CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA
FATEC SANTO ANDRÉ

Tecnologia em Eletrônica Automotiva

BRUNO MAJORES RELA

EDSON RAMOS DA SILVA

GUILHERME ALENCAR RODRIGUES

SISTEMA DE SINALIZAÇÃO E ACIONAMENTOS VEICULARES POR COMANDO DE VOZ

SANTO ANDRÉ

2018

CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA
FATEC SANTO ANDRÉ

BRUNO MAJORES RELA

EDSON RAMOS DA SILVA

GUILHERME ALENCAR RODRIGUES

TÍTULO:

Sistema de Sinalização e Acionamentos Veiculares

Por Comando de Voz

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Santo André como requisito parcial para obtenção do título de Tecnólogo em Eletrônica Automotiva.

Orientador: Weslley Medeiros Torres

SANTO ANDRÉ

2018

R382s

Rela, Bruno Majores

Sistema de sinalização e acionamentos veiculares por comando de voz / Bruno Majores Rela, Edson Ramos da Silva, Guilherme Alencar Rodrigues. - Santo André, 2018. – 104f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.
Curso de Tecnologia em Eletrônica Automotiva, 2018.

Orientador: Prof. Wesley Medeiros Torres

1. Eletrônica. 2. Veículos. 3. Sistemas eletrônicos. 4. Desenvolvimento. 5. Acionamento. 6. Comando de voz. 7. Bluetooth. 8. Pessoas. 9. Habilidades especiais. I. Silva, Edson Ramos da. II. Rodrigues, Guilherme Alencar. III. Sistema de sinalização e acionamentos veiculares por comando de voz.

612.389

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

SANTO ANDRÉ, 20 DE DEZEMBRO DE 2018.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA **"SISTEMA DE SINALIZAÇÃO E ACIONAMENTOS VEICULARES POR COMANDO DE VOZ"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. WESLLEY MEDIEROS TORRES_____

MEMBROS:
PROF. CARLOS ALBERTO MORIOKA_____

PROF. FERNANDO GARUP DALBO_____

ALUNOS:

BRUNO MAJORES RELA Bruno Majores Rela

EDSON RAMOS DA SILVA_____

GUILHERME ALENCAR RODRIGUES Guilherme Alencar Rodrigues

AGRADECIMENTOS

Nossos sinceros agradecimentos ao nosso Professor Orientador Weslley Medeiros Torres, que nos orientou e nos deu um caminho perfeito a ser seguido para a realização deste projeto e por tirar todas as nossas dúvidas durante a montagem do mesmo.

Agradecemos também ao Professor Fernando Garup Dalbo, pela paciência e pelo esforço em auxiliar no nosso projeto em si, no qual a ideia deste projeto se formou, a partir de uma opinião do mesmo.

Agradecemos a Instituição FATEC Santo André por disponibilizar o espaço e os equipamentos no qual foram ideais para a verificação de sinais e valores e para analisarmos o funcionamento do nosso projeto e ao Flavson, funcionário da instituição e colega de classe, que compartilhou uma parte de seu conhecimento conosco durante nossos testes realizados nos blocos.

RESUMO

Este trabalho tem como principal objetivo o desenvolvimento de um sistema que possibilite a execução de acionamentos veiculares pelo usuário através de comando de voz. Este sistema visa o auxílio a pessoas que possuem algum tipo de deficiência física ou mobilidade reduzida nos membros superiores na condução de veículos automotores. Será utilizado para reconhecimento de voz um aplicativo desenvolvido para Smartphone com o sistema operacional Android disponível em mais de 200 milhões de aparelhos em utilização no Brasil em 2018, sendo que a comunicação entre o smartphone que se comunicará com o veículo através do Bluetooth.

Palavras chave: Bluetooth, comando de voz, pessoas com deficiência.

ABSTRACT

The objective of this work is the implementation of vehicular adaptations for the physically disabled and development of a system which enables the execution of common commands done by the driver inside the vehicle through voice commands. This system aims to assist people who have some type of physical disability or reduced mobility in the upper limbs in the driving of motor vehicles. It will be used for voice recognition an application developed for Smartphone with the Android Operating System available in more than 200 million devices in use in Brazil in 2018, the communication between the smartphone, and the vehicle will be stablished through Bluetooth.

Keywords: Bluetooth, voice command, disabled people

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APK	*Android Package*
CET	Companhia de Engenharia de Tráfego
CID	Código Internacional de Doenças
CMOS	*Complementary Metal Oxide Semiconductor*
CNH	Carteira Nacional de Habilitação
DETRAN	Departamento Estadual de Trânsito
DEX	*Dalvik Executable*
DSL	*Domain Specific Language*
FGV	Fundação Getúlio Vargas
I2C	*Inter-Integrated Circuit*
IBDD	Instituto Brasileiro dos Direitos da Pessoa com Deficiência
IBGE	Instituto Brasileiro de Geografia e Estatística
ICMS	Imposto sobre Circulação de Mercadorias e Serviços
IDE	*Integrated Development Environment*
IOF	Imposto Sobre Operações Financeiras
IPI	Imposto Sobre Produtos Industrializados
IPVA	Imposto Sobre Propriedade de Veículos Automotores
RAM	*Random Access Memory*
SSP	*Serial Port Protocol*
USART	*Universal Synchronous Asynchronous Receiver Transmitter*
WCED	*World Commission on Environment and Development*

SUMÁRIO

1 INTRODUÇÃO

Pesquisas demográficas são realizadas no Brasil desde 1872 através do Censo e têm a função de enumerar a população brasileira, extraindo informações sobre suas características e como eles vivem. O Instituto Brasileiro de Geografia e Estatística (IBGE) é o órgão responsável por fazer este tipo de coleta de dados no Brasil, ultimamente obedecendo uma periodicidade de um recenseamento a cada dez anos, fazendo com que sejam levantadas informações socioeconômicas que são extremamente importantes para o governo e para a sociedade como um todo (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA, 2016).

Características pessoais como sexo, condição no domicílio, idade, cor ou raça são amplamente investigadas. Os agentes do Censo são os responsáveis pela aquisição dos dados coletados. Eles têm a missão de falar com pelo menos uma pessoa de cada residência do Brasil, baseando-se em questionários previamente formulados que são utilizados para a classificação da população. Em seguida é feita a apuração dos dados e a criação dos bancos de dados de acordo com as características coletadas e após a sua conclusão, que costuma demorar mais de um ano, o resultado é publicado (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA, 2016).

