXV	*TextView*
WAP	*Wireless Application Protocol*
Wifi	*Wireless Fidelity*

SUMÁRIO

1. INTRODUÇÃO

O crescente aumento de veículos nas ruas e o aumento da poluição e congestionamento nas cidades, torna-se necessário o surgimento de novas tecnologias e metodologias visando solucionar os diversos problemas na mobilidade urbana. O *Carsharing Service* (CSS) é um desses métodos.

O CSS conforme Stone (2013) é similar ao aluguel de carros, no qual os usuários possuem acesso a uma rede de veículos, que podem ser alugados por qualquer período de tempo. O CSS é um modo de transporte compartilhado, onde uma determinada empresa vende a utilização do veículo pelo tempo desejado pelo cliente, mas ainda possuindo a propriedade do automóvel. Surgiu por volta de 1948 em Zurique, na Suíça e atualmente o CSS é utilizado em centenas de países, até mesmo no Brasil.

Existem três modelos principais de CSS, sendo estes: o modelo chamado de estação, que tem o objetivo de se colher e devolver os veículos a uma mesma estação, um segundo tipo, onde os usuários podem iniciar e encerrar o período de locação em diferentes estações, e por último, temos o do tipo flutuante, onde se pode pegar e devolver o veículo em qualquer estacionamento da área de operação, pagando apenas pelo tempo em que se utilizou o veículo.

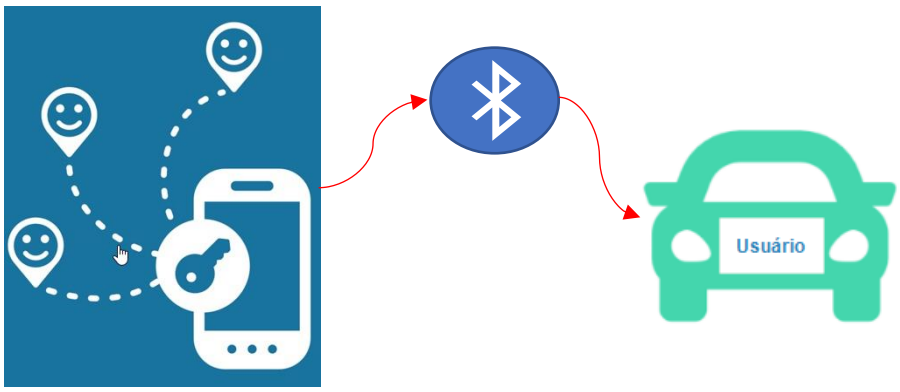
1.1. Objetivo Geral

Este trabalho tem como objetivo desenvolver um protótipo de um sistema de compartilhamento veicular aplicando o conceito de mobilidade compartilhada e aprimorando estes conceitos, utilizando informações simuladas do barramento de comunicação do veículo, permitindo a substituição da chave mecânica por chaves eletrônicas e realizar o gerenciamento de usuários.

Desenvolver um aplicativo para o *smartphone,* aplicando as metodologias de CSS e realizar gestão do banco de dados de usuários, para o acesso às informações do veículo, e conectar o *smartphone* ao microcontrolador (µC) ESP32. Dessa forma ela realiza a comunicação via *Bluetooth* (BT) com o *smartphone* e, acessa o

barramento de comunicação do carro para o controle de partida e travamento de portas do veículo. Essa comunicação é ilustrada na Figura 1.

Figura 1: Conexão *Smartphone* e veículo via BT



Fonte: Autoria Própria (2020)

1.2. Motivação

Em todo o mundo, temos visto novas tecnologias sendo desenvolvidas para solucionar os diversos problemas da mobilidade urbana. Demonstrar que os sistemas de CSS oferecem uma forma de diminuição de fatores ambientais e excesso de veículos, com o foco em criar uma solução de baixo custo, que ofereça lucro as empresas e cumpra com as exigências de seus consumidores.

2. DESENVOLVIMENTO

Neste capítulo foi realizada uma pesquisa para entender como o *Carsharing* é aplicado no mundo, quais são os tipos de compartilhamento veicular mais predominam nas grandes cidades mundiais, como o *Carsharing* é classificado no mercado pelos clientes e estudos sobre mobilidade em algumas cidades pioneiras nesse tipo de serviço.

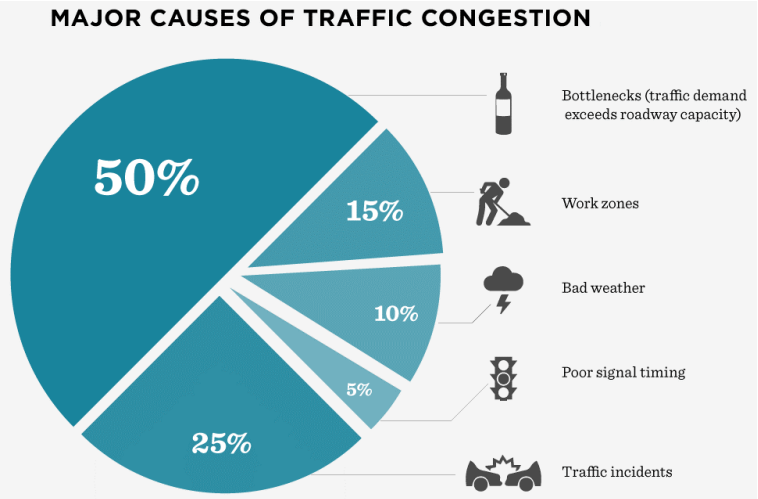
2.1. Mobilidade compartilhada

“A era atual é marcada pelo uso indiscriminado de recursos naturais, pela falta de planejamento urbano e a deterioração da qualidade de vida dos habitantes das grandes cidades” (Machado, et al., 2018).

Se considerarmos a expansão populacional, inevitavelmente nos depararemos com a crescente demanda de transporte, que em situações extremas, sobrecarrega o sistema de tráfego. É bem comum todos os dias, em horários de pico encontrarmos altas concentrações de veículos, estacionamentos lotados e ruas congestionadas, principalmente em grandes cidades nos conta Mindur, Sierpinski e Turón (2018).

INRIX (2018) ressalta que somente nos Estados Unidos são gastos cerca de 87 bilhões de dólares anualmente com congestionamentos, tendo como as cidades mais “congestionadas” do país, Boston e Washington D.C, onde seus motoristas perderam 164 e 155 horas no trânsito respectivamente no ano de 2018. A Figura 2 ilustra as cinco maiores causas de congestionamento de trânsito, sendo metade do problema gerado pela demanda de tráfego que excede as capacidades da via:

Figura 2: Maiores causas do congestionamento



Fonte: (SABA, 2015)

A autora Saba (2015) diz que essa quantidade excessiva de veículos trafegando nas ruas se mostra um grande problema, rendendo impactos na economia, aumento da poluição, diversos tipos de acidentes, locomoção dificultosa, e horas de estresse e cansaço.

Outro fator que agrava o problema de poluição é a taxa de ocupação veicular. De acordo com Bergamini (2014) na cidade de São Paulo a maioria dos veículos em movimento costuma ter em média ocupação de 1,47, ou seja, menos de dois ocupantes. Conforme os resultados da pesquisa realizada pelo Instituto de Pesquisa e Planejamento Urbano de Londrina (2016) em 2013, verificou-se a ocupação de veículos em 5 locais diferentes na cidade de Londrina, das 13:30 às 14:30 horas, onde foi registrada a mesma taxa de 1,47 encontrada na cidade de São Paulo, sendo que cada veículo costuma ter entre 4 a 5 acentos além do motorista.

Machado et al. (2018) diz que para se obter um sistema eficiente de transporte, é necessário a significativa redução do uso e circulação de carros, adotando formas de transporte eficientes, como o transporte público, ciclismo, caminhada e modos de compartilhamento.

Os autores Mindur, Sierpinski e Turón (2018) afirmam que apesar de o transporte público, atender a demanda, ele não oferece a solução dos problemas

apresentados. A sua expansão, apesar de trazer benefícios, à longo prazo acaba sobrecarregado novamente, um exemplo disso são as faixas exclusivas de rodagem.

Apesar de útil, o sistema público não oferece dinamismo de transporte, já que seus serviços são em locais específicos e em horários pré-determinados. Dessa forma são necessárias soluções de maior abrangência, como a reconstrução do sistema de tráfego existente, mudanças em sua organização e maior aproveitamento da crescente evolução tecnológica.

Segundo Martinez e Viegaz (2017, apud Machado et al. 2018), é importante termos em mente que os veículos pessoais ainda possuem vantagens, como flexibilidade, conforto e disponibilidade. Com essa afirmação podemos concluir que os modos de compartilhamento oferecem soluções efetivas para os problemas atuais de mobilidade, já que possui todas as características apresentadas.

Machado et al. (2018) explica que, nos países onde a mobilidade compartilhada se popularizou, houve redução de custos externos, aumento das taxas de ocupação e significantes resultados na diminuição dos congestionamentos e emissão de poluentes.

Neste contexto surge a ideia de mobilidade compartilhada, com diversas modalidades e serviços que visam a redução de veículos nas ruas, redução de emissões de poluentes, facilitando a mobilidade de seus usuários com a integração dos meios de transporte e aplicando a mobilidade sustentável.

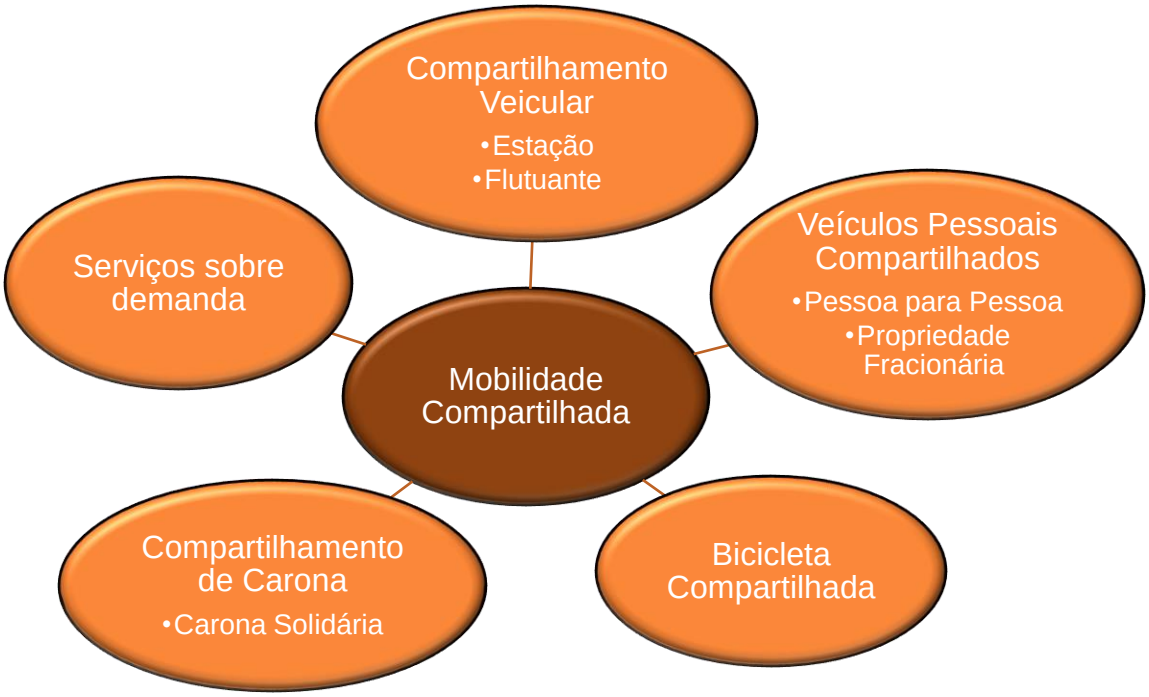
Uma de suas características, conforme Mindur, Sierpinski e Turón (2018), são os modelos de negócio que utilizam de plataformas online ou aplicativos *mobile*.

Existe da parte do governo a iniciativa para que a população adote estas formas de transporte. Conforme Machado et al. (2018), nos EUA, em cidades como

Houston e Texas, existem vagas de estacionamento exclusivas para veículos de compartilhamento.

Ele complementa demonstrando vários modelos de mobilidade compartilhada, ilustrados pela Figura 3.

Figura 3: Modos de Mobilidade Compartilhada



Fonte: (Machado, et al., 2018)

Conforme a Figura 3, a mobilidade compartilhada pode ser dividida em cinco modalidades:

1. Compartilhamento Veicular (CSS): Serviço semelhante ao de aluguel veicular, aonde o seu custo é pelo tempo efetivo de utilização. É apresentado como uma forma de mobilidade alternativa ao transporte público, e possui dois tipos de frotas: a de veículos a combustão, seja à gasolina ou diesel e os do tipo ecológico, que são veículos híbridos e elétricos. Ao redor do mundo, temos três

tipos principais de CSS em operação, sendo estes: Estação, Ida e Volta e Flutuante, conceitos esses que serão abordados mais a frente;

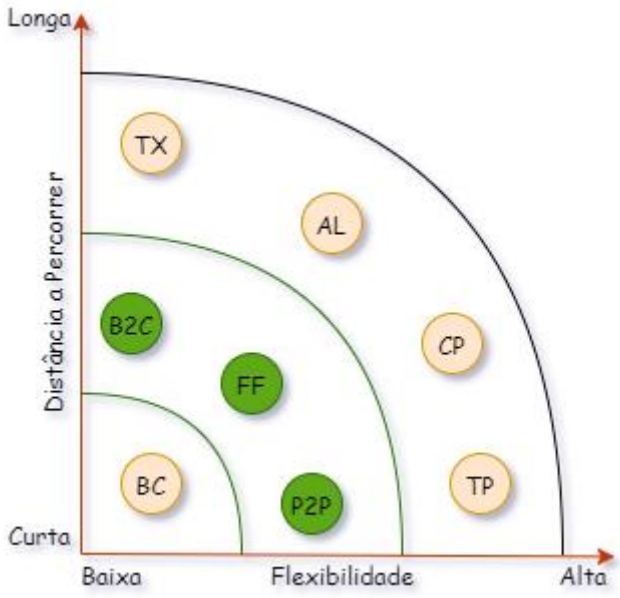
2. Serviços sobre demanda: São meios inovadores onde pode-se acessar as opções de transporte usando *smartphone* quando necessário. Devido a seu custo-benefício, facilidade de reserva e informações automáticas de destino, tornaram esse serviço popular. Seus segmentos principais são no fornecimento de materiais e montagem de equipamentos;
3. Compartilhamento de carona: É característico por ser um modelo de mobilidade sem fins lucrativos, onde três ou mais indivíduos sem parentesco, compartilham uma viagem e os passageiros ajudam nas despesas do motorista. Geralmente esse esquema é utilizado por pessoas que viajam longas distâncias para trabalho ou escola;
4. Bicicleta compartilhada: Esse sistema é bem semelhante ao de CSS, suas estações de compartilhamento são concentradas em áreas urbanas e se divide entre os mesmos tipos do CSS. Conforme Machado et al. (2018), a bicicleta compartilhada oferece redução de tempo e custos de viagem nos centros das cidades, melhorando a saúde do corpo e mais experiências sociais e de lazer. Para a escolha deste meio de transporte é importante considerar: Condições ambientais ou naturais, onde o clima e a topografia do local podem interferir, atributos de viagem, como por exemplo a distância a ser percorrida e características sócio econômicas, como idade, sexo e saúde, onde pessoas de menor condição financeira tendem a buscar esse tipo de serviço pelo seu baixo custo;
5. Veículos pessoais compartilhados: Um tipo específico de CSS, onde o anúncio do automóvel é feito pelo seu proprietário, e tanto a coleta, quanto a devolutiva do veículo é feita no local determinado por ele.

Segundo a Deloitte (2017), os conceitos de mobilidade podem ser classificados pela sua flexibilidade e pela distância a percorrer. Com base nessa afirmação, a escolha do meio de transporte, depende de fatores, climáticos, econômicos e distância. Comparando conceitos gerais da mobilidade atual durante a escolha dos serviços, é necessário considerar:

1. Bicicletas pessoais ou de compartilhamento: Se mostram ideais para curtas distâncias, porém com baixa flexibilidade, já que condições climáticas, atributos de viagem e sócio – econômicas interferem diretamente na sua utilização;
2. Serviços de Taxi, Aluguel, Compartilhamento de Carona e Transporte Público: Perfeitos para longas distâncias, com flexibilidade relativa, pois apesar de não serem afetados pelos problemas climáticos ou sócio – econômicos, podem não oferecer total privacidade, liberdade, ou comodidade, dependendo só serviço escolhido;
3. Serviços de CSS do tipo estação, ida e volta e flutuante: Oferecem flexibilidade e custo-benefício, junto da liberdade e comodidade semelhantes a de veículos pessoais, mostrando – se ideal para o dia a dia.