Informações sobre portadores de algum tipo de deficiência física fazem parte da lista de informações adquiridas com esta pesquisa e os números do último censo que foi realizado em 2010 demonstraram que a quantidade de pessoas que estão declarando possuírem algum tipo de deficiência está aumentando consideravelmente, chegando aos mais de 45 milhões de casos, partindo da dificuldade mais leve até as mais severas. Para o Censo 2010, foram identificadas deficiências voltadas para a visão, audição e os casos mais graves de mobilidade, pois afetam diretamente às políticas públicas voltadas para esta parcela da população (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA, 2012).

De acordo com o Art. 227 § 2º da Constituição da República Federativa do Brasil de 1988, todos os residentes no território brasileiro têm direito à livre locomoção. Com isso estabeleceram-se normas para construção dos logradouros e dos edifícios de uso público e de fabricação de veículos de transporte coletivo, a fim de garantir acesso adequado às pessoas portadoras de deficiência (BRASIL, 1988).

Desde então, o Ministério Público tem o dever de buscar opções, conforme previsto no Estatuto da Pessoa com Deficiência (Lei nº 13.146/2015), para a inclusão da pessoa com deficiência, com o objetivo de garantir a autonomia, a liberdade, a capacidade de autodeterminação, o respeito e a inserção plena na sociedade (BRASIL, 2015).

Com o aumento da conscientização da população em relação aos direitos adquiridos pelos portadores de deficiência física, conseguiram adquirir grande autonomia buscando total locomoção sem o auxílio pleno de terceiros. A acessibilidade passou a ser promovida através de adaptações feitas em edificações públicas, privadas e particulares, seus espaços, mobiliários e equipamentos urbanos, proporcionando a maior independência possível e dando ao portador de deficiência o direito de ir e vir a todos os lugares que necessitar. Desta forma, foram geradas possibilidades de alcance para utilização, com autonomia e segurança, a estas áreas mencionadas.

No Decreto-Lei Federal nº 5.296 (2004), foram estabelecidos prazos para a elaboração de normas técnicas para adaptação de veículos usados, desenvolvimento de projetos e para fabricação de veículos acessíveis de utilização coletiva (BRASIL, 2004). Com isso, iniciou-se o desenvolvimento de acessórios para instalação em veículos particulares que proporcionariam aos portadores de deficiência uma independência ainda maior para sua locomoção, pois poderiam, além de tudo, ter acesso aos seus veículos e utilizá-los como uma pessoa normal no nosso trânsito de todo dia. Desta forma, diversas adaptações para as mãos e para os pés podem ser feitas de acordo com a necessidade real do cliente, fazendo com que a sua relação homem/carro seja a mais fácil possível para que ele possa, novamente, voltar a desfrutar do prazer que o ato de dirigir proporciona.

Para casos específicos relacionados aos membros superiores, a paralisia nas mãos pode ser utilizada como exemplo, faz com que algum tipo de adaptação para uma boa condução. Para os acionamentos internos no veículo, como sinalização de direção ou até mesmo acendimento dos faróis, em casos mais leves, pode-se utilizar botões em locais de fácil acesso para o motorista. Já para casos mais severos, talvez esta possibilidade não fosse a mais adequada. Fica a pergunta: Que alternativa seria viável para resolução de problemas em que os acionamentos veiculares com facilidade de uso e total segurança?

A *Figura 1* ilustra alguns acionamentos que são possíveis de serem utilizados através do sistema desenvolvido com comandos de voz.

Figura 1: Exemplos de Acionamentos a Serem Empregados

Fonte: Elaborado pelo Autor

O objetivo deste trabalho é o desenvolvimento de um sistema que seja capaz de reunir os acionamentos internos no veículo em um ambiente que possibilite ao condutor utilizá-los quando necessário sem a utilização de suas mãos. Para isso, foi desenvolvido um sistema que trabalha fazendo a comunicação entre um aplicativo para tablet ou smartphone capaz de transformar comandos feitos por voz pelo condutor e comunique-se com uma interface embarcada que se comunique com o aplicativo e converta os comandos em acionamentos para o veículo.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo será abordado todo o conteúdo sobre conceitos relacionados à mobilidade urbana, passando por questões como acessibilidade para portadores de algum tipo de deficiência física e o histórico sobre a evolução das adequações feitas em automóveis, descrevendo de que forma surgiu a necessidade de se fazer adaptações nos mesmos e todas as pesquisas necessárias para o desenvolvimento, versões e revisões do produto, assim como suas modelagens matemáticas e informações de componentes utilizados.

2.1 Mobilidade Urbana

A facilidade de deslocamento de pessoas e bens em uma área urbana é determinada levando em consideração não apenas as características da população de uma cidade. Fatores como dimensão local, a forma como a cidade foi construída, complexidade das atividades e qualidade de serviço de transporte disponível. O acesso de bens, serviços, cultura e conhecimento pelos habitantes de uma cidade deve ser considerada como uma das prioridades pois aproxima os habitantes às suas necessidades.

As transformações que o Brasil sofreu entre os anos de 1940 e 1970 fizeram com que o Brasil deixasse de ser considerado um país exclusivamente agrário para se transformar em um país urbano-industrial. Com isso, houve um processo de migração muito grande de estados do Norte e Nordeste do Brasil para a região Sudeste, principalmente para o Estado de São Paulo em busca de emprego e melhorias na qualidade de vida (CÂMARA DOS DEPUTADOS, 2015).

Com o aumento significativo da população nas grandes cidades, problemas sérios tornam-se corriqueiros no dia-a-dia. Centenas de quilômetros de congestionamento e transporte público completamente lotado em determinados períodos do dia são, apenas, dois exemplos de uma vasta lista de possibilidades que acabam tornando o ato de ir e vir extremamente difícil todos os dias. Com isso, Seabra e Taco (2014) esclarecem que a mobilidade urbana e suas implicações fazem com que a sustentabilidade seja um dos principais desafios para as cidades, pois está representada como uma das principais insatisfações expostas pela população neste século.

Segundo Rubim e Leitão (SANTOS e NOIA, 2015), São Paulo está entre as dez primeiras cidades em que se perde grande tempo em congestionamentos no deslocamento de suas casas até o local de destino, sem contar que a quantidade de combustível que é gasto com o carro ligado e parado no trânsito e a quantidade de gases tóxicos que são lançados na atmosfera, trazem consigo dois problemas extras que são a perda de dinheiro e a própria saúde.

2.2 Acessibilidade

As pessoas que possuem algum tipo de deficiência física enfrentam no seu dia-a-dia uma série de limitações que os impede de ter total autonomia perante a sociedade. Na maioria dos casos, estas limitações podem ser resumidas a problemas que o impeçam de fazer com que tenha plenas condições de participar da sociedade por si só (WAGNER, LINDEMAYER, *et al.*, 2010).