A Figura 4 ilustra o conceito apresentado, onde BC representa bicicletas, B2C, FF e P2P são os serviços de CSS do tipo estação, flutuante e ida e volta respectivamente. TX são serviços de taxi, AL significa aluguel veicular convencional, CP simboliza o compartilhamento de carona e TP o sistema público de transporte.

Figura 4: Relação dos conceitos de mobilidade por flexibilidade e distância



Fonte: (DELOITTE, 2017)

Independente do modelo escolhido Mindur, Sierpinski e Turón (2018), afirmam que é impossível satisfazer o gosto pessoal de todos os indivíduos, significando que a melhor forma de transporte, irá depender acima de tudo, daquilo que melhor satisfaz seus clientes em sua utilização.

2.2. Sistemas Produtos-Serviços

O conceito do *Product-Service System* (PSS) é a união de produtos e serviços, focados em satisfazer as necessidades dos seus usuários, trazendo-lhes soluções benéficas para a criação ou inovação de negócios, atentando-se para conforto, qualidade, customização, novas funções ou combinações na oferta do produto oferecido, de acordo com Melo, Moro e Miguel (2017). Ela também diz que um PSS representa para as empresas uma oportunidade competitiva de redução de consumo, sem perder o valor de seu produto, mudando apenas a maneira como eles são utilizados nas prestações de serviços.

O PSS tem por objetivo conforme Beuren, Amaral e Miguel (2013) redirecionar a venda de “produtos físicos” para vender “soluções” aos clientes, sendo benéfico para ambas as partes.

Com isso as empresas podem conseguir aumento de lucros, diferenciação de concorrentes, renda estabilizada e clientela fidelizada. A Figura 5 representa as categorias e subcategorias do PSS, segundo o autor Reis e Junior (2015) sendo estes:

- Produto Orientado ao Uso: Negócio voltado à venda de um produto, agregando valor a este, por meio da oferta de serviços extras ao produto, por exemplo: serviços de atendimento ao cliente e garantia estendida;
- Uso Orientado ao Serviço: Vende-se ao cliente a utilização de um determinado produto, porém a propriedade ainda pertence a empresa. Podemos utilizar o exemplo de aluguéis de todos os tipos ou locadoras;
- Serviço Orientado Para os Resultados: Oferta de resultados em concordância entre cliente e fornecedor, não havendo produtos envolvidos. A remuneração é pela solução, por exemplo: lavanderias e empresas de manutenção veicular.

Figura 5: Categorias e subcategorias do PSS

Fonte: (Tukker, 2004)

Os serviços de CSS são considerados neste contexto como um PSS de uso orientado ao serviço, porque as empresas oferecem serviços de um veículo e este mesmo continua sendo de sua propriedade.

2.3. Compartilhamento Veicular (*Carsharing*)

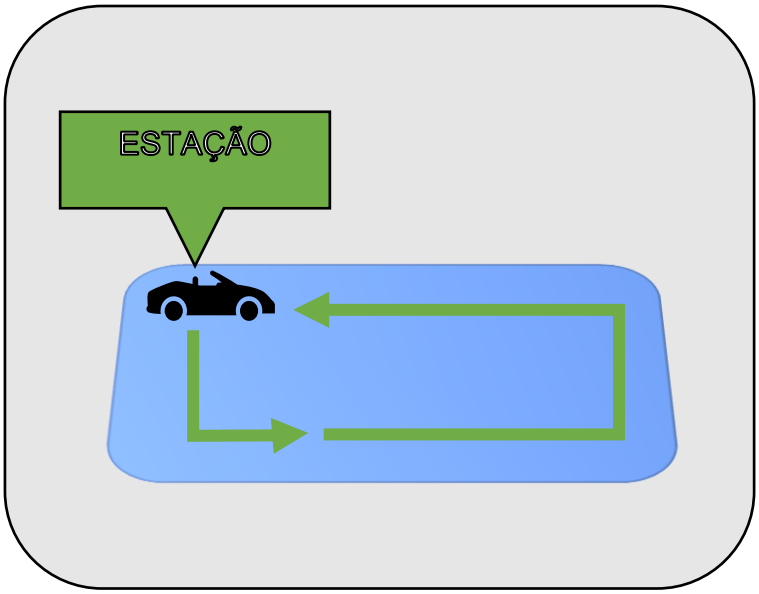
Os serviços CSS são semelhantes aos aluguéis de veículos, aonde o pagamento é feito de acordo com o tempo e a distância percorrida efetivamente.

A autora Stone (2013) o CSS pode ser definido como um sistema que permite que pessoas utilizem carros disponíveis, por meio de reserva ou não, por qualquer duração. Os autores Mindur, Sierpinski e Turón (2018), descrevem o CSS como uma possível substituição ao veículo particular, por oferecer um meio eficiente de locomoção e devido eliminação de gastos fixos com combustível, manutenções e documentações.

2.4. Tipos de CSS

As primeiras empresas a aplicar o *CSS* segundo Münzel, et al. (2017) começaram a aplicar pelo método *business-to-consumer* (B2C), onde o cliente retirava o veículo em uma base central e ao término do uso o veículo deveria ser retornado a mesma estação. Entretanto, postos de estacionamento espalhados pela cidade eram disponibilizados as empresas pelos governos locais para promover uma forma de mobilidade mais sustentável, exemplificado pela Figura 6:

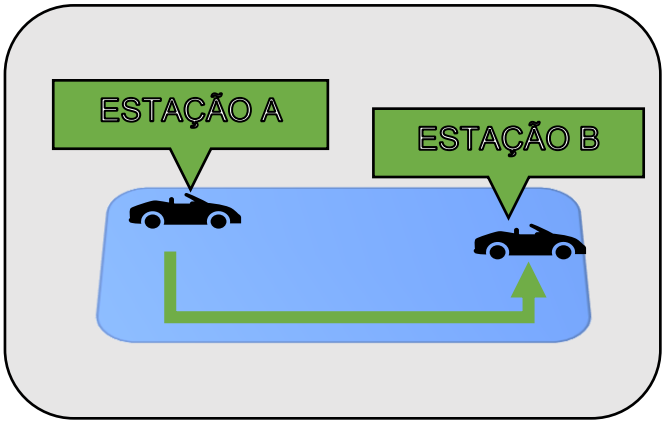
Figura 6: Método de CSS tipo B2C



Fonte: (Machado, et al., 2018)

Münzel, et al. (2017) ainda afirma que o serviço do tipo *round trip* (RT) é muito semelhante ao B2C onde os veículos necessitavam de ser devolvidos ao mesmo local de onde foram retirados, representados pela Figura 7:

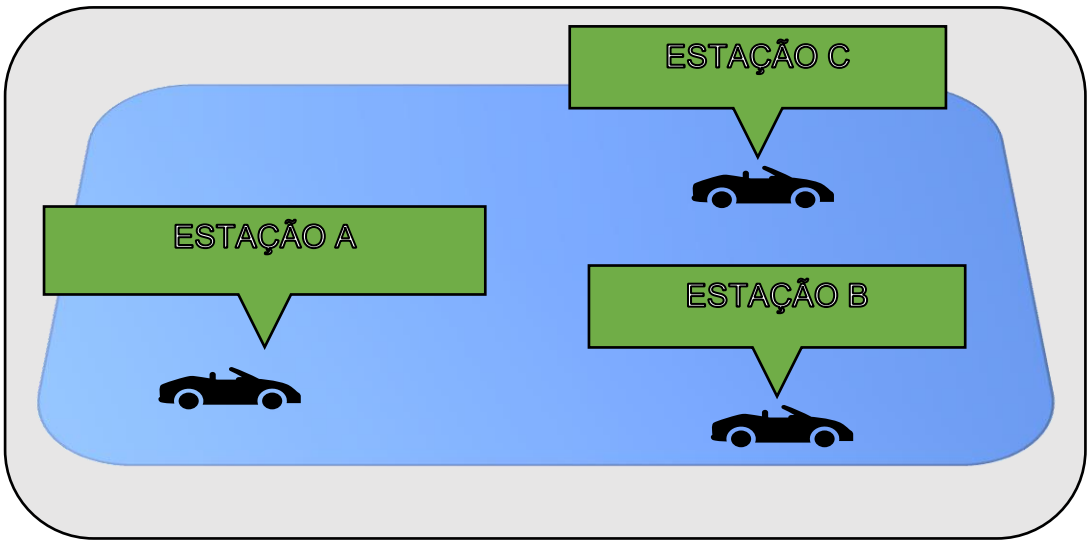
Figura 7: Método de CSS do tipo RT



Fonte: (Machado, et al., 2018)

No modelo de serviço do tipo *free-floating* (FF) de acordo com Münzel, et al. (2017), é dada a liberdade ao cliente para deixar o veículo alugado em lugares diferentes de onde ele foi retirado, representado pela Figura 8:

Figura 8: CSS tipo FF



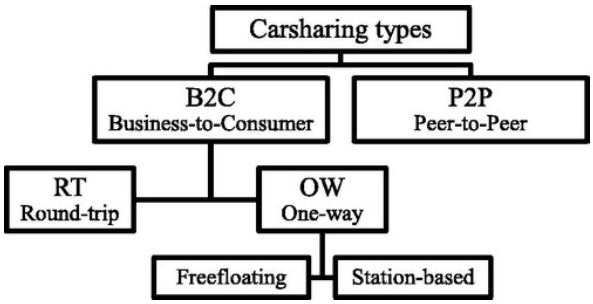
Fonte: (Machado, et al., 2018)

O veículo deve ser entregue em uma vaga de estacionamento público dentro de uma área pré-estabelecida pela empresa, reafirma Münzel, et al. (2017). O serviço

do tipo *one-way* (OW) se assemelha ao do tipo FF, porém o veículo deve ser devolvido em um ponto definido, por exemplo, uma vaga de estacionamento da empresa em outro ponto da cidade.

Também nos diz que no tipo chamado *peer-to-peer* (P2P) o proprietário do veículo é quem faz o anúncio do seu veículo em uma plataforma, onde será cobrada uma taxa pela oferta e demanda e os consumidores podem alugar os veículos quando ociosos através da plataforma. Neste tipo de serviço o veículo deve ser retirado e devolvido ao proprietário do carro. A Figura 9 ilustra cada um dos tipos:

Figura 9: Tipos de CSS

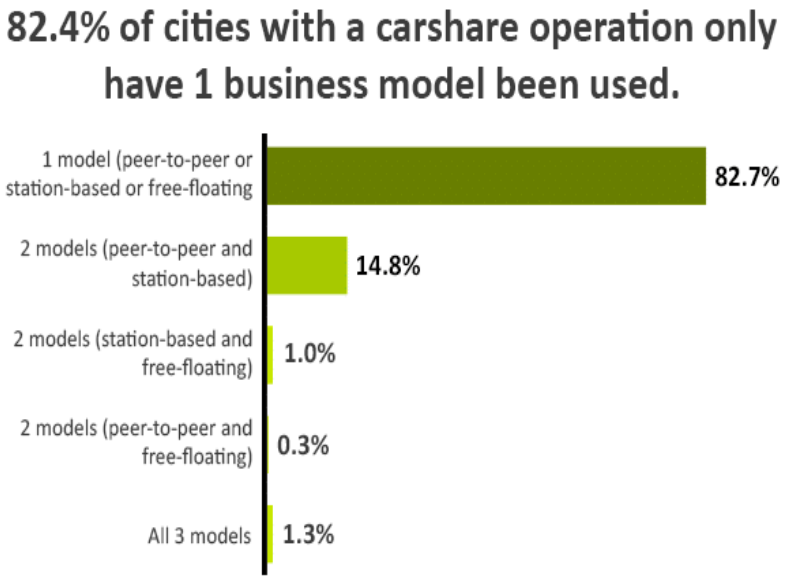


Fonte: (Münzel, et al., 2017)

2.4.1 CSS no mundo

Em 2019 conforme a Movmi e Carsharing Association (2019), 82,4% das cidades em todo o mundo operam somente com uma prestadora de serviço local e com apenas um modelo de CSS. Onde, de fato o compartilhamento de carros em estações ainda é dominante, sendo pequeno o número de cidades que aplicam mais de um exemplo de compartilhamento. Essa informação é ilustrada pela Figura 10.

Figura 10: Porcentagem de tipos diferentes de CSS aplicados em cidades



Fonte: (Movmi e Carsharing Association, 2019)

No geral, a indústria de CSS vem crescendo ao longo dos anos, hoje este tipo de serviço já está presente em 59 países e em mais de 3000 cidades, contando com mais de 200 prestadoras ao redor do mundo, conforme mostrado no quadro 1. Um dos motivos do modelo de ponto a ponto abranger muito mais cidades, se dá pelo motivo de que na maioria dessas cidades os veículos possuem um seguro diferenciado, facilitando a aplicação deste tipo de sistema de compartilhamento (Movmi e Carsharing Association, 2019).

Quadro 1: Proporções da indústria de CSS em 2019

	Crescimento da indústria de *Carsharing* em 2019 por tipos		
	B2C	FF	P2P
Operadoras	143	77	25
Cidades	1358	160	2190
Países	51	36	19

Fonte: (Movmi e Carsharing Association, 2019)

2.4.2 CSS na Europa

Na Europa, segundo a empresa de estatísticas STATISTA (2014a) a Alemanha é o país que tem o maior número de usuários, com cerca de 700 mil em todo o país, e o maior número de veículos compartilhados, com mais de 13 mil (2014b). Entretanto, afirma Münzel, et al. (2017) que a cidade com o maior número de carros compartilhados atualmente é Paris, capital da França, tendo uma frota de aproximadamente 11 mil veículos.

A maioria desses veículos podem ser encontrados nas capitais, entretanto a menor população das cidades periféricas contribui para que as cidades menos populosas, tenham o maior número de veículos compartilhados disponíveis por habitantes. Podemos observar isto no quadro 2.

Quadro 2: As cinco principais cidades de CSS na Europa

	As 5 principais cidades de compartilhamento de carros na Europa					
	Carros compartilhados por 100.000 habitantes			Carros compartilhados		
	total	B2C	P2P	total	B2C	P2P
1	Karls ruhe Alemanha 237	Karls ruhe Alemanha 223	Montpellier França 202	Paris França 11477	Paris França 3961	Paris França 7516
2	Utrecht Países Baixos 214	Stuttgart Alemanha 125	Bordeaux França 177	Londres Reino Unido 3390	Berlim Alemanha 2676	Lyon França 1453
3	Montpellier França 211	Amsterdã Países Baixos 125	Versalhes França 175	Berlim Alemanha 3221	Londres Reino Unido 1955	Londres Reino Unido 1435
4	Amsterdã Países Baixos 210	Köln Alemanha 119	Toulouse França 171	Amsterdã Países Baixos 2172	Munique Alemanha 1589	Lille França 1386
5	Bordeaux França 204	Heidelberg Alemanha 116	Nantes França 162	Munique Alemanha 1806	Hamburgo Alemanha 1449	Bordeaux França 1295

Fonte: (Münzel, et al., 2017)

2.4.3 CSS na América do Norte

Na América do Norte houve um salto notável no número de prestadora de serviços, especificamente nos Estados Unidos da América (EUA) e no Canadá, segundo Shaheen, Cohen e Roberts (2006) o número de usuários está crescendo muito mais rápido do que o número de veículos compartilhados.

O seguro ainda é um obstáculo para este tipo de serviço nos conta o autor, os seguros para quem presta esse tipo de serviço tem um custo bem elevado, o que nos leva a concluir que a segurança é um ponto fundamental para as empresas de CSS, o tamanho e o crescimento do mercado de CSS nos EUA tem a ver com os incentivos do governo, tanto de níveis municipais quanto estaduais. De acordo com o autor, as legislações se concentram em incentivar o uso de veículos compartilhados, como por exemplo, parcerias entre operadoras de CSS e de transporte público. Ele conclui dizendo que os cidadãos de Vancouver no Canadá tinham acesso a uma frota de quase 3 mil veículos em 2016, e 29% da população adulta tinha alguma associação com operadoras de compartilhamento.

2.4.4 CSS no Brasil e na América Latina

A Movmi (2018) diz que a américa latina está muito atrás dos três principais mercados de compartilhamento, que são: Ásia, Europa e América do Norte. De acordo com o autor, a América do Sul em 2016 operava apenas com o modelo de compartilhamento do tipo B2C, um modelo que tem taxas de adoção mais baixas do que as do tipo FF. As questões econômicas e instabilidades das moedas sul – americanas contribuem para que operadores estrangeiros não invistam neste mercado tão volátil.