> Pessoas com deficiência são aquelas que têm impedimentos de longo prazo de natureza física, mental, intelectual ou sensorial, os quais, em interação com diversas barreiras, podem obstruir sua participação plena e efetiva na sociedade em igualdades de condições com as demais pessoas (CONVENÇÃO INTERNACIONAL DOS DIREITOS DA PESSOA COM DEFICIÊNCIA, 2012).

O Artigo 5º da Constituição Federal do Brasil (1988) diz que todos, sem distinção de qualquer natureza, são iguais perante a lei, todos os residentes no Brasil têm o direito à igualdade. Desta forma, todas as pessoas que possuem qualquer tipo de deficiência física ou dificuldade de locomoção têm o direito à livre locomoção dentro do território local com total acessibilidade para o mesmo.

Segundo Gomes, Rezende e Tortorelli (2016), acessibilidade pode ser definida como possibilidades e condições de alcance para utilização dos espaços e equipamentos urbanos, das edificações, dos transportes e dos sistemas e meios de comunicação por pessoas portadoras de deficiência ou com mobilidade reduzida com total segurança e autonomia.

Com o intuito de assegurar que as pessoas com deficiência possam gozar ou exercer, em igualdade de oportunidades com as demais pessoas, todos os direitos humanos e liberdades fundamentais, a Convenção das Pessoas com Deficiência (2012) exemplifica como medida fundamental o conceito de Adaptação Razoável, ou

seja, modificações e ajustes necessários a serem feitos em locais de livre circulação da população afim de proporcionar total liberdade de locomoção a todos sem distinção. Porém o conceito de Desenho Universal é uma medida que tende a ser mais eficiente se comparado com as adaptações citadas anteriormente pois pode ser compreendida como concepção de produtos, ambientes, programas e serviços que pudessem ser utilizados pela maior quantidade de pessoas possível, lembrando sempre que casos muito específicos de deficiência ainda necessitariam de auxílios igualmente específicos.

Com a implantação dos conceitos de Desenho Universal ou basicamente de Adaptação razoável, fica a cargo do governo tomar as medidas necessárias para assegurar com que as pessoas com deficiência possam usufruir de sua mobilidade pessoal com a máxima independência. Para isso, é fundamental proporcionar o acesso a tecnologias que contribuam para proporcionar ou ampliar habilidades funcionais destas pessoas, proporcionar o acesso a dispositivos e ajudas técnicas de qualidade e formas de assistência humana ou animal, inclusive tornando-os disponíveis a custo acessível. A principal intenção é propiciar às pessoas com deficiência e ao pessoal especializado uma capacitação em técnicas de mobilidade através do incentivo a entidades produtoras (CONVENÇÃO INTERNACIONAL DOS DIREITOS DA PESSOA COM DEFICIÊNCIA, 2012).

2.3 Direção de Veículos

Os candidatos com deficiência que pretendam solicitar uma Carteira Nacional de Habilitação (CNH) devem, primeiramente, se apresentar em uma junta médica especial onde devem ser apresentados laudos com no máximo 3 meses de validade com o Código Internacional de Doenças (CID) e a descrição do quadro da deficiência. Ele pode solicitar desde que tenha aptidão para passar nos exames necessários e que a capacidade de dirigir não seja interferido pela deficiência (DETRAN - ES, 2015).

Com a carteira de habilitação em mãos, os portadores de necessidades especiais contam com benefícios fiscais na hora da aquisição de um veículo, tais como isenção do Imposto sobre Produtos Industrializados (IPI), do Imposto sobre Operações Financeiras (IOF) e do Imposto sobre Circulação de Mercadorias e Serviços (ICMS). Com relação ao Imposto sobre Propriedade de Veículos

Automotores (IPVA), a isenção varia porque a legislação que regula a isenção do IPVA compete a cada estado (QUATRO RODAS, 2018).

As vendas de veículos para Pessoas com Deficiência estão aumentando ano a ano. Com um salto de 42 mil veículos em 2012 para 139 mil veículos vendidos em 2016, impulsionados pela isenção de impostos, fizeram com que as montadoras passem a adequar modelos para atender à lei. Os veículos precisam ser de valor inferior a R$70.000,00 e existem algumas restrições relacionadas à potência do motor (PCDEF, 2017).

2.4 Adaptações Veiculares

Veículos adaptados são aqueles que sofrem modificações em sua estrutura de funcionamento com a intenção de se adaptarem às necessidades do usuário. Normalmente estas modificações são feitas em pedais de freio, no acelerador e na embreagem e a intenção é que ele consiga dirigir com segurança e comodidade (FARIA, 2015). Buscando dar às pessoas com deficiência a oportunidade de conduzir e mover-se com autonomia e segurança, diversas empresas passaram a fazer parte do ramo de adaptações veiculares, buscando trazer soluções para estes problemas de mobilidade.

Controles auxiliares para pessoas com deficiência em ambos os membros superiores também são amplamente instalados em veículos equipados com transmissão automática e direção hidráulica, pois permite ao usuário a operação de controles auxiliares, tais como chave de setas, luzes, buzina, controle de limpadores de vidros dianteiro e traseiro, sinal de emergência e lavador através do acionamento de botões posicionados dentro da cabine, dependendo das necessidades individuais do motorista.

A *Figura 2* ilustra um sistema que transfere ao volante o comando do pedal do acelerador para acionamento manual, permitindo a utilização do veículo por condutores com perda parcial ou total da função nas pernas. Ele pode ser instalado em carros com transmissão automática com a condução caixa sequencial ou embreagem automática (KIVI BRASIL, 2016).

Figura 2: Acionamento Manual do Acelerador no Volante

Fonte: (KIVI BRASIL, 2016)

Posicionado sob o volante original do veículo, permite ao motorista bom controle em acelerações e frenagens além de bom controle em curvas, mantendo o volante original, não altera a visão sobre a instrumentação e não dificultando a funcionalidade do airbag em caso de possível acionamento (KIVI BRASIL, 2016).

Instalado no volante do veículo, as Centrais de Comando com Pomo funcionam por infravermelho e acionam todos os comandos das alavancas de seta e limpador do para-brisa. Este dispositivo é utilizado quando o condutor possui algum tipo de deficiência motora parcial em um membro superior e pode ser instalado à direita ou à esquerda, dependendo das necessidades do mesmo (KIVI BRASIL, 2016). A central de comandos com pomo mostrada na *Figura 3* é um exemplo para esta aplicação.