2.5. Estudos de mobilidade urbana

Segundo Fornier et al. (2015, apud Melo, Moro e Miguel, 2017) os pontos positivos do compartilhamento veicular impactam diretamente na diminuição do

número de veiculos nas cidades, e pelo mesmo motivo não seriam necessários grandes estacionamentos de veículos e a construção de novos espaços. Com essa iniciativa também é possível a inserção de uma frota com emissões igual a zero. Ele afirma que as pessoas terão prazer em dirigir carros novos, sem a pressão de precisar comprá-lo.

O autor Breuil e Baratt (2009) diz que desde 1999 está sendo executado em *La Rochelle*, uma cidade na costa oeste da França, com uma população de 160 mil habitantes e que pode chegar a 250 mil no período de férias, um CSS de autoatendimento. Contando com uma frota de 50 carros elétricos espalhados em 7 estações localizadas no sudoeste da cidade.

O autor considera que a comunidade urbana de *La Rochelle* vê o CSS como uma estratégia promissora para reduzir o congestionamento do tráfego.

De acordo com ele os principais objetivos do projeto eram:

- A diminuição do número de veículos na cidade;
- A diminuição dos veículos estacionados pela cidade;
- Aumentar o compartilhamento de carros pela área da cidade;
- Reduzir as emissões de poluentes;
- Melhorar a segurança dos cidadãos e
- Promover o uso de veículos limpos para a população;

Os resultados obtidos foram:

- Incremento médio no número de usuários do serviço;
- Poucos turistas utilizando o serviço;
- Aumento do uso de veículos limpos aplicados no programa e
- Compartilhamento de furgões para atender a demanda de transporte de mercadorias por artesãos, comerciantes e cidadãos.

De acordo com Jung e Koo (2018) foi realizado um estudo que examina os impactos das emissões resultantes da mudança no modo de transporte, usando um modelo com base nas preferências do consumidor e da probabilidade de escolher o CSS como meio de transporte, onde foi concluído que muitas pessoas renunciariam ao veículo próprio optando por veículos compartilhados, diz também que com um serviço de compartilhamento mais flexível é tendencioso que mais pessoas o utilizem.

Os autores nos conta também que é importante obter uma forma sustentável de geração de energia elétrica para que o ciclo de vida do serviço de CSS não seja tão danoso ao meio ambiente, onde sim, a emissão de poluentes totais poderiam sim ser reduzidas a zero, assim, em vez de apoiar o compartilhamento de carros em si, seria muito melhor uma proliferação de veículos sustentáveis, com consumo eficiente de combustível, ou elétricos.

Conforme a *American Automobile Association* (2008, apud. Cohen; Shaheen e McKenzie, 2008), o CSS pode oferecer uma grande economia pessoal. Pois, em média um veículo para ser mantido por uma pessoa custa em torno de 20% de sua renda, perdendo apenas para o custo em habitação, onde, segundo Cohen, Shaheen e McKenzie (2008) em vez dos usuários pagarem por todos os custos operacionais do veículo, como: seguro, licença, registro, encargos financeiros, manutenção etc., pagarão apenas pelo serviço, pelo tempo e distância que irão dirigir. Portanto os custos operacionais do veículo são compartilhados com um grupo maior de usuários.

O autor complementa dizendo que o CSS custava em média U$ 11 por hora, onde devido a simplicidade, cria uma alternativa acessível para trabalhadores de baixa renda, estudantes e idosos, tendo acesso a veículos confiáveis e com seguro. Dessa forma, ocorrem mudanças coletivas com relação a propriedade de veículos e no comportamento de viagens pessoais, demanda reduzida de estacionamento, onde poderia ser o espaço realocado para obter mais áreas verdes, e redução de emissão de poluentes atmosféricos e gases de efeito estufa. Afirmando que cada veículo de compartilhamento retira das ruas uma média de 15 carros particulares da comunidade, onde estes possivelmente seriam veículos mais antigos, mais poluentes e menos confiáveis na estrada, sendo substituídos por uma pequena quantidade de veículos compartilhados de alta eficiência e baixa emissão, gerando uma grande melhoria no ar local e redução de ruídos e emissões.

“O transporte é um dos principais contribuintes das emissões de CO_2 e outros gases de efeito estufa, representando aproximadamente 27 por cento do total de emissões antropogênicas nos Estados Unidos e 14 por cento no mundo” (Shaheen e Lipman 2007, apud. Cohen; Shaheen e McKenzie, 2008).

2.6. Mercado de dispositivos *mobile*

Não é uma novidade o fato de que os dispositivos móveis, apelidados de *Smartphones* se tornaram parte essencial na sociedade, facilitando as atividades diárias das quais, muitas estão sendo substituídas ou aprimoradas pelas muitas aplicações extraídas dos celulares. Segundo os dados da Apple (2015), são oferecidos pela *App Store* mais de 1,4 milhões de aplicativos para usuários de *iPhone*, *iPad* e *iPod Touch* em 155 países ao redor do mundo, e de diversas categorias, como jogos, redes sociais, esportes, saúde e entre outros. Já seus concorrentes da *Google*, conforme informações recentes da AppBrain (2020), a *Google play* atingiu a marca em junho de 2020 de 2.941.859 aplicativos disponíveis em sua loja.

Esse abrangente mercado tem levado muitas pessoas de volta aos cursos técnicos e superiores em tecnologia da informação, com o objetivo de ingressar nessa área de atuação pela possibilidade de remuneração por meio destes aplicativos, sendo que em 2014, chegou a arrecadar uma receita global de 13 bilhões de dólares conforme Batista e Bazzo (2015).

No segmento automotivo não é diferente, Nobrega (2016) desenvolveu o "*4CAR*", uma aplicação *mobile* com o intuito de lembrar o proprietário do veículo a datas das revisões preventivas através de notificações através de mensagem sonora, utilizando a *Integrated Development Environment* (IDE) de desenvolvimento para *Android* denominada *Android Studio*.

2.7. Sistema Operacional *Android*

Conforme Faustino, et al. (2017) O sistema operacional (SO) *Android* nasceu no ano de 2003, desenvolvido pelos fundadores da *Android Inc*.: Andy Rubin, Rich Miner, Nick Sears e Chris White. A ideia inicial era criar um sistema público, de fácil compreensão dos programadores e que fosse gratuito, com base no *kernel* do SO *Linux*, o *Android* seria aplicado para operar em câmeras digitais, porém percebeu-se que esse mercado não era tão abrangente, levando as atenções da empresa para os smartphones. Devido aos poucos recursos, a *Android* Inc., junto de seus

desenvolvedores foram comprados pela Google para trabalharem numa divisão voltada para celulares.

Apesar de grande parte do mercado do ramo de dispositivos móveis duvidarem da Google dentro deste mercado, a empresa manteve os ideais de Andy Rubin, de um SO *Open Source*, gratuito e de fácil entendimento, sendo essa a característica que tornou o *Android* líder do mercado e acessível as empresas que não tinham seus próprios SO.

2.8. Introdução ao *Bluetooth*

Segundo Câmara (2012), o BT é uma tecnologia de comunicação entre dispositivos de curto alcance, que originalmente foi estudada pela empresa Ericsson que tentava estabelecer um novo tipo de conexão de baixo custo e funcionasse no modo sem fio entre seus celulares e os dispositivos acessórios.

Em 1998 uma parceria entre as maiores fabricantes de telefones móveis do mundo realizou um consórcio e trabalharam em equipe para aprofundar o estudo dessa forma de conexão formando assim o grupo *Bluetooth Special Interest Group.*

Conforme Alecrim (2018)**,** o BT é uma tecnologia de comunicação sem fio que permite que dispositivos troquem dados entre si a partir de ondas de rádio, onde a ideia é que a conexão seja simples, rápida e descomplicada. Ele define o BT como sendo um padrão global de comunicação sem fio e de baixo consumo de energia que permite a transmissão de dados entre dispositivos dentro de um pequeno raio, onde a transmissão de dados é feita por meio de radiofrequência.

O alcance máximo é dividido em três classes, sendo elas:

- Classe 1: Alcance de até 100 m;
- Classe 2: Alcance de até 10 m;
- Classe 3: Alcance de até 1 m.

O autor conclui dizendo, que a classe 2 é a mais utilizada, e que dispositivos de classes distintas podem sim se comunicar sem nenhum problema, bastando apenas respeitar o limite daquele que possui menor alcance.

2.8.1 Frequência e comunicação

De acordo com Alecrim (2018) o BT é uma tecnologia criada para funcionar no mundo todo, motivo pelo qual se usa uma frequência de rádio aberta mundialmente aceita. A faixa *Industrial, Scientific, Medical* (ISM), opera à uma frequência de 2,45 Giga Hertz (GHz), com variações em alguns países que operam entre 2,4 GHz e 2,5 GHz. A faixa ISM é aberta e pode ser utilizada por qualquer sistema de comunicação, onde o sinal transmitido não pode sofrer e nem gerar interferência, para isso é utilizado o esquema de comunicação *Frequency Hopping – Code Division Multiple Access* (FH-CDMA), que realiza a divisão da frequência em vários canais.

O dispositivo que estabelece a conexão muda de um canal para outro de maneira rápida, fazendo com que a largura de banda da frequência seja muito pequena, diminuindo assim as chances de interferência. No BT pode-se utilizar até 79 frequências dentro da faixa ISM, cada uma espaçada por intervalos de 1 Mega Hertz (MHz).

O BT trabalha em modo *full-duplex*, o dispositivo pode tanto receber como transmitir dados, desse modo a transmissão é alternada entre transmissão e recepção de dados, esse esquema é denominado *Frequency Hopping / Time Division Duplex* (FH/TDD).

No que se refere ao enlace, o BT pode operar com dois padrões distintos: *Syncronous Connection-Oriented* (SCO) e *Assyncronous Connection – Less* (ACL).

O SCO estabelece um *link* sincronizado entre o dispositivo emissor e o receptor, assim o SCO acaba sendo utilizado em aplicações de envio contínuo de dados, como por exemplo transmissão de voz.

Em seguida o ACL estabelece um *link* entre o dispositivo, que inicia e gerencia a comunicação e os demais que estão em sua rede. Esse *link* é assíncrono, assim esse padrão acaba sendo utilizado em aplicações que envolvem transferências de arquivo, por exemplo.

2.8.2 Protocolo de transporte, middleware e aplicação

Como em qualquer tecnologia de comunicação, o BT precisa de uma série de protocolos para funcionar. Conforme Alecrim (2018), os mais importantes são basicamente divididos nas seguintes camadas:

- *Radio Frequency* (RF): Camada que lida com os aspectos relacionados ao uso da radiofrequência.
- *Baseband*: Camada que determina como os dispositivos localizam e se comunicam com outros aparelhos BT, é onde os padrões SCO e ACL atuam.
- *Link Manager Protocol* (LMP): Essa camada responde pela comunicação em si, lidando com parâmetros de autenticação, taxas de transferência de dados, criptografia, níveis de potência etc.
- *Host Controller Interface* (HCI): Camada que disponibiliza uma interface de comunicação com *hardware* BT.
- *Logical Link Control and Adaptation Prootocol* (L2CAP): Essa camada serve de ligação com as camadas superiores e inferiores, lida com parâmetros de *Quality of Service* (QoS).

Existem também os protocolos chamados de *middleware* que que possibilitam compatibilidade com aplicações já existentes utilizando protocolos e padrões de outras entidades, como: o *Internet Protocol* (IP), o *Wireless Application Protocol* (WAP) e outros.

2.9. Versões do *Bluetooth*

Alecrim (2018) afirma que o BT é uma tecnologia em constante evolução, fazendo suas especificações mudarem e novas versões surgirem com o tempo.

Ele afirma que o BT 1.0 corresponde as primeiras especificações do BT, onde os fabricantes encontravam problemas que dificultavam a sua implementação, esta versão opera com velocidade de 721 Kb/s. Na versão lançada em 2001 muitos

problemas encontrados na versão anterior foram solucionados. A velocidade de transmissão de dados foi mantida em 721 Kb/s. A versão lançada em 2003 tem como principais novidades conexões mais rápidas, melhor proteção contra interferências e processamento de voz mais avançado. Mas também não houve alteração no limite de transferência de dados.

Segundo Triggs (2018), oficialmente em 2004 surgiu uma nova versão que trouxe importantes aperfeiçoamentos a tecnologia, tais como: diminuição do consumo de energia, aumento na velocidade de transmissão, passando para até 3 Mb/s e melhor comunicação entre dispositivos. Esta versão passou a contar com o padrão *Enhanced Data Rate* (EDR) que fornece uma taxa de transferência mais rápida. Em 2007 surgiu uma nova versão que possui um processo de seleção mais apurado dos dispositivos antes de estabelecer a conexão, melhorias nos procedimentos de segurança e melhor gerenciamento do consumo de energia, também possui compatibilidade com EDR. Já em 2009, surgiu uma nova versão que possuía taxas mais altas de transferência de dados, podendo atingir a marca de 24 Mb/s de transferência. Possui também controle mais inteligente sobre o gasto de energia. As velocidades mais altas da versão só podem ser alcançadas em dispositivos compatíveis com instruções *High Speed* (HS).

De acordo com Alecrim (2018), o BT 4.0 tem como principal diferencial o aspecto da economia, o padrão é capaz de exigir muito menos energia quando o dispositivo está ocioso, característica interessante para telefones celulares que consumiam energia quando o BT não estava sendo utilizado, nesta versão o dispositivo ficaria em modo ocioso com um baixo consumo de energia. Na versão oficializada em 2013, têm como característica a incorporação de recursos que tornam mais receptiva a dispositivo moveis, especialmente aqueles que se enquadram na chamada internet das coisas. A versão que surgiu em 2014 tem pleno suporte ao IPv6, tornando a internet das coisas muito mais relevante, conseguindo assim transmitir dados de câmeras de vigilância, lâmpadas inteligentes, termostatos, tornando assim a tecnologia mais otimizada para a comunicação no mesmo ambiente ou para acesso à internet.

Conforme o autor Triggs (2018), em 2016 foi apresentada a versão 5 do BT, que permitiu que dispositivos se comuniquem em distâncias de até 240m, além disso a velocidade foi aumentada para até 50 Mb/s.

De acordo com Alecrim (2018), o *Bluetooth Low Energy* (BLE) trata-se de uma especificação que faz a tecnologia consumir uma quantidade muito pequena de energia elétrica. Essa tecnologia é aplicada em dispositivos médicos portáteis, *smartwatches* e pulseiras inteligentes, que por serem muito compactos usam baterias de baixa capacidade.

O autor conclui que para consumir menos energia o BLE utiliza várias técnicas, uma delas é a redução na velocidade de transferência, que normalmente não passa de 1Mb/s, outra técnica seria o modo de descanso, que faz com que a tecnologia fique ocioso consumindo pouquíssima energia e realizar uma conexão de poucos milissegundos para enviar ou receber todas as informações. Podemos ver essas informações detalhadas conforme o Quadro 3.