Figura 3: Central de Comandos com Pomo

Fonte: (KIVI BRASIL, 2016)

O dispositivo possui um suporte que é instalado no volante do veículo e possui o encaixe para a central. Com o giro do volante, um dispositivo interno da base que fica em contato com a central também gira e a mantém, sempre, na posição vertical. O pomo tem a função de auxiliar o condutor nas manobras de direção e o teclado permite a gestão dos comandos de serviço, tais como sinais de seta, limpadores do para-brisa, farol alto, farol baixo, farolete, buzina e sinal de emergência (KIVI BRASIL, 2016).

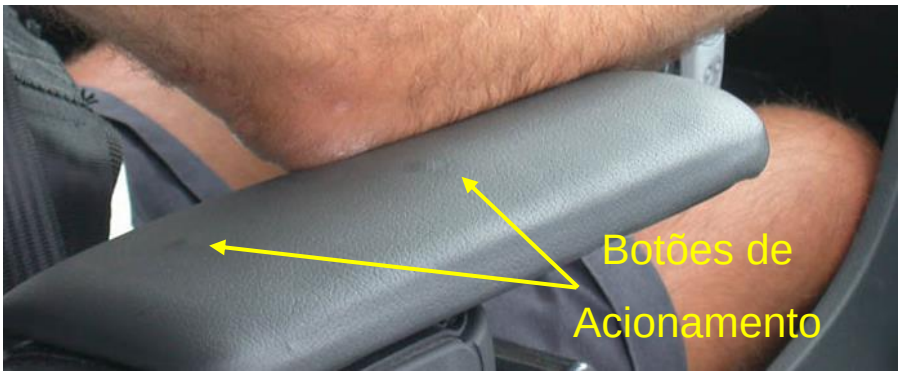
Outro tipo de dispositivo que está relacionado aos indicadores de direção é mostrado na *Figura 4*. Ele é instalado no encosto de cabeça do condutor do veículo para ser utilizado para acionamento das luzes de indicação de direção e para acionamento dos faróis. Para ativar basta um leve toque da nuca e para desativar basta apenas repetir a ação.

Figura 4: Encosto de Cabeça com Acionamentos de Direção e Faróis

Fonte: (KIVI BRASIL, 2016)

Os acionamentos dos limpadores de para-brisa e da iluminação podem ser acionados através de botões que podem ser instalados em locais que facilitem a sua ativação. Desta forma, a variedade de locais onde podem ser instalados os controles para estes comandos pode ser maior. A *Figura 5* mostra um controle instalado no apoio de braço central para acionamento dos limpadores de para-brisa através do cotovelo (KIVI BRASIL, 2016).

Figura 5: Controle no Antebraço para os Limpadores de Para-Brisa



Fonte: (KIVI BRASIL, 2016)

Adaptações que auxiliam o acesso ao veículo também são utilizadas, sendo elas manuais ou automáticas, novamente dependendo bastante do nível de redução da capacidade motora do usuário. O banco de transferência manual apresentado na *Figura 6* possui capacidade máxima de 120kg, podendo ser Manual Removível, onde o motorista pode retirar do local após o embarque pra que consiga fechar a porta do veículo e seguir viagem ou Manualmente Retrátil, que possui uma rotação de 90º no seu eixo, possibilitando o seu recuo para a lateral do banco do veículo (KIVI BRASIL, 2016).

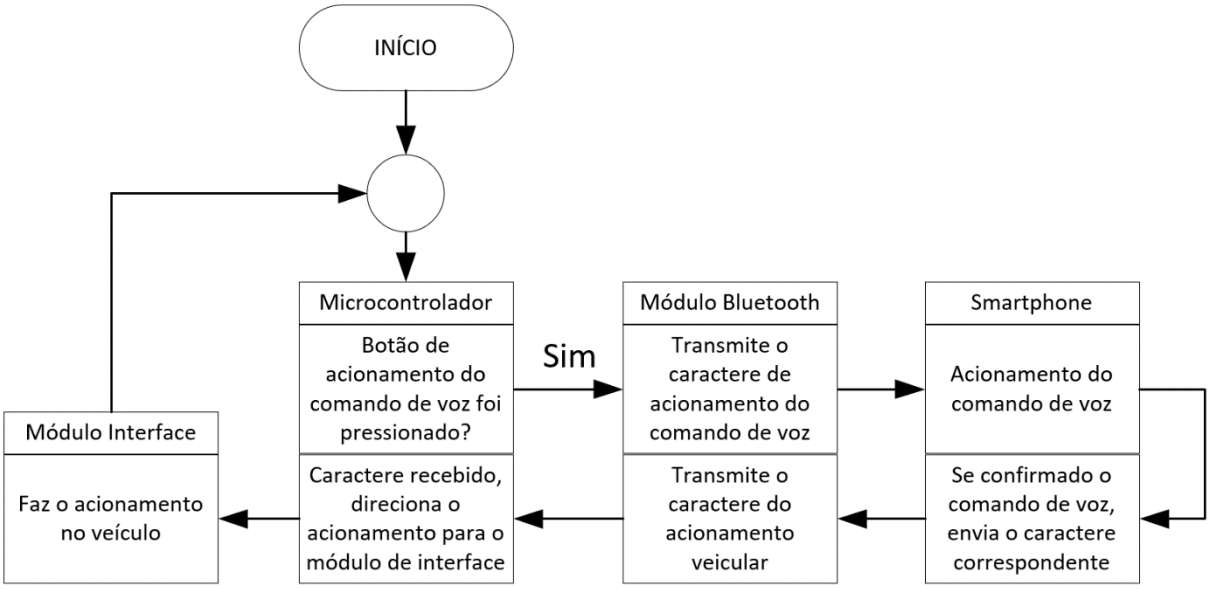
Figura 6: Banco de Transferência Manual RM205

Fonte: (KIVI BRASIL, 2016)

3 METODOLOGIA

Este trabalho consiste no desenvolvimento de um sistema a ser utilizado por pessoas que possuem mobilidade reduzida nos membros superiores. O sistema será composto por um aplicativo desenvolvido para o Sistema Operacional Android capaz de receber os comandos de voz do usuário e fazer a comunicação com o módulo de controle que faz interface com o veículo. O módulo de controle é controlado por um microcontrolador e a comunicação com o smartphone é feita através de um módulo bluetooth modelo HC-06. O diagrama básico de funcionamento do sistema está ilustrado na *Figura 7*.