Quadro 3: Especificações do *Bluetooth*

Especificação Bluetooth	1.0	1.1	1.2	2.0 + EDR	2.1 + EDR	3.0 + HS	4.0	4.1	4.2	5
Ano de implementação	1999	2001	2003	2004	2007	2009	2009	2013	2014	2016
Taxa de transmissão (Mbps)	0,7	0,7	0,7	3	3	3	3	3	3	3
Alcance (m)	10	10	10	30	30	30	60	60	60	240

Fonte: (TRIGGS, 2018)

2.10. Sistemas embarcados

Os sistemas embarcados são encontrados em diversos dispositivos eletrônicos. De acordo com Oliveira (2019)**,** um sistema embarcado é definido como um circuito integrado com capacidade computacional que realiza apenas uma tarefa pré - determinada

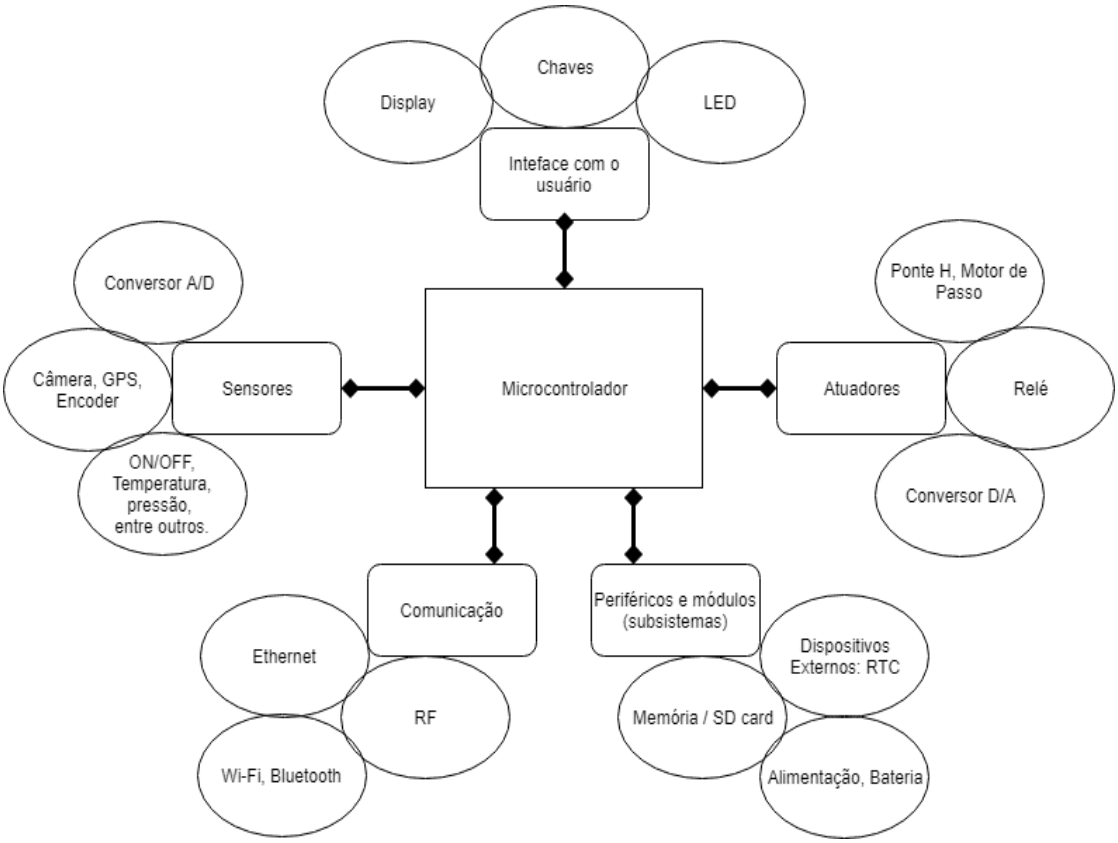
Autor continua afirmando que o sistema embarcado é composto por um processador de dados, que possui periféricos internos e uma programação gravada na memória de dados, e sensores e atuadores, estes o processador irá ler valores do mundo externo e atuar.

O mercado de sistemas embarcados é atrativo devido ao fato de que todos os equipamentos ligados a eletricidade possuem algum sistema de processamento embutido. Segundo Chase (2007), um sistema embarcado por sua natureza pode ter inúmeras aplicações e pode ser classificado como embarcado, quando é dedicado a uma única tarefa e interage continuamente com o ambiente por meio de sensores e atuadores.

A denominação de embarcado (do inglês *Embedded Systems*) vem do fato de que estes sistemas são projetados para serem independentes de uma fonte de energia fixa.

Pode – se concluir, que todo sistema embarcado é composto por uma unidade de processamento, um circuito integrado, que responde aos comandos de um *software* que está sendo processado internamente nessa unidade, logo o *software* está embarcado na unidade de processamento, e todo *software* embarcado é classificado de *firmware*. Pode – se ver na Figura 11 que a unidade computacional recebe informações de sensores e através de cálculos gravados no interior da memória do microcontrolador irá ser tomada decisões que iram impactar diretamente no estado dos atuadores ou até mesmo tomar decisões com base nas informações inseridas pelo usuário através de uma interface de comunicação como por exemplo botões, chaves, ou ser externada alguma informação para que o usuário tenha conhecimento através de *display*s ou *led*s.

Figura 11: Diagrama básico de um sistema embarcado



Fonte: (GARCIA, 2018)

2.11. Internet das coisas

O conceito de internet das coisas segundo Oliveira (2019) tem origem do idioma inglês *Internet of Things* (IoT), e que teve sua primeira aparição por um pesquisador chamado Kevin Ashton em 1999.

Segundo Alecrim (2016) a proposta da IoT é tornar os objetos mais eficientes ou receberem atributos e características complementares, onde dispositivos possam acessar de forma autônoma a internet e interagir com atuadores no mundo real a partir de dados obtidos da rede mundial, ou serem controlados de forma remota por um operador, ou até mesmo um dispositivo comunicar-se com outro, ativando assim configurações personalizadas ou gerando relatórios.

De acordo com Ribeiro (2019), a funcionalidade de um sistema IoT é a de permitir a troca autônoma de informações úteis entre diferentes dispositivos com

capacidade computacional e embutida em uma rede, informações estas obtidas através de sensores que são utilizados para a tomada de decisões autônomas pelos próprios aparelhos, ou auxiliar um humano a fazer escolhas, com ajuda dos dados disponibilizado pelos dispositivos.

3. METODOLOGIA

Neste capítulo será apresentado o desenvolvimento desse projeto, que é constituído de duas partes: aplicativo Android e Sistema Embarcado no veículo. Em ambas as partes haverá uma descrição do processo de construção de cada uma das partes e a sua iteração.

3.1. Aplicativo *Android*

O desenvolvimento do aplicativo *mobile* está fundamentado nos mesmos utilizadas em sistemas de CSS, tendo a funcionalidade de uma locadora de automóveis por hora, em que seriam simulados os comandos dos sistemas de travas elétricas e partida, pelo *smartphone*, através de conexão BT.

Para desenvolver o aplicativo, foi utilizado a IDE *Android* Studio com a linguagem Java para codificação, um smartphone modelo LG-M250ds com a versão do *Android* 24 instalada como emulador, devido aos atrasos do emulador virtual.

O *Android* Studio é um *Software* oficial para desenvolvimento de aplicativos *Android* conforme a Google (2019), que utiliza o *Gradle* como sistema de compilação. Sua codificação pode ser feita tanto em Java quanto em *Kotlin*, porém o sistema possui compatibilidade com C++ e NDK, sendo uma IDE conhecida pelas suas ferramentas que auxiliam na criação de diversas aplicações diferentes, tendo suporte de diversos aplicativos. Alguns arquivos importantes nos projetos com a IDE são as pastas:

- *Manifest*: Contém o arquivo AndroidManifest.xml, que transmite informações diretamente ao SO *Android* e ao compilador;
- *Java*: Arquivos de código – fonte e código de teste do JUnit;
- *Res*: Todos os recursos que não são código, como os layouts XML e imagens em bitmap.

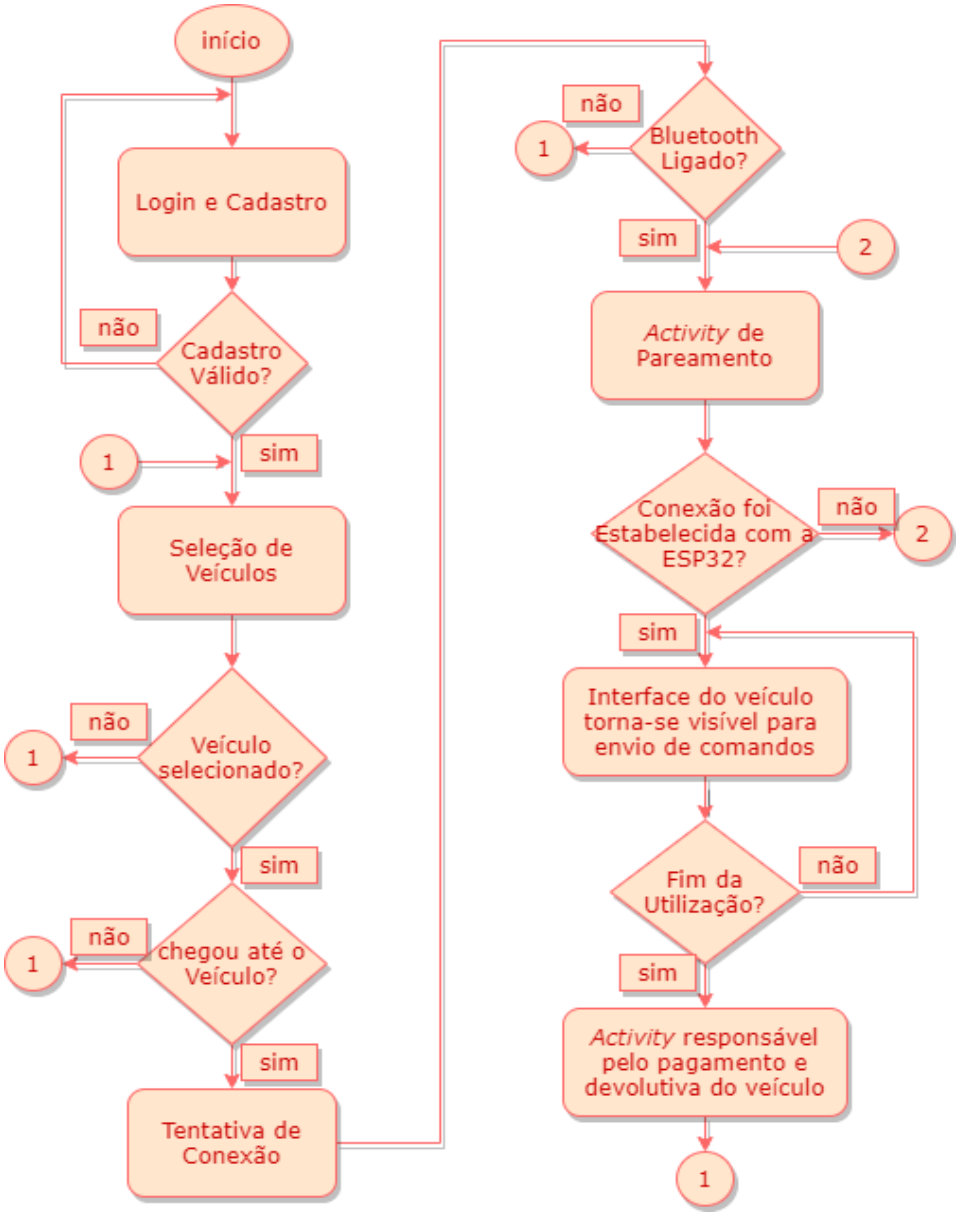
O desenvolvimento do aplicativo, foi dividido em três partes:

1. Criação das *Activity* (ACT) responsáveis pelo Login e cadastro dos usuários do aplicativo;

2. Local onde será selecionado os automóveis registrados no sistema;
3. ACT designadas para pareamento BT com o ESP32 junto da IU de comunicação de envio de comandos ao µC.

Na Figura 12 é possível verificar o fluxograma de funcionamento do aplicativo e a comunicação com a ESP32:

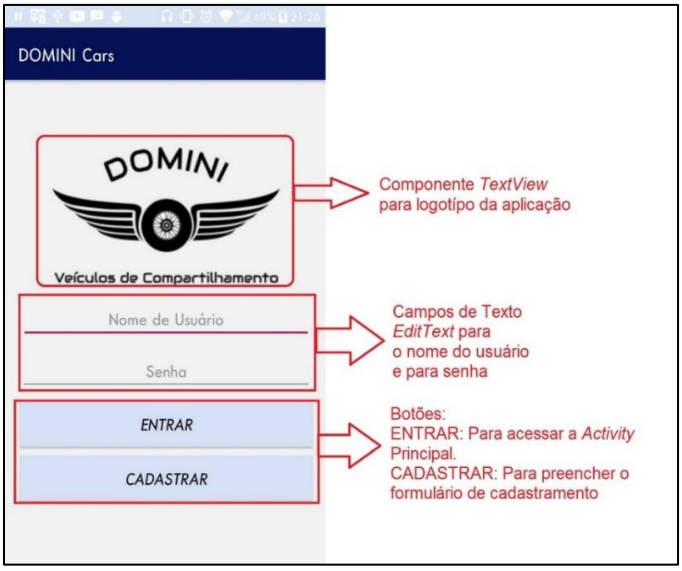
Figura 12: Fluxograma de funcionamento do Aplicativo e da Comunicação



Fonte: Autoria Própria (2020)

A tela de Login é composta por dois componentes visuais de texto editável *EditText* (EDT*)* para digitação do nome e a senha, junto de dois botões chamados: “CADASTRAR”, que aponta através de um *Intent* (IT) para tela de cadastro, e outro denominado “ENTRAR” que leva para atividade principal. A Figura 13 apresenta estes componentes de tela da ACT de Login, junto de um *ImageView* com o logotipo da aplicação:

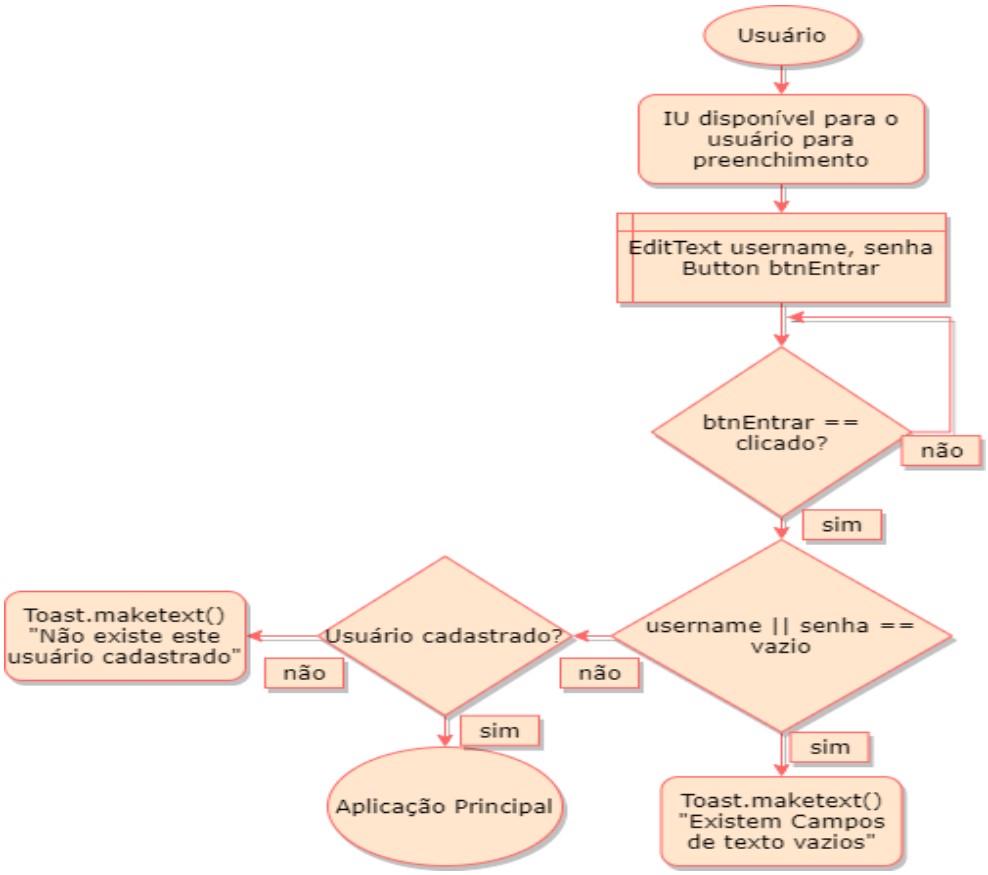
Figura 13: Tela de *Login* de Usuários



Fonte: Autoria própria (2020)

Na lógica desta tela, caso o usuário tente entrar na atividade principal sem ter um cadastro no aplicativo, através do método *Toast.MakeText*() será exibido na tela um texto indicando o erro. Caso algum dos campos de EDT estejam vazios, ou seja, sem preenchimento e o usuário tente entrar, o campo em questão, ou ambos ficarão vermelhos e uma mensagem aparecerá sobre o campo indicando a correta forma de se acessar. O fluxograma na Figura 14 detalha esse funcionamento.