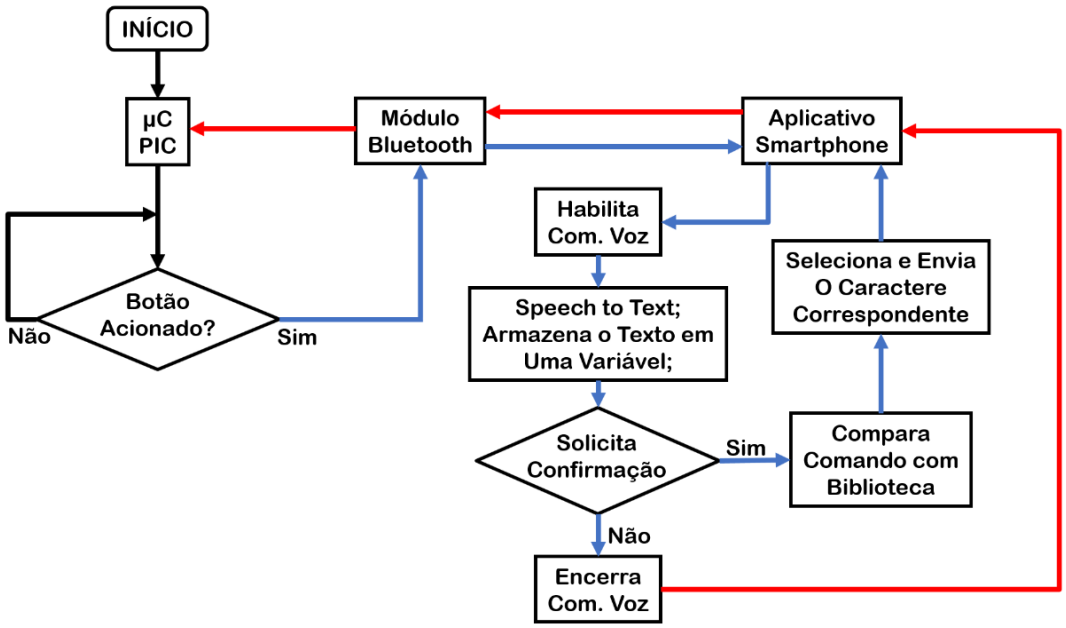
Figura 7: Diagrama de funcionamento do sistema



Fonte: Elaborado pelo autor

Para que o sistema entre em operação, a conexão através do Bluetooth entre a placa de controle e o celular deve ser estabelecida. Esse procedimento irá ocorrer sempre que o usuário inicia o processo de ignição do veículo, dessa forma, o sistema fica em estado de espera, e quando o usuário pressiona o botão do comando de voz, é iniciado o sistema de controle como mostrado na *Figura 8*.

Figura 8: Diagrama de Funcionamento Comando de Voz



Fonte: Elaborado pelo Autor

3.1 Sistema de Reconhecimento de Voz

De acordo com a 29ª Pesquisa Anual de Administração e Uso de Tecnologia da Informação nas Empresas ministrada pela Fundação Getúlio Vargas (FUNDAÇÃO GETÚLIO VARGAS, 2018), o Brasil superou a marca de um smartphone por habitante e hoje conta com 220 milhões de celulares inteligentes ativos, podendo este número ultrapassar os 300 milhões se forem incluídos, nesta lista, os tablets e notebooks.

Como todos os smartphones que possuem o Sistema Operacional Android possuem como função nativa do sistema o reconhecimento de voz através do assistente virtual do Google, o Google Assistente (GOOGLE, 2017), foi decidido por utilizá-lo por ser de fácil acesso e só seria necessário o desenvolvimento da aplicação que trataria os dados da voz recebidos pelo assistente.

A tecnologia de reconhecimento de voz está cada vez mais presente no nosso cotidiano devido à grande quantidade de dispositivos existentes no mercado que fazem uso deste tipo de tecnologia. Inicialmente ligados aos smartphones com poucos comandos específicos, a maioria delas eram perguntas em que o aparelho respondia de acordo com informações captadas pela internet, gigantes da tecnologia como Microsoft e Apple passaram a investir pesado nesta tecnologia e passaram a

desenvolver os seus assistentes virtuais e hoje em dia temos a Cortana (Microsoft) e a Siri (Apple) como bons exemplos para esta tecnologia atualmente.

A Google também tem o seu assistente virtual e é amplamente utilizado em dispositivos que rodam o Sistema Operacional Android. O comando de voz foi implementado pela primeira vez já na primeira versão do Android lançada em 2008, mas a partir de 2008 com o lançamento do Android 4.4 Kit-Kat o assistente de voz passou a ser utilizado para tarefas mais gerais no sistema.

3.1.1 Google Now e Google Assistente

O Google Now é uma funcionalidade nativa do Android desde a sua versão de número 4.1 (Jelly Bean) e esteve disponível para download até 2017, sendo substituído pelo Google Assistente. Este aplicativo foi desenvolvido com a intenção de ajudar a organizar a rotina dos usuários e aumentar sua produtividade através funções que mostram a previsão do tempo, trânsito e notícias de acordo com o perfil de cada usuário (GOOGLE, 2017).

A forma mais simples de utilizar todos os recursos do aplicativo é executando comandos de voz a partir do “Ok Google”. Basta falar essas palavras e começar a interagir com o assistente pessoal. A *Figura 9* mostra a barra de pesquisa do Google em um smartphone com o microfone, símbolo do Google Assistente.

Figura 9: Barra de Pesquisa com Comando de Voz do Google Assistente

Fonte: Elaborado pelo autor

Criar lembretes, verificar rotas ou simplesmente fazer pesquisas são algumas das funções que é possível fazer com este comando. Ultimamente, é possível fazer uma interação mais ágil e profunda com aplicativos de terceiros, fazendo com que, com o passar do tempo, novas funcionalidades possam ser utilizadas diretamente pelo Assistente (TECMUNDO, 2017).

Para conversão da fala em texto pela aplicação de reconhecimento de voz da Google, o dispositivo precisava se conectar com uma central de dados na nuvem

capaz de interpretar a informação que acabou de receber. Desta forma fazia a conversão dos fonemas encontrados na fala e os convertia em texto. Desde antes do lançamento do novo Google Assistente, não é mais necessário a utilização de dados de internet, pois o novo sistema desenvolvido já está totalmente incluso no sistema operacional do dispositivo, fazendo com que a resposta seja muito mais rápida por não ter a necessidade de consultas a bancos de dados na nuvem (CANALTECH, 2016).

Este novo método de reconhecimento de voz utiliza técnicas de aprendizado de máquina como memória de curto e longo prazo e redes neurais recorrentes para que não seja mais necessária a busca de informações na internet. A base de dados utilizada neste método é oriunda de mais de 3 milhões de enunciados do tráfego de pesquisa de voz do Google, somando um total de mais de 2 mil horas de gravações, incluindo amostras sonoras obtidas em vídeos do YouTube (CANALTECH, 2016).