Figura 14: Fluxograma da lógica da ACT de *Login*



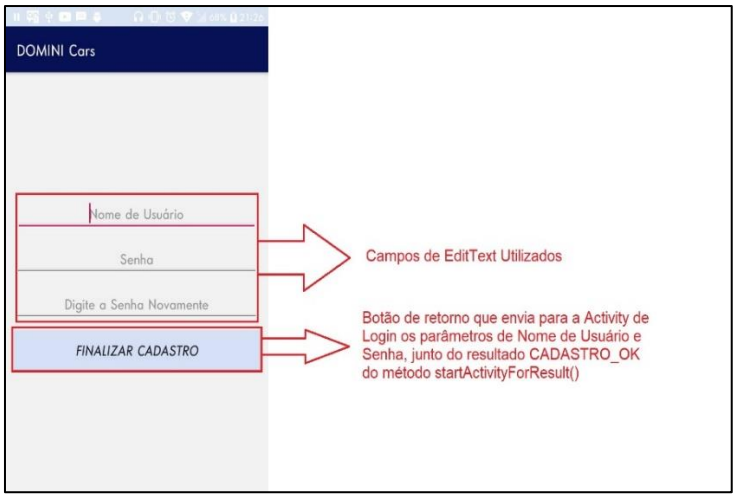
Fonte: Autoria própria (2020)

Dentro da ACT de *Login,* são encapsuladas as informações dos usuários por meio de uma *ArrayList* com os parâmetros enviados pela atividade de cadastro, através do método *startActivityForResullt()* que recebe uma resposta da outra atividade por meio de *setResult*(CADASTRO_OK). O *ArrayList* que contém as informações de cada usuário, compara tudo o que for digitado nos campos de EDT, para permitir apenas aqueles que possuem um cadastro no aplicativo possam utilizar, através do método *contains()*, porém, por não ter um banco de dados, sempre que a atividade de *Login* for finalizada, as informações serão perdidas.

A ACT de cadastro tem uma construção parecida com a tela de *Login*, onde os EDT são para cadastrar os nomes de usuário, a senha e outro para reescrever a senha novamente, verificando se ambas são iguais. Para finalizar a ação de cadastro, foi posicionado o botão de retorno chamado “FINALIZAR CADASTRO”, que envia para a tela de *Login*, os dados do usuário e envia como *feedback* a constante

“CADASTRO_OK” como resultado do método *startActivityForResullt()*. Na Figura 15 estão cada um dos componentes citados:

Figura 15: ACT de Cadastro

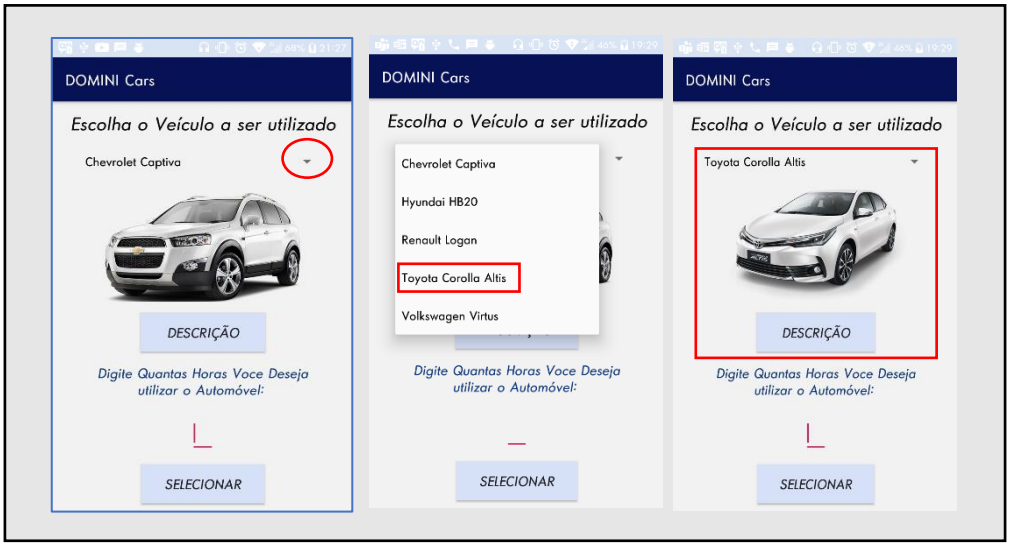


Fonte: Autoria Própria (2020)

Não é possível retornar a atividade anterior se os campos de texto não forem preenchido e se a senha escolhida pelo usuário for menor que seis dígitos, ou haja diferença entre os campos de Senha e Senha reescrita, aparecerá mensagens de erro que notificam e indicam o que o ocasionou para que o usuário se oriente em como se cadastrar de forma correta. Com os dados cadastrados, na tela de *Login* o usuário poderá acessar a atividade principal

Uma vez acessado a ACT de Seleção, o usuário poderá escolher o veículo que desejar. A implementação se inicia com a criação de dois vetores, um vetor de caracteres chamado “nomeCarros” e um vetor de imagens “imagemCarros”, que pelo método pertencente a classe *ArrayAdapter*, denominado *setAdapter*, o vetor de caracteres vinculados aos nomes dos veículos pré-definidos é adaptado para uma caixa de botões *SpinnerBox* que gera uma pequena lista com os nomes de cada carro e por meio do comando *setOnItemSelectedListener()*, que recebe a posição do item clicado, irá apontar para a imagem do veículo naquela posição, tornando-a visível conforme representado na Figura 16. O botão “Descrição” aponta uma *URL* externa ao aplicativo, que contém as informações referentes ao automóvel, como o nome do proprietário e placa.

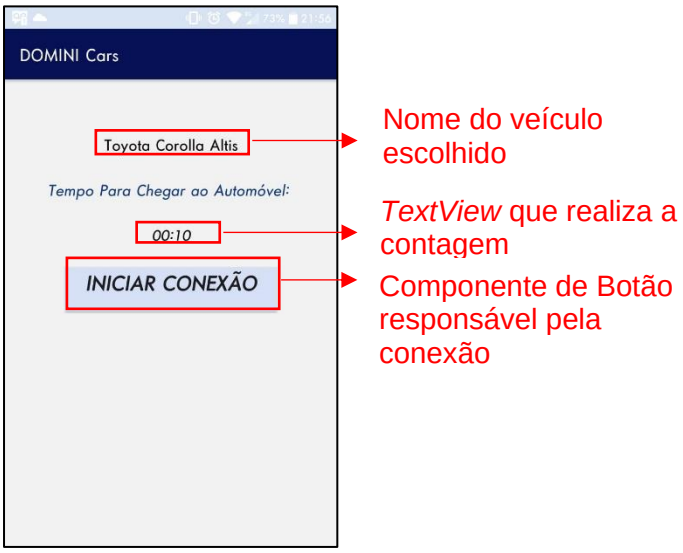
Figura 16: Transição de telas por meio do componente *SpinnerBox*



Fonte: Autoria própria (2020)

Quando a seleção é finalizada uma ACT será inicializada, na qual dentro do projeto a sua função se divide em: limitar o tempo em que o cliente do aplicativo tem para chegar até o veículo, tentando iniciar uma conexão por meio do BT, e a outra de realizar a cobrança pelo tempo de utilização da IU do veículo junto de sua devolução que será abordada mais à frente. Na Figura 17 são mostrados os componentes da ACT temporizadora.

Figura 17: Componentes da ACT de temporização



Fonte: Autoria própria (2020)

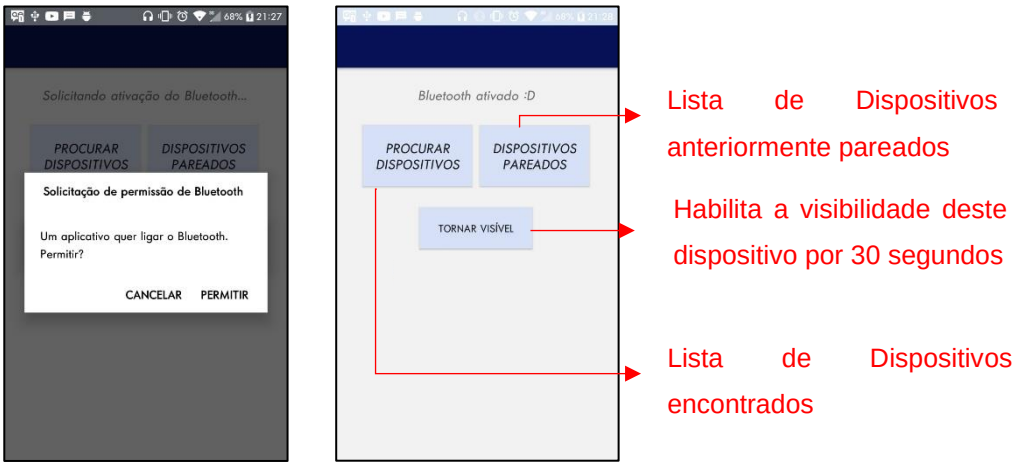
No instante que a atividade se torna visível no *onResume* na primeira vez que é inicializada, se inicia uma contagem de 30 minutos para que o usuário possa chegar até o veículo e solicitar o acesso por meio do botão “INICIAR CONEXÃO”, que só será permitido, caso o BT do celular do usuário esteja ligado, do contrário, será enviado de volta ao menu de seleção do veículo, junto de uma mensagem *Toast.makeText* indicando falha de conexão.

Para realizar a contagem, foi criada uma classe *MyCountDownTimer,* que tem por herança os métodos de *CountDownTimer*, com o propósito de configurar os contadores que seriam utilizados no projeto. Essa classe possui dois comandos importantes conhecidos como: *onFinish()* que é chamado ao fim da temporizador e *onTick()* que recebe na sua inicialização as bases de tempo desejadas (no nosso projeto, para realização dos testes foi utilizado 30 segundos como base de tempo fixo), junto de uma *TextView* (TXV) que torna o contador visível na IU e decrementa a variável *timerChegarAoVeiculo.* Caso a tentativa de conexão não inicie antes da contagem se encerrar, a aplicação irá retornar para o menu de seleção, indicando que o usuário não chegou ao veículo à tempo.

Uma vez clicado o botão “INICIAR CONEXÃO”, será iniciado por meio de um IT a interface relacionada aos controles do veículo, mas que só será permitida, quando o pareamento com o µC for estabelecido. Por meio da classe *BluetoothAdapter* o

aplicativo irá pedir a permissão do usuário para habilitar o BT do dispositivo. Estando habilitado, a ACT responsável pela conexão e a IU será disponibilizada, onde dois botões são responsáveis por listar os endereços de BT nas proximidades ou que em algum momento estiveram conectados com o celular do usuário, e um último que controla a visibilidade desse dispositivo a outros. A ativação e os componentes estão exemplificados na Figura 18:

Figura 18: Permissão de ativação do BT e botões de pareamento



Fonte: Autoria própria (2020)

Quando estabelecida com sucesso, o usuário terá a permissão de acessar os controles da IU do automóvel. Todo o estado do pareamento será mostrado a partir de um TXV inicialmente vazio.

Duas *ListActivitys* foram usadas para armazenar os nomes e endereços dos dispositivos em forma de lista: “DescobrirDispositivos” e “DispositivosPareados”. Suas codificações são muito semelhantes, inflando um arquivo XML padronizado para este tipo de atividade por meio do comando *Layout.inflater*.

A realização da descoberta de dispositivos, acontece por meio do método *startDiscovery()* que inicia o processo de descoberta e filtra através de um *IntentFilter* quando encontra, definindo em seguida um receptor para o evento por um *BroadcastReceiver*. Ao fim dessa TXV, é retornada uma constante a atividade da Interface do veículo indicando que houve a tentativa de conectar-se com outro aparelho (a conexão BT dessa aplicação permite encontrar dispositivos diferentes,

não obrigatoriamente apenas os utilizados nessa aplicação), sendo que todo resultado referente ao estado da conexão é indicado por um TXV.

Para a lista dos dispositivos anteriormente pareados, o adaptador irá pelo comando *getBondedDevices()* obter o uma listagem dos nomes e endereços dos aparelhos registrados, independente do estado da conexão. Quando um destes aparelhos for escolhido, da mesma forma que a ACT anterior, retornará uma constante.

O *Android* quando se trata de envio e recebimento de qualquer tipo de parâmetros de forma externa, pode causar o travando de uma aplicação quando realizada em sua THR principal. Então para solucionar esse problema, todo o envio de comandos para o µC através do BT será realizada numa THR secundária, mais especificamente por meio da classe *ConnectionThread*, onde seus objetos enviam os caracteres definidos por meio da IU, compactados em um vetor de bytes. Dessa classe é importante destacar o método *write()* que recebe como parâmetro de entrada os bytes referentes as mensagens do aplicativo como saída para a ESP.

Com o pareamento estabelecido, torna-se visível a interface principal do veículo pelo comando *setVisibility(View.Visible)*. Composta por um botão deslizável *SwitchButton* que comanda as travas elétricas e um botão *Start/Stop* para o acionamento do veículo, de acordo com a Figura 19.

Figura 19: Componentes da *interface* principal do veículo

Fonte: Autoria própria (2020)

Todos os comandos são constantes do tipo *string*, onde, durante a alternância dos estados dos botões, será enviado: “TRAVAR_PORTAS”, “DESTRAVAR_PORTAS”, “LIGAR_VEICULO”, “DESLIGAR_VEICULO”, como comandos a serem interpretados pelo µC. Por fim, esta ACT irá retornar à atividade anterior.

O cálculo e a devolução do automóvel são realizados após o cliente usar o veículo pelo tempo que foi por ele pré-definido e retornar a partir da atividade pertencente a interface de comunicação com o veículo, representando que está finalizada a utilização. Nessa condição, o método *onActivityResult* retorna uma constante que desabilita o contador de espera de conexão e remove a sua visibilidade com o comando *setVisibility(View.Gone)*, então os componentes responsáveis pela cobrança e devolutiva do veículo se tornam visíveis através do comando chamado *setVisibility(View.Visible),* a Figura 20 mostra o *layout* do pagamento e devolutiva do automóvel referente ao tempo de 2 horas de utilização.

Figura 20: *Layout* da *interface* de pagamento e devolutiva

Fonte: Autoria própria (2020)

Ao fim da parte principal do aplicativo, houve a alteração da forma com que os usuários eram gerenciados, por meio do banco de dados interno. A ferramenta utilizada para essa mudança foi o SQLite, para permitir a criação de contas e a sua validação de forma mais robusta em relação ao método anterior, já que passou a armazenar o *Login* do usuário mesmo quando o aplicativo é fechado. Na prática os *layouts* das ACT de *Login* e cadastro não foram alteradas, porém foi necessário a inclusão de uma nova classe “BDUsuarios” que herda atributos e métodos de *SQLiteOpenHelper*. Dos métodos obrigatórios, apenas o *onCreate()* foi implementado, pois através dele se torna possível criar ou abrir tabelas em *SQL.*

Através do método “checarUsuarioPorUsername”, na ACT de cadastro realiza a verificação do nome digitado. Caso ele já exista no banco, ele retorna o valor nulo, e uma mensagem indicando que existe um usuário com aquele *Username*, senão encapsula as informações, e retorna num objeto do tipo “Usuário”, indicando registro válido. Após isso, o comando *insert()* armazena as informações no banco de dados.

Na ACT de Login o método “validarLogin” verifica se a senha e o *username* já existem dentro do banco de dados. Caso o retorno seja positivo, o usuário tem a

permissão de acessar o aplicativo principal, caso o valor de retorno seja nulo, a mensagem de erro irá indicar o não cadastro das informações digitadas.

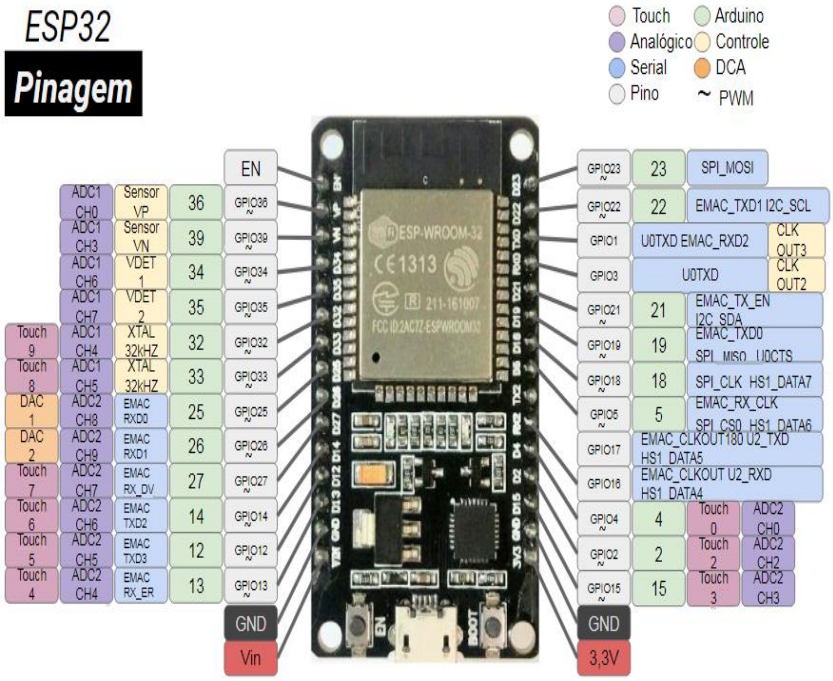
3.2. Sistema Embarcado

O sistema embarcado no veículo é composto por um microcontrolador ESP32 e um relés para simular a abertura do veículo e controle de partida

De acordo com a fabricante Espressif Systems (2020) o ESP32 é um µC que possui alto desempenho, baixo consumo de energia, conexão Wifi (*Wireless Fidelity*) padrão 802.11 b/g/n e BT v4.2 *Basic Rate* (BR) com EDR e BLE, além de trabalhar com transmissão em classe 1, 2 ou 3 sem precisar de nenhum amplificador externo. O chip possui também 4MB de memória flash, CPU *dual-core,* 448 KB de memória ROM e 520 KB de memória RAM, 36 portas, onde 18 podem ser utilizadas como conversor analógico-digital de 12 bits. A sua tensão de operação é de 3.3V.