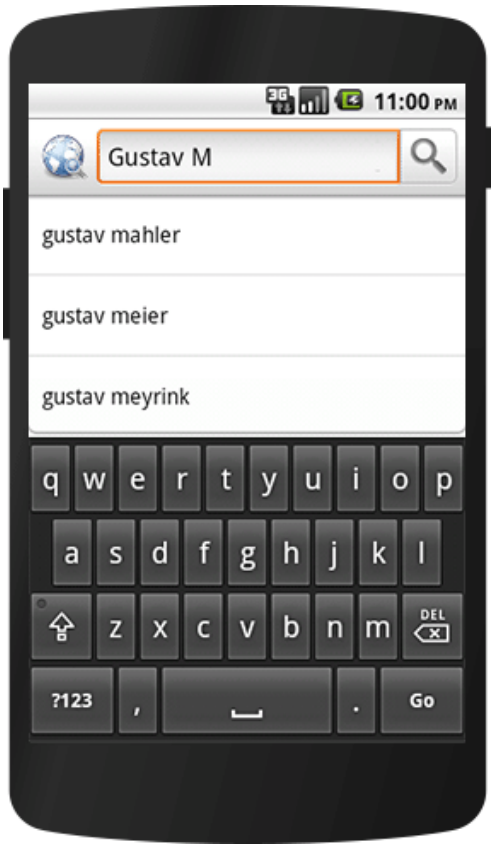
3.2 Android

Nos dias de hoje milhões de dispositivos eletrônicos estão sendo produzidos e comercializados no mundo inteiro. Segundo a Fundação Getúlio Vargas (FUNDAÇÃO GETÚLIO VARGAS, 2018), somente no Brasil em 2018 são aproximadamente 220 milhões de smartphones ativos.

Criado em 2008, o Android é um sistema operacional móvel baseado em Linux que, por ser de código aberto, permite que qualquer fabricante que utilize a plataforma tenha liberdade para usar, modificar e distribuir o sistema em seus dispositivos, fazendo com que fabricantes de dispositivos móveis passassem utilizar o sistema operacional em seus smartphones (OLHAR DIGITAL, 2018).

A primeira versão do Android comercializada foi a 1.6 Donut. Nesta versão, a principal funcionalidade foi a caixa de pesquisa rápida, que tinha o dever de retornar ao usuário os resultados de pesquisa da Web na única caixa contida na tela inicial como mostra a *Figura 10* (ANDROID, 2018).

Figura 10: Caixa de Pesquisa Rápida Android Donut 1.6



Fonte: (ANDROID, 2018)

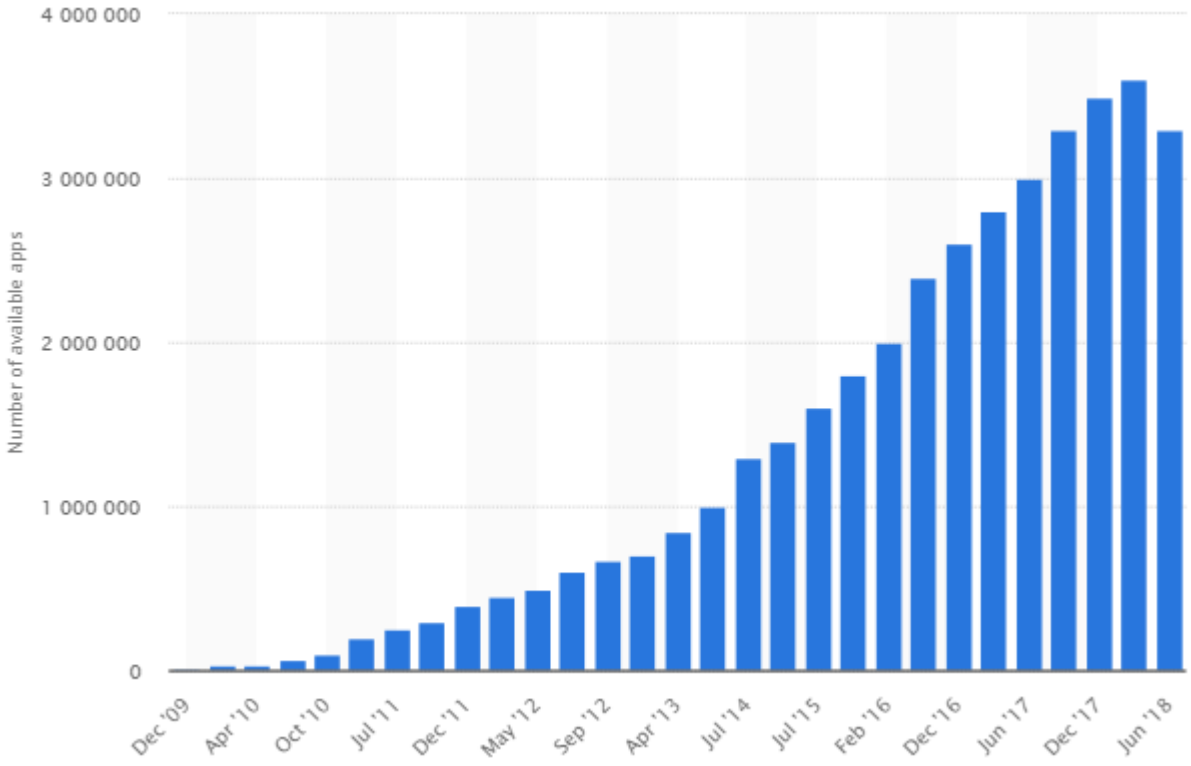
Com a expansão do mercado de smartphones no mundo, surgem também milhões de aplicativos suportados por esta plataforma, sendo que cada vez fica mais fácil encontrar aplicativos que servem para qualquer coisa que esteja precisando. Esta expansão se deu por causa da popularização dos sistemas de desenvolvimento deste tipo de aplicação e que muitas vezes pode ser disponibilizado para o mundo baixar e instalar em seu dispositivo. Em 2008, quando lançado, a loja de aplicativos do Google se chamava Android Market, hoje em dia chama-se Google Play (ANDROID, 2018).

A partir do Android 2.1 Eclair, o Sistema Operacional passou a ser mais interativo, ganhando funcionalidades de Sistemas de Navegação do Google Maps. O sistema de conversão de voz em texto também passou a ser desenvolvido a partir desta versão (ANDROID, 2018).