Para este projeto utilizaremos o ESP32 DEVKIT V1 DOIT, que é a plataforma de prototipagem baseada no ESP32, a placa já conta com conversor USB serial integrado, porta micro USB para alimentação e programação, antena embutida, regulador de alimentação, podendo ser alimentada de 4,5V à 9V. A Figura 21 contém a pinagem do µC.

Figura 21: Pinagem ESP32 DEVKIT V1 DOIT



Fonte: (BLOG DA CURTO, 2018)

3.3. Desenvolvimento do *Hardware*

O µC possui um regulador AMS 1117 – 3.3 de tensão para que possamos alimentar ele com o conector micro USB, de acordo com o fabricante Advanced Monolithic Systems (2020?), ele entregará na saída uma tensão de 3,3V quando alimentado por uma tensão de 4.8V à 10,3V, onde o cabo micro USB fornecerá 5V para alimentação do componente.

Colocamos um *led* na saída do regulador para indicar que o µC está sendo alimentado.

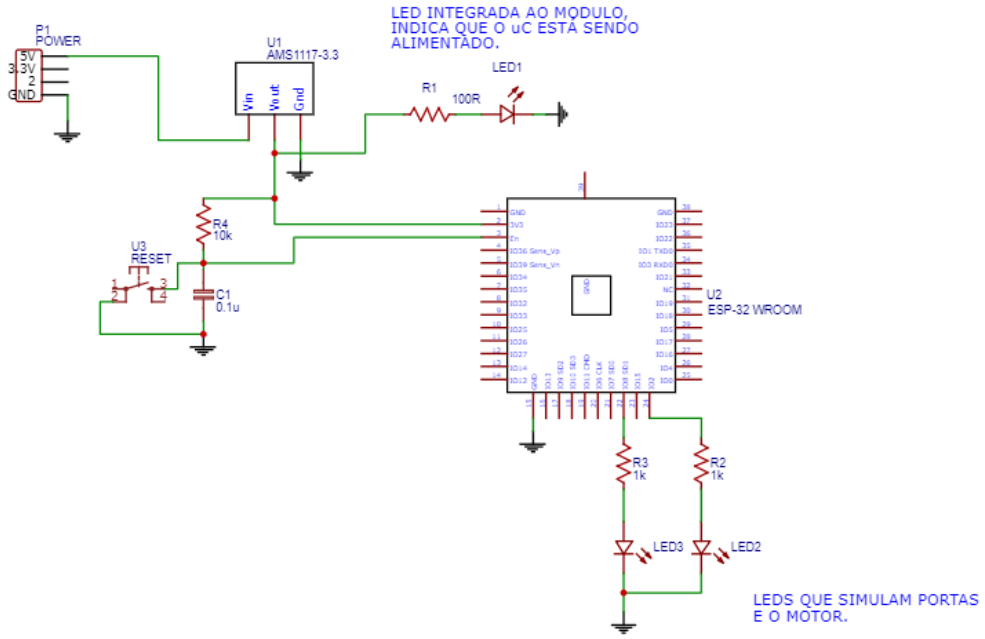
Utilizamos no nosso projeto duas *leds* para simular as condições do veículo, um para fazer a simulação das portas e outro o motor, instalados nos pinos 22 e 24.

O *led* 3acende para indicar que o veículo foi trancado e apaga para indicar que o veículo foi destrancado.

O *led* 2 acende para indicar que o veículo foi ligado e apaga para indicar que o veículo foi desligado.

Na Figura 22 podemos ver o diagrama do *hardware*.

Figura 22: Diagrama do *hardware* ESP32



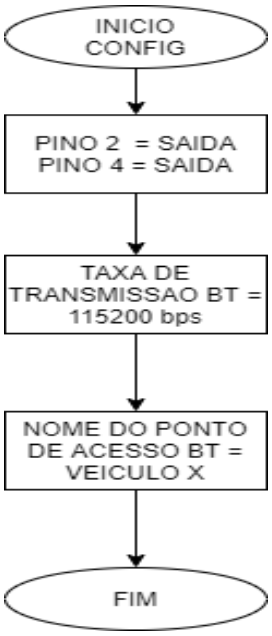
Fonte: Autoria Própria (2020)

3.4. Desenvolvimento do *firmware* do Microcontrolador

A programação do µC foi desenvolvida utilizando a IDE do Arduino. O programa implementado realizará a interface entre de comunicação serial entre o dispositivo *Android* e o sistema embarcado utilizando a ESP32.

Na função de configuração do µC é definido quais pinos funcionarão como saída digital, bem como a taxa de transmissão da comunicação e o nome do ponto de acesso BT, podemos checar isso no fluxograma da Figura 23.

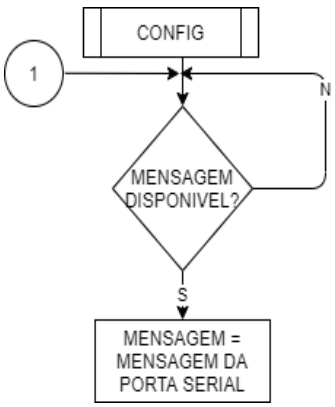
Figura 23: Fluxograma da configuração



Fonte: Autoria Própria (2020)

Na Figura 24, a função principal verifica se existe alguma mensagem através da comunicação serial e se está disponível para leitura, o que está armazenado no *buffer* de recebimento é copiado para uma variável, é checado se há um caractere nulo, que significa que é o fim da mensagem transmitida, caso a condição seja satisfeita, o conteúdo é copiado para uma variável do tipo *String*, caso não seja satisfeita, a variável mensagem continuará em branco.

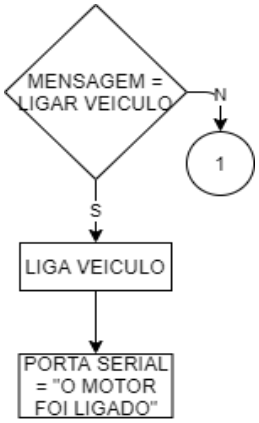
Figura 24: Fluxograma da variável *String*



Fonte: Autoria Própria (2020)

Em seguida, na Figura 25 é mostrado como é comparado se o conteúdo da mensagem que chegou é o mesmo dos comandos que habilitam a interação com o veículo. Caso a mensagem for para ligar o motor, é liberado um comando para o veículo para que ele possa colocar o motor em funcionamento e retornado uma mensagem pela porta serial notificando de que o motor entrou em funcionamento.

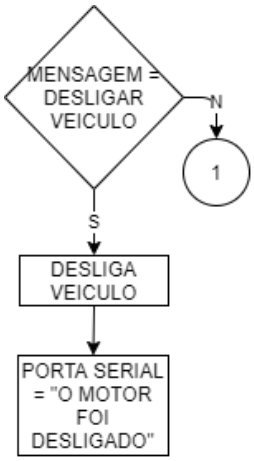
Figura 25: Fluxograma do acionamento do motor



Fonte: Autoria Própria (2020)

Caso a mensagem for para desligar o motor, é liberado um comando para o veículo para que ele possa parar o funcionamento do motor e notificado por meio de uma mensagem que o motor parou seu funcionamento, conforme a Figura 26.

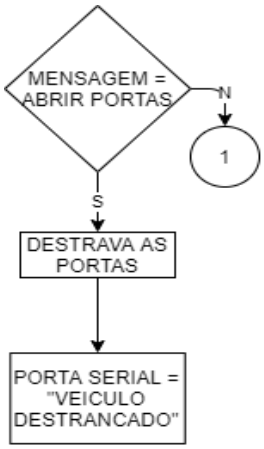
Figura 26: Fluxograma do desacionamento do motor



Fonte: Autoria Própria (2020)

Verifica – se o conteúdo da mensagem recebida, comparando sua semelhança, com os comandos que habilitam o destravamento e travamento das portas do veículo. Caso a mensagem seja para abrir o veículo, é liberado um sinal digital de nível lógico alto para o veículo para que ele permita o destravamento das portas, em seguida é retornada uma mensagem de notificação através da comunicação serial, de acordo com a Figura 27.

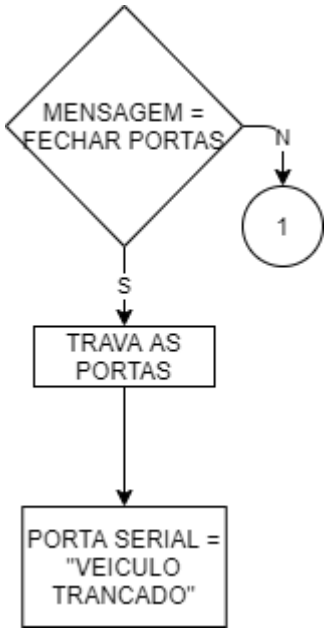
Figura 27: Fluxograma do destrancamento do veículo



Fonte: Autoria Própria (2020)

Conforme a Figura 28, se a mensagem for referente ao trancamento das portas, é liberado um sinal de nível lógico baixo para informar ao veículo que ele deve travar as portas, em seguida é enviado para o dispositivo *Android* uma notificação de que as portas foram travadas.

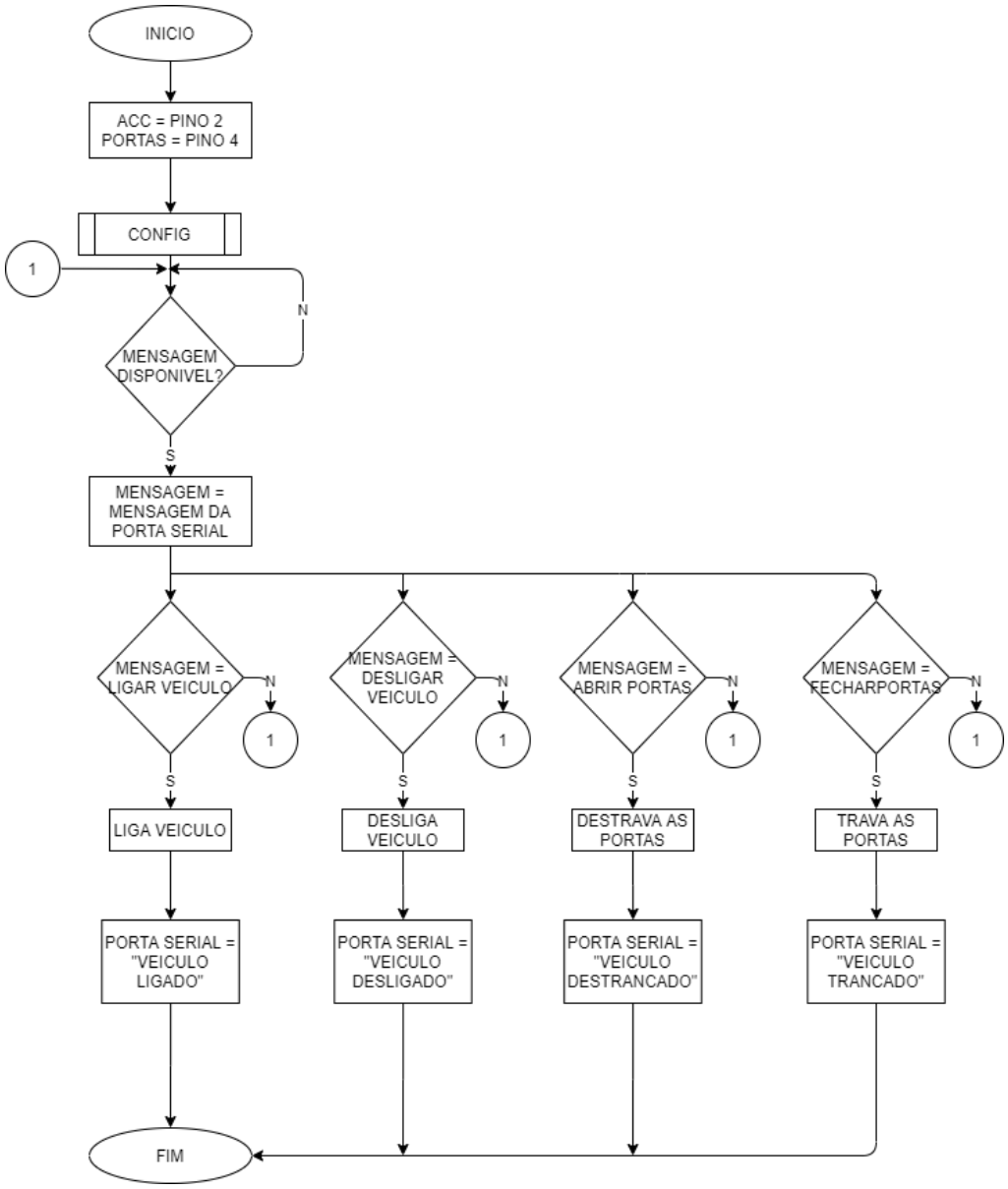
Figura 28: Fluxograma do trancamento do veículo



Fonte: Autoria Própria (2020)

Todo o processo pode ser visto de maneira simplificada através do fluxograma, na Figura 29.

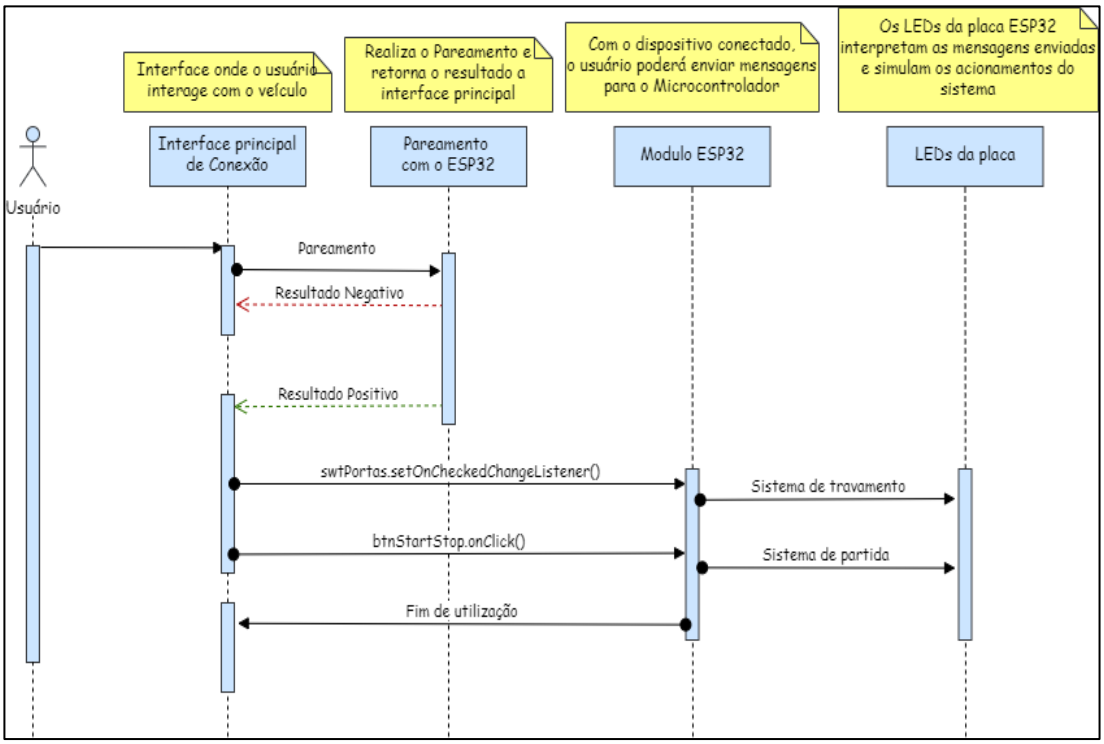
Figura 29: Fluxograma do *firmware* gravado no µC



Fonte: Autoria própria (2020)

O diagrama de iteração da Figura 30, detalha a lógica de iteração *Bluetooth,* entre o aplicativo e o µC. Na ilustração, dentro da interface principal, para que o usuário acesse e utilize a interface de conexão, precisa estabelecer o pareamento com o ESP32. Quando estabelecida, o usuário poderá enviar as mensagens pelos componentes “swtPortas”, responsável pelo acionamento do *led* do sistema de travas, e “btnStartStop”, designado para o *led* do sistema de partida.

Figura 30: Diagrama de iteração entre o aplicativo e o µC



Fonte: Autoria própria (2020)

4. RESULTADOS

Deste projeto, grande parte de sua proposta foi concluída. Seu foco inicial foi a elaboração da pesquisa sobre o funcionamento do CSS, e como essa implementação trouxe benefícios para cidades ao redor do mundo. O propósito disso era para que compreendêssemos a essência daquilo que iriamos desenvolver, sendo este objetivo alcançado com sucesso.

Ao concluirmos esta etapa, definimos as interfaces de desenvolvimento que utilizaríamos para o desenvolvimento do aplicativo e a programação da ESP32. Como tivemos por base o projeto realizado por Granja e Yamawaki (2019) as nossas escolhas seriam focadas em trazer o aprimoramento daquilo que já vinha sendo produzido, logo escolhemos a IDE do *Android Studio* por ser muito utilizado por profissionais da área de aplicações *mobile*, e para o código da ESP, a IDE Arduino com a codificação em C.

Foram definidos alguns pontos para que nos guiássemos durante a implementação: o aplicativo deveria simular um sistema básico de compartilhamento veicular, escrito em Java. Ele deverá permitir a seleção, uso e devolução do automóvel selecionado, operando junto de um banco de dados externo que gerenciasse os usuários e os dados pertencentes ao veículo. Por fim a comunicação seria via BT com a placa ESP32, por meio de comandos das quais ela interpretaria.

Inicialmente os testes seriam realizados no veículo e tentaríamos comunicar via CAN, devido a pandemia do COVID 19, deixamos como uma proposta futura.

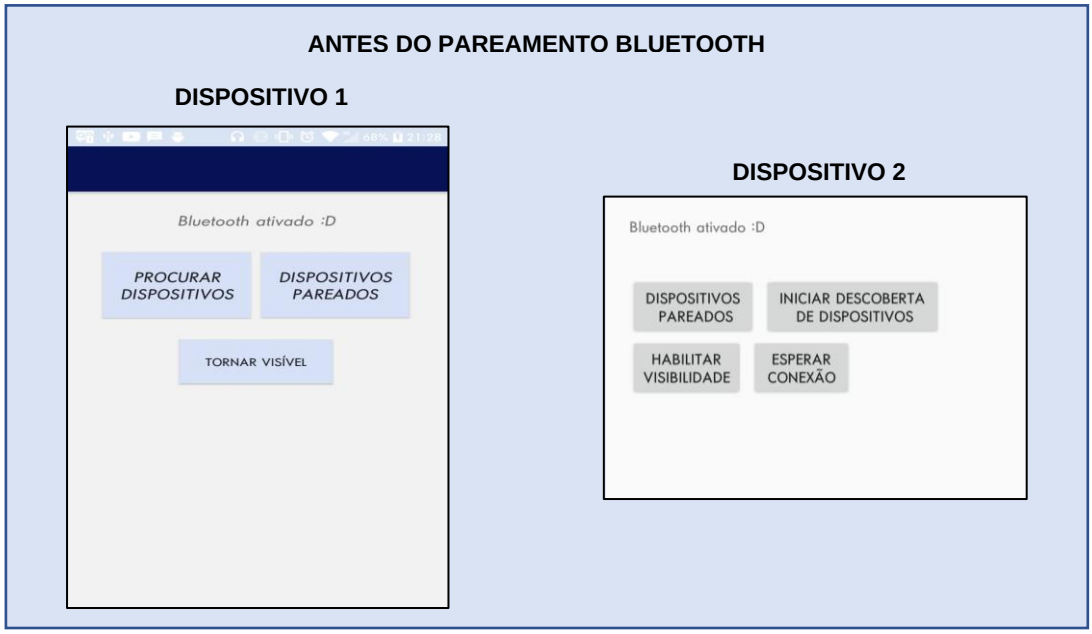
A partir deste ponto, dividiu-se as tarefas entre os orientados, e começamos a enfrentar nossos primeiros desafios, já que a única linguagem que tínhamos familiaridade é a C. Para poder dar andamento no aplicativo, parte do tempo do desenvolvimento foi voltado a aprender os fundamentos da linguagem Java e programação orientada a objeto. Isso, somado ao tempo de familiarização com a IDE, acabou por reduzir o tempo de produção além do planejado, o que resultou na falta de um banco de dados externo. Mas para mantermos a proposta inicial dentro do proposto a solução foi utilizar a ferramenta do *Android, SQLiteOpenHelper.* Esta classe é de vital importância para qualquer base de dados dentro do *android*, já que muitos profissionais recomendam sua utilização, mesmo durante o uso de um servidor

externo. Ela permitiu que mesmo de forma simples, fosse adicionado ao projeto um gerenciamento de usuários. Na prática as alterações não modificaram os layouts da aplicação, mas trouxe consigo algumas linhas de código a mais.

De início, o código do µC foi escrito em *Python*, porém o software *upyCraft* acabou apresentando muitos problemas que poderiam atrapalhar nossos testes. No princípio da implementação da programação em *MicroPython,* não foi possível estabelecer comunicação entre o dispositivo *Android* e o µC, não efetuando pareamento *bluetooth*, observando erro de memória flash que gerava conflitos na memória interna sempre que se tentava a comunicação entre os dispositivos, dificultando assim a gravação do código em *MicroPython*, onde necessitaria fazer o reset da memória do µC sempre que se quisesse fazer a gravação do código. Para a solução deste problema foi apagada a memória flash da ESP32 e substituída a IDE pela do Arduino, que aceita a sintaxe em linguagem C.

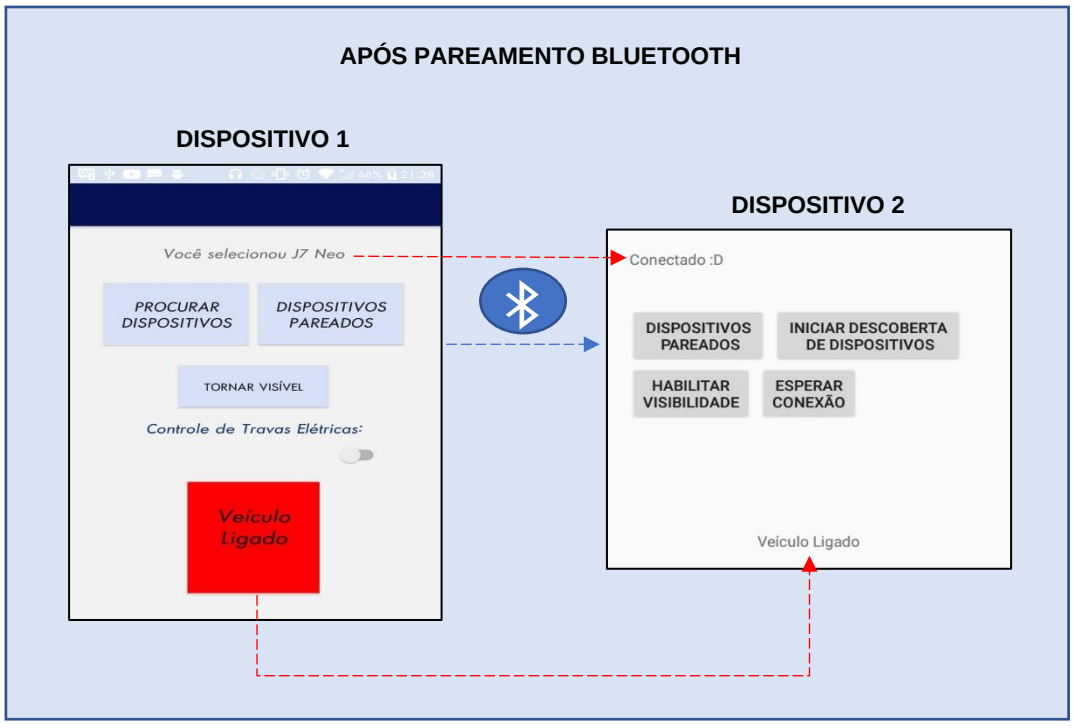
Após o término de todas as etapas, começamos os testes do sistema. Inicialmente o BT, foi testado com um segundo aplicativo, que tinha a função de receber e interpretar as mensagens e através de um TXV, escrever na tela a mensagem enviada pelo usuário. A Figura 31 e 32 são apresentados as ACTs usadas para implementação do primeiro teste. Na primeira imagem, é ilustrado o estado da IU antes do pareamento BT ser estabelecido. Na segunda, Figura 32, temos o pareamento estabelecido e envio de mensagens habilitado, por meio de componentes TXV, o dispositivo 1 escreve o nome do aparelho em que foi conectado, enquanto o componente do dispositivo 2 representa o estado da conexão. Neste exemplo, é demonstrado o envio do comando para acionamento da partida.

Figura 31: Estado do *layout* antes do pareamento



Fonte: Autoria própria (2020)

Figura 32: Pareamento e envio de mensagens

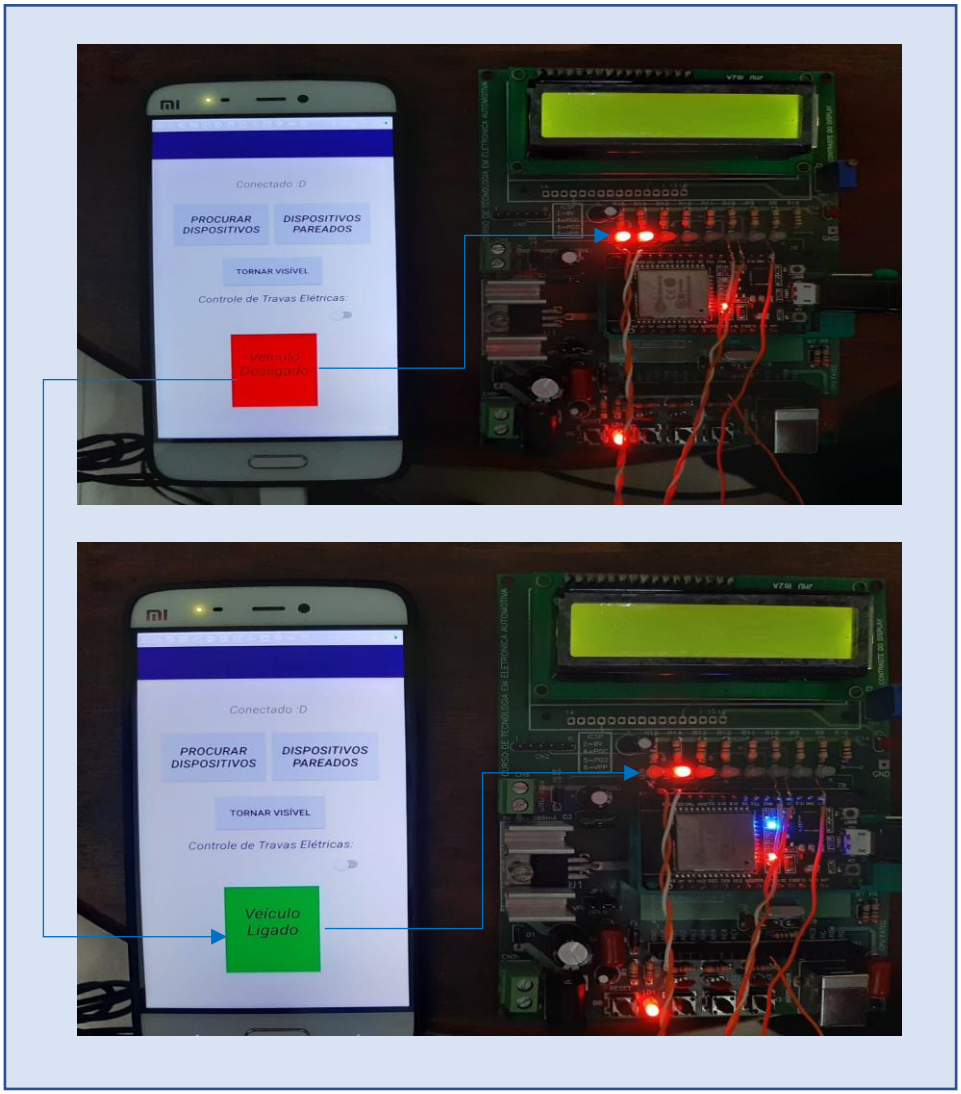


Fonte: Autoria própria (2020)

Para os testes finais, apesar de não podermos nos encontrar pessoalmente, o componente do grupo responsável pela placa do µC instalou o *Android Studio* e por meio do envio do aplicativo, foi possível compilar o código no *smartphone*. Os resultados da conexão BT foram bem sucedidos, onde cada um cos comandos enviados em forma de texto pelo aplicativo foi interpretado pela ESP.

Para demonstrar a leitura dos dados lidos na porta serial da ESP32, a Figura 33 apresenta uma simulação em tempo real, dos acionamentos do sistema de partida, controlado pelo botão “Start/Stop” e do *led* número 1.

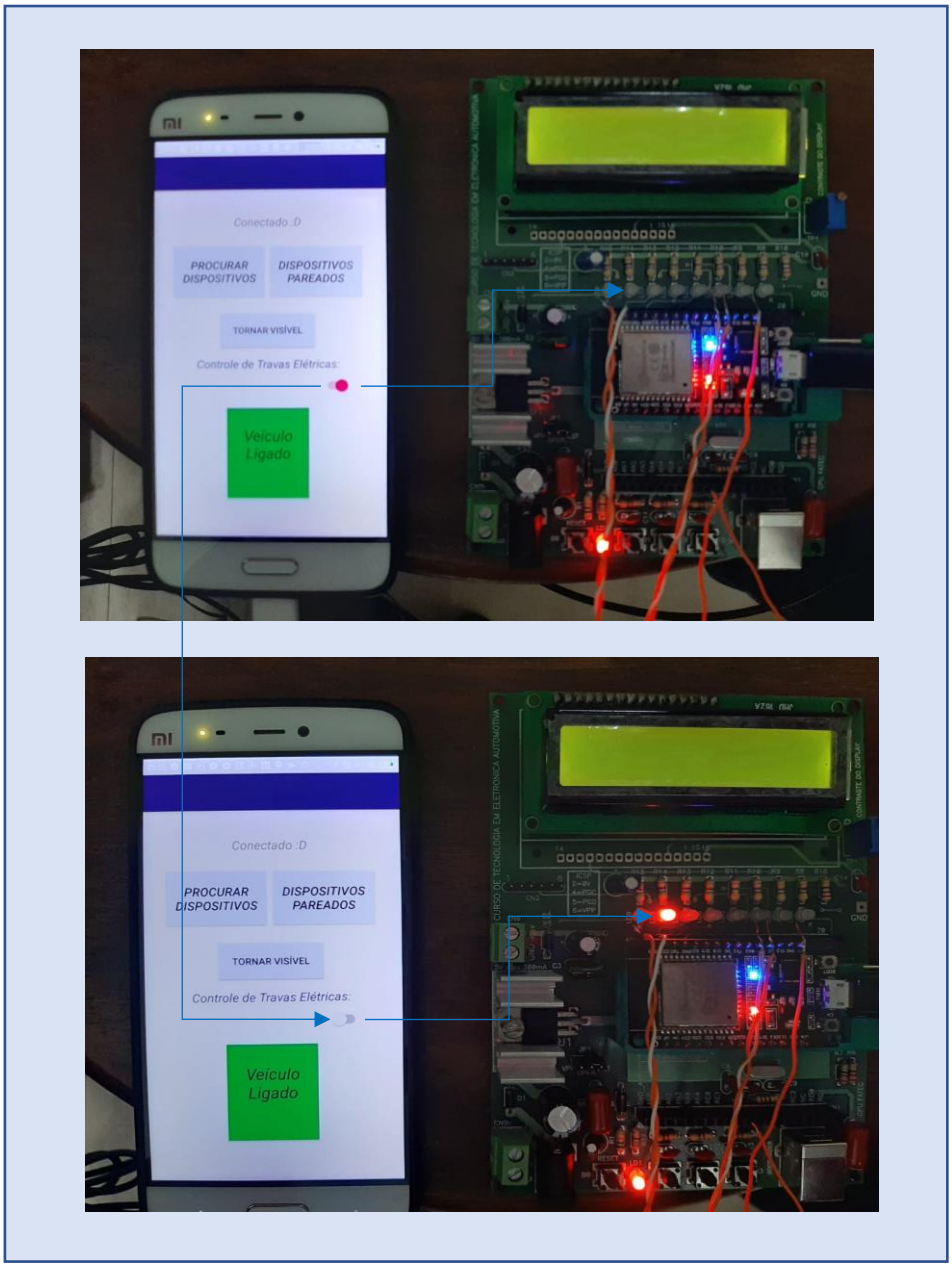
Figura 33: Representação dos acionamentos do sistema de partida



Fonte: Autoria própria (2020)

A Figura 34 ilustra o acionamento do *led* número 2, referente ao sistema de travas elétricas, controlado pelo *SwitchButton*.