A partir de 2012 quando foi lançado o Android 4.1 Jelly Bean, o Android inaugurou uma nova era pra o sistema operacional, pois deu início à assistência móvel personalizada com o Google Now (ANDROID, 2018). A partir daí o desenvolvimento de aplicações para o Android disponíveis na loja oficial da empresa só vem crescendo

todos os anos, saindo de 100 mil aplicativos em 2010 para cerca de 3 milhões em 2018 como mostrado no *Gráfico 1* (STATISTA, 2018).

Gráfico 1: Evolução do Número de Aplicativos na Play Store



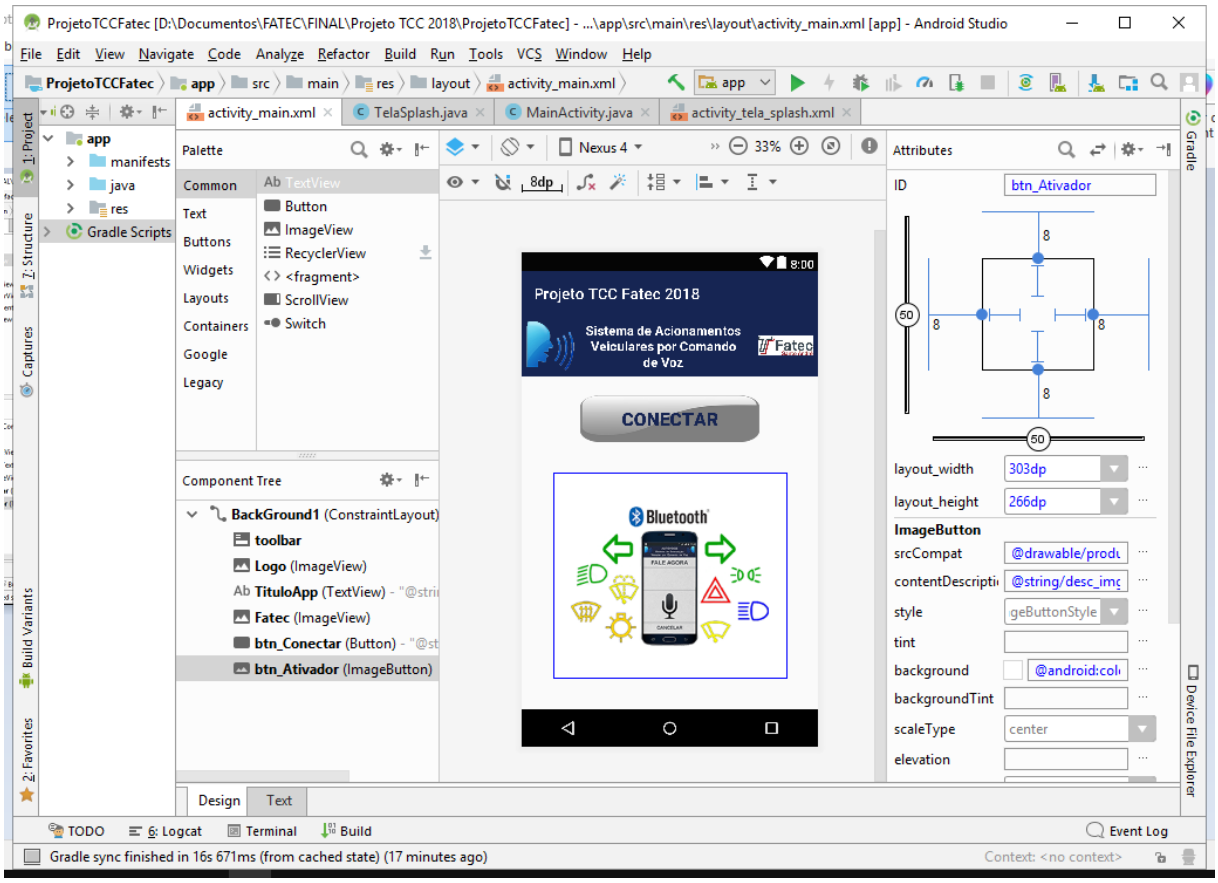
Fonte: (STATISTA, 2018)

3.3 Android Studio

O Android Studio é o Ambiente de Desenvolvimento Integrado oficial para o desenvolvimento de aplicativos Android. Além do editor de código e das ferramentas de desenvolvedor avançados, o Android Studio oferece ainda mais recursos para aumentar sua produtividade na criação de aplicativos Android (ANDROID DEVELOPER, 2018).

O Android Studio possui dois modos principais de trabalho, o Modo Design, onde é possível efetuar a montagem do layout do aplicativo a ser desenvolvido de forma gráfica, e o Modo Texto, onde se trabalha com linhas de programação. A página de desenvolvimento do layout da aplicação no Modo Design pode ser vista na *Figura 11* a seguir.

Figura 11: Layout IDE Android Studio



Fonte: Elaborado pelo autor

3.3.1 Android Emulator

O processo de produção de um aplicativo para dispositivos com o Sistema Android necessita de etapas de teste com certa frequência. Desta forma, trabalhar com simulações direto na tela do seu computador pode ser uma solução bem eficiente para esta finalidade.

Segundo Laureano (2006, p. 18), um emulador é o oposto da máquina real, isto é, ela se utiliza de todas as informações sobre o funcionamento de uma máquina real contidas na sua memória implementando em um ambiente abstrato todas as instruções realizadas por ela, possibilitando executar com perfeição um aplicativo de um tipo de plataforma em outra. A IDE Android Studio possui Interfaces de Programação de Aplicativos (APIs), que são emuladores que rodam versões do Android que o programador escolhe, rodando como se fosse um dispositivo Android real, possibilitando a simulação da aplicação diretamente na tela do computador. Através destas APIs é possível criar protótipos, desenvolver e testar aplicativos do

Android sem usar um dispositivo de hardware. Além de celulares e tablets Android, como o exemplo da *Figura 12*, o Android Emulator também é compatível com o Android Wear e com dispositivos Android TV para simulações de projetos (ANDROID STUDIO, 2018).