Figura 34: Representação dos acionamentos do sistema travas



Fonte: Autoria própria (2020)

4.1. Custo do Projeto

Ao todo, o protótipo desenvolvido demorou cerca de 5 meses. Desde sua idealização, tínhamos a intenção de utilizar ferramentas gratuitas, para reduzir ao máximo o seu custo. No quadro 4 é detalhado cada componente utilizado e seus respectivos preços.

Quadro 4: Custo do projeto

Componentes	Quantidade	Valor Unitário
ESP32	1	R$40,00
Cabo Micro USB	1	R$19,90
Monografia	2	R$45,00
Total Geral		R$104,90

Fonte: Autoria própria (2020)

5. CONCLUSÃO

Com o desenvolvimento do protótipo foi possível verificar a integração *smartphone*/veículo unindo duas figuras do cotidiano em uma mesma aplicação, o que nos proporcionou o aprendizado além da FATEC Santo André, pois tivemos que aprender diferentes tecnologias e como a iteração de diferentes tecnologias pode mudar a forma como nós entendemos o que são os sistemas automotivos.

Vimos através dele que o avanço da tecnologia, nos leva a ter a redefinição de diversos conceitos pré-definidos em nossa sociedade, como por exemplo, o que abordamos em nosso trabalho: a transferência das funções das chaves mecânicas, para uma aplicação *mobile*. Antes, algo completamente mecânico. Isso nos leva a ter o vislumbre de como o avanço científico leva ao surgimento de inovações.

O protótipo realizado, não se trata de algo extremamente novo, afinal, podemos ver diversos tipos de integração das funções do smartphone sendo aplicadas nos veículos. Porém o seu maior propósito era mostrar que essa interação pode ser muito mais aprofundada. Por mais que esse nível de aplicação, ainda seja muito recente e instável para se tornar uma aplicação em veículos em geral, faz parte de um futuro não muito distante e da qual muito provavelmente, estaremos inseridos.

Durante o desenvolvimento do aplicativo, a IDE *Android Studio* nos surpreendeu com sua vasta gama de ferramentas, oferecendo inúmeras possibilidades e formas de se desenvolver projetos profissionais. Neste protótipo, foi explorado apenas uma pequena parte de seu potencial, em vista de que somos inexperientes no ramo de desenvolvimento *mobile*, mas sabemos do incrível potencial que nosso aplicativo pode alcançar com o nosso aprendizado.

Outra ferramenta que obteve resultados satisfatórios, foi a ESP32. Enquanto desenvolvíamos, pudemos ver a variedade de aplicações e funcionalidades desse µC, podendo não apenas utilizar comunicação BT serial, como também Wifi, oferecendo diversas possíveis melhorias para nosso protótipo, como o envio em tempo real das coordenadas de posição para bancos de dados externos. Por fim, é importante lembrar que o ESP32 aceita diversos tipos de linguagem de programação.

Grande parte da proposta inicial foi cumprida com sucesso, apesar das mudanças ocorridas no decorrer do processo de desenvolvimento. Com relação ao

aplicativo, a maioria dos requisitos foram concluídos mesmo com certos pontos a serem melhorados, dentre estas, tornar a IU do aplicativo mais funcional e flexível, oferecendo um ambiente semelhante aos aplicativos *mobile* que estamos mais habituados. Por exemplo, reduzindo o número de ACTs e substituí – las por menus laterais. Outro ponto a ser destacado é o gerenciamento de usuários por servidores externos, permitindo o acesso a uma conta em qualquer dispositivo com o aplicativo instalado. O sistema embarcado funcionou completamente como o esperado, mesmo com a alteração na IDE utilizada. Por fim a integração entre o µC e o aplicativo obteve êxito.

Trabalhos Futuros:

Os principais pontos a serem melhorados posteriormente para dar continuidade ao projeto e/ou deixá-lo mais robusto para assim poder ter um controle mais bem definido sobre a plataforma são os seguintes:

- Implementação de bancos de dados externos, tornando o projeto mais robusto para o gerenciamento da frota e dos usuários, fazendo com que o administrador do sistema tenha dados para poder assim formatar relatórios ou levantar informações para fazer uma boa gestão;
- Comunicação do sistema embarcado pelo barramento CAN ou comutação direta dos atuadores do veículo, bem como a partida do motor de forma remota, a fim de elaborar um circuito capaz de interagir remotamente com o acionamento do veículo;
- Desenvolvimento do software para smartphone utilizando linguagem multiplataforma. Nesse sistema, seria utilizado um *Web* site e um aplicativo *mobile* para o aluguel e monitoramento dos veículos da corporação, onde a chave desse projeto é o monitoramento GPS em tempo real de cada automóvel, realizando a junção com aplicativos externos, como o *Google Maps.*

6. REFERÊNCIAS

ADVANCED MONOLITHIC SYSTEMS. 1A LOW DROPOUT VOLTAGE REGUTATOR. **1A LOW DROPOUT VOLTAGE REGUTATOR**, 2020? Disponivel em: <https://static.chipdip.ru/lib/552/DOC001552809.pdf>. Acesso em: 06 jul. 2020.

ALECRIM, E. **O que é Internet das Coisas (IoT)?**, 7 Março 2016. Disponivel em: <https://www.infowester.com/iot.php>. Acesso em: 8 Junho 2020.

ALECRIM, E. **Tecnologia Bluetooth:** o que é e como funciona?, 29 Agosto 2018. Disponivel em: <https://www.infowester.com/bluetooth.php>. Acesso em: 8 Junho 2020.

APPBRAIN. **Number of Android Aplications**, 04 jun. 2020. Disponivel em: <https://www.appbrain.com/stats/number-of-android-apps>. Acesso em: 2020.

APPLE INC. **Press Info. App Store rings in 2015 with new records**, 2015. Disponivel em: <https://www.apple.com/newsroom/2015/01/08App-Store-Rings-in-2015-with-New-Records/>. Acesso em: 2020.

BATISTA, A. L. F.; BAZZO, A. **Questões contemporâneas e desenvolvimento de aplicativos móveis: onde está a conexão?**, 8, n. 4, 2015.

BERGAMINI, A. D. L. Revista LABVERDE. **TRANSPORTE SUSTENTÁVEL – CIDADE DE SÃO PAULO**, 1, 2014.

BEUREN, F. H.; AMARAL, C. E.; MIGUEL, P. A. C. **Caracterização de um sistema produto-serviço com base no seu ciclo de vida:** análise em um purificador de água disponível no Brasil, 2013.

BLOG DA CURTO. curtocircuito. **CONHECENDO O ESP32**, 2018. Disponivel em: <https://www.curtocircuito.com.br/blog/conhecendo-esp32>. Acesso em: 28 jun 2020.

BREUIL, D.; BARATT, I. **CAR SHARING Deliverable 9.2 of the Success Project**. [S.l.]: Anna Trentini, 2009.

CÂMARA, M. Techtudo. **Bluetooth:** O que é e como funciona, 24 jan. 2012. Disponivel em: <https://www.techtudo.com.br/artigos/noticia/2012/01/bluetooth-o-que-e-e-como-funciona.html>. Acesso em: 5 jun. 2020.

CHASE, O. SBAJovem. **Sistemas Embarcados**, 2007. Disponivel em: <http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf>. Acesso em: 28 Junho 2020.

COHEN, A.; SHAHEEN, S.; MCKENZIE, R. **Carsharing: A Guide for Local Planners**. [S.l.]. 2008.

DANTAS, T. brasilescola. **Bluetooth**. Disponivel em: <https://brasilescola.uol.com.br/informatica/bluetooth.htm>. Acesso em: 5 Junho 2020.

DELOITTE. **Car Sharing in Europe: Business Models, National Variations and Upcoming Disruptions**. Monitor Deloitte. [S.l.], p. 8. 2017.

ESPRESSIF SYSTEMS. **ESP32 Series Datasheet**, 2020. Disponivel em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 05 jul 2020.

FAUSTINO, G. ; CALAZANS, H. K.; LIMA, W. D. Tecnologia em Projeção. **Android e a influência do Sistema Operacional Linux**, 8, n. 1, 2017.

GARCIA, F. D. Embarcados. **Introdução aos sistemas embarcados e microcontroladores**, 2018. Disponivel em: <https://www.embarcados.com.br/sistemas-embarcados-e-microcontroladores/>. Acesso em: 28 Junho 2020.

GOOGLE. Developers. **GOOGLE Inc.**, 27 dez. 2019. Disponivel em: <https://developer.android.com/guide/components/processes-and-threads?hl=pt-br#Processes>. Acesso em: 18 jun. 2020.

GRANJA, J.; YAMAWAKI, M. **VIRTUAL KEY TECHNOLOGY:** Aplicativo para substituição de chaves veiculares. Santo André: [s.n.], 2019.

INRIX. **Global Traffic scorecard Report**. [S.l.]. 2018.

IPPUL - INSTITUTO DE PESQUISA E PLANEJAMENTO URBANO DE LONDRINA. **Taxa de ocupação veicular**, 20 maio 2016. Disponivel em: <https://ippul.londrina.pr.gov.br/index.php/taxa-de-ocupacao-veicular.html>. Acesso em: 28 jun. 2020.

JUNG, J.; KOO, Y. **Analyzing the Effects of Car Sharing Services on the Reduction of Greenhouse Gas (GHG) Emissions**. Seoul National University. Seoul. 2018.

MACHADO, C. A. S. et al. **An Overview of Shared Mobility**. São Paulo: [s.n.], 2018.

MELO, Y. O.; MORO, S. R.; MIGUEL, P. A. C. Produto & Produção. In: ALEJANDRO GÉRMAN FRANK, N. F. A. **Compartilhamento de veículos no contexto de sistema produto-serviço:** análise de uma iniciativa de implementação no Brasil e comparação com sistemas na Europa. Rio Grande do Sul: [s.n.], v. 18, 2017. Cap. 3.

MINDUR, L.; SIERPINSKI, G.; TURÓN, K. Logistic and Transport. **Car-Sharing Development – Current State and Perspective**, 39, 2018.

MOVMI. CARSHARING MARKET ANALYSIS: GROWTH AND INDUSTRY ANALYSIS. **SHARED MOBILITY THOUGHTS**, 17 Maio 2018. Disponivel em: <http://movmi.net/carsharing-market-growth/>. Acesso em: 9 Novembro 2019.

MOVMI; CAR SHARING ASSOCIATION. **CARSHARING MARKET & GROWTH ANALYSIS 2019**, 2019. Disponivel em: <http://movmi.net/carsharing-market-growth/>. Acesso em: 6 Novembro 2019.

MÜNZEL, K. et al. Taylor and Francis Online. **Explaining carsharing supply across Western European cities**, 4 Outubro 2017. Disponivel em: <https://www.tandfonline.com/doi/full/10.1080/15568318.2018.1542756>. Acesso em: 6 Novembro 2019.

NOBREGA, R. D. S. **Android Auto Estudo Para Desenvolvimento de Aplicações Embarcadas**. Santo André: [s.n.], 2016. Acesso em: 06 jun. 2020.

OLIVEIRA, I. F. **DESENVOLVIMENTO DE UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL BASEADO EM IOT PARA CONTROLE E MONITORAMENTO DE DISPOSITIVOS ELÉTRICOS**. Ouro Preto: [s.n.], 2019.

REIS, N. G.; JUNIOR, A. E. B. **CRITÉRIOS PARA A CARACTERIZAÇÃO DE UM SISTEMA PRODUTO-SERVIÇO**. Fortaleza: [s.n.]. 2015.

RIBEIRO, F. T. MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA. **INTERNET DAS COISAS: DA TEORIA À PRÁTICA**, Ouro Preto, 2019. Acesso em: 28 Junho 2020.

SABA, T. H. Caution: Planetary Traffic Jam Ahead. **A world with out traffic Congestion**, 2015. Disponivel em: <https://tasmiahussainsaba.wordpress.com/>. Acesso em: 2 Novembro 2019.

SHAHEEN, S. A.; COHEN, A. P.; ROBERTS, J. D. **Compartilhamento de carros na América do Norte:** crescimento do mercado, desenvolvimentos atuais e potencial futuro. [S.l.]: [s.n.], v. 1986, 2006. 116-124 p.

STATISTA. **Number of car sharing users in Europe in 2014, by country**. [S.l.]: [s.n.], 2014a. Disponivel em: <https://www.statista.com/statistics/415263/car-sharing-users-in-europe-by-country/>. Acesso em: 6 Novembro 2019.

STATISTA. **Number of car sharing vehicles in Europe in 2014, by country.** [S.l.]: [s.n.], 2014b. Disponivel em: <https://www.statista.com/statistics/415280/car-sharing-vehicles-in-europe-by-country/>. Acesso em: 6 Novembro 2019.

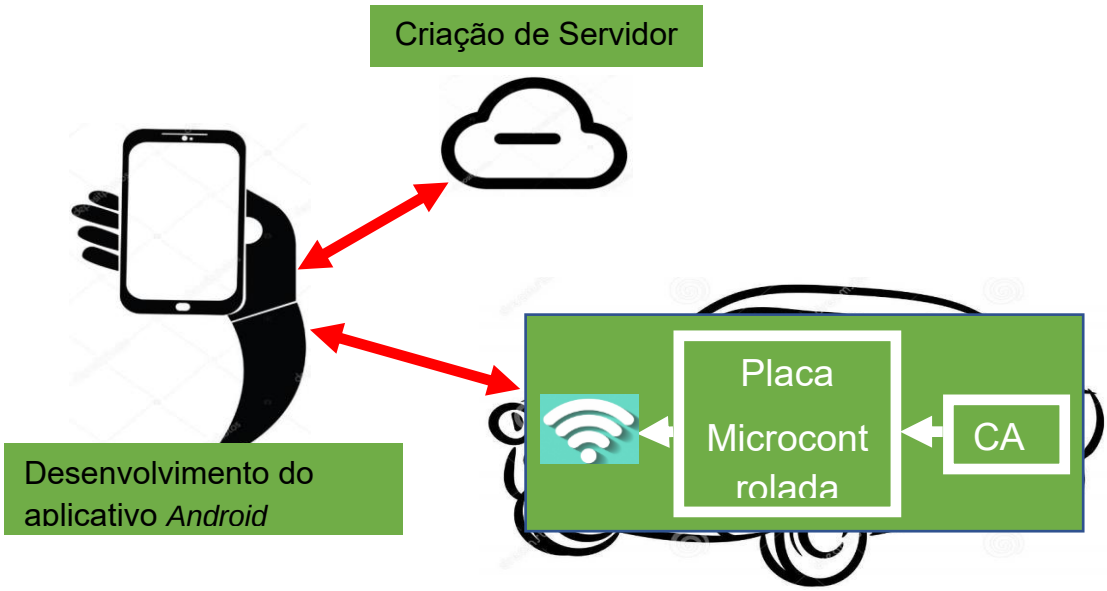
STONE, T. **Lessons Learned from the History of Car Sharing**. [S.l.]: [s.n.], 2013. Disponivel em: <https://tiffanydstone.com/2013/08/23/lessons-learned-from-the-history-of-car-sharing/>. Acesso em: 02 nov. 2019.

TRIGGS, R. Android Authority. **Android Authority**, 23 Março 2018. Disponivel em: <https://www.androidauthority.com/history-bluetooth-explained-846345/>. Acesso em: 28 Junho 2020.

TUKKER, A. **Eight types of product–service system:** eight ways to sustainability? Experiences from SusProNet, 2004.

7. APÊNDICE

Apêndice A: Proposta do sistema de CSS



Apêndice B: Programação contida no ESP32

https://github.com/Brunodt/APP_TCC/blob/master/fw_esp32_tg2.c

Apêndice C: Programação do aplicativo *mobile*

https://github.com/Brunodt/APP_TCC/tree/master/app