Figura 12: Android Emulator

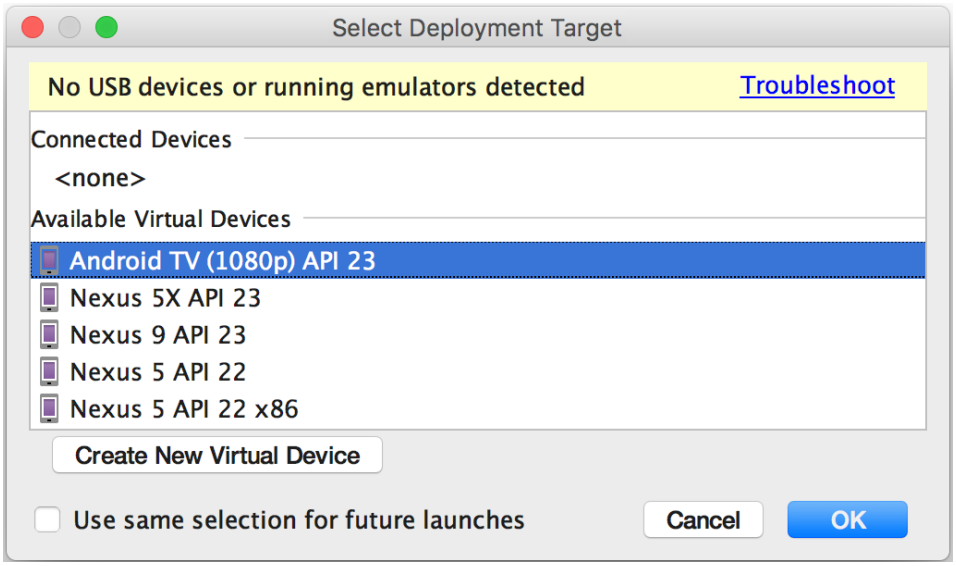
Fonte: (ANDROID STUDIO, 2018)

É possível redimensionar dinamicamente a janela do emulador conforme a necessidade, aumentar e reduzir o zoom, alterar a orientação e até registrar uma captura de tela. Quando o aplicativo em desenvolvimento é executado no emulador, ele pode:

- Usar os serviços da plataforma Android para chamar outros aplicativos;
- Acessar a rede de internet;
- Reproduzir mídias como áudio e vídeo;
- Aceitar entradas de áudio;
- Armazenar e recuperar dados;
- Especificar a velocidade e o status da rede;
- Simular um cartão SD e o armazenamento interno de dados, entre outras (ANDROID STUDIO, 2018).

Para a simulação de um aplicativo no Android Emulator clicando em RUN ▶, onde será apresentada a caixa de diálogo ilustrada na *Figura 13*.

Figura 13: Tela de Seleção do Emulador ou Depuração USB



Fonte: (ANDROID STUDIO, 2018)

Nesta tela de seleção você pode escolher entre os emuladores que foram anteriormente baixados e instalados na IDE e utilizados na criação de dispositivos virtuais que ficam disponíveis para utilização. A navegação no emulador é idêntica com a do dispositivo real, a diferença é que é necessária a utilização de mouse e teclado, salvo se o emulador estiver em execução em notebooks com tela sensível ao toque (ANDROID STUDIO, 2018).

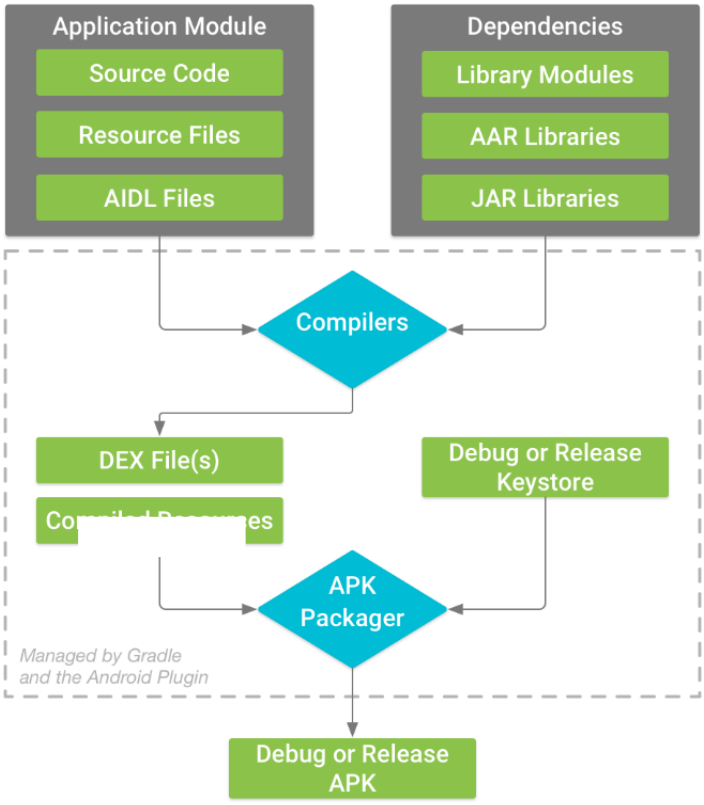
3.3.2 Sistema de Compilação

O sistema de compilação do Android Studio trabalha em função de um kit avançado de ferramentas chamado Gradle (2018). Ele tem a função de automatizar e gerenciar o processo de compilação, permitindo que a definição das configurações de compilação dos projetos de criação dos aplicativos sejam personalizadas e flexíveis. Existem diversos modos de configuração disponíveis de compilação com a intenção de definir o conjunto ideal de códigos e recursos, reutilizando as partes comuns a todas as versões do aplicativo.

O processo de compilação para geração de um aplicativo no Android Studio segue, como ilustrado na *Figura 14*, as seguintes etapas:

1. *Conversão do código-fonte em arquivos Dalvik Executable (DEX), que são arquivos de inicialização dos APKs para Android. Estes arquivos incluem o bytecode que é executado em dispositivos Android e todo o restante em recursos compilados;*
2. *O APK Packager faz a combinação entre os arquivos DEX e os recursos compilados em um só APK, preparando-o para assinatura do mesmo (necessário para a instalação no dispositivo);*
3. *O APK Packager assina o APK. Se a geração do mesmo for de depuração, são usadas chaves internas do Android Studio do repositório de chaves de depuração. Para o lançamento do APK na Play Store, é necessária a geração de chaves de lançamento pelo desenvolvedor;*
4. *Antes da geração do APK final, o Packager usa a ferramenta Zipalign* (2018)*, que é uma ferramenta que fornece otimização importante para os aplicativo Android garantindo que todos os dados não compactados iniciem com um alinhamento específico relativo ao início do arquivo, com o benefício reduzir a quantidade de RAM consumida ao executar o aplicativo.*

Figura 14: Processo de Compilação de um Módulo de Aplicativo Android



Fonte: (ANDROID DEVELOPERS, 2018)

No final do processo de compilação, o APK de depuração ou de lançamento gerado e instalado no dispositivo fica gravado nos arquivos do projeto no computador. Este APK pode ser usando para implantar, testar ou lançar o aplicativo ao público.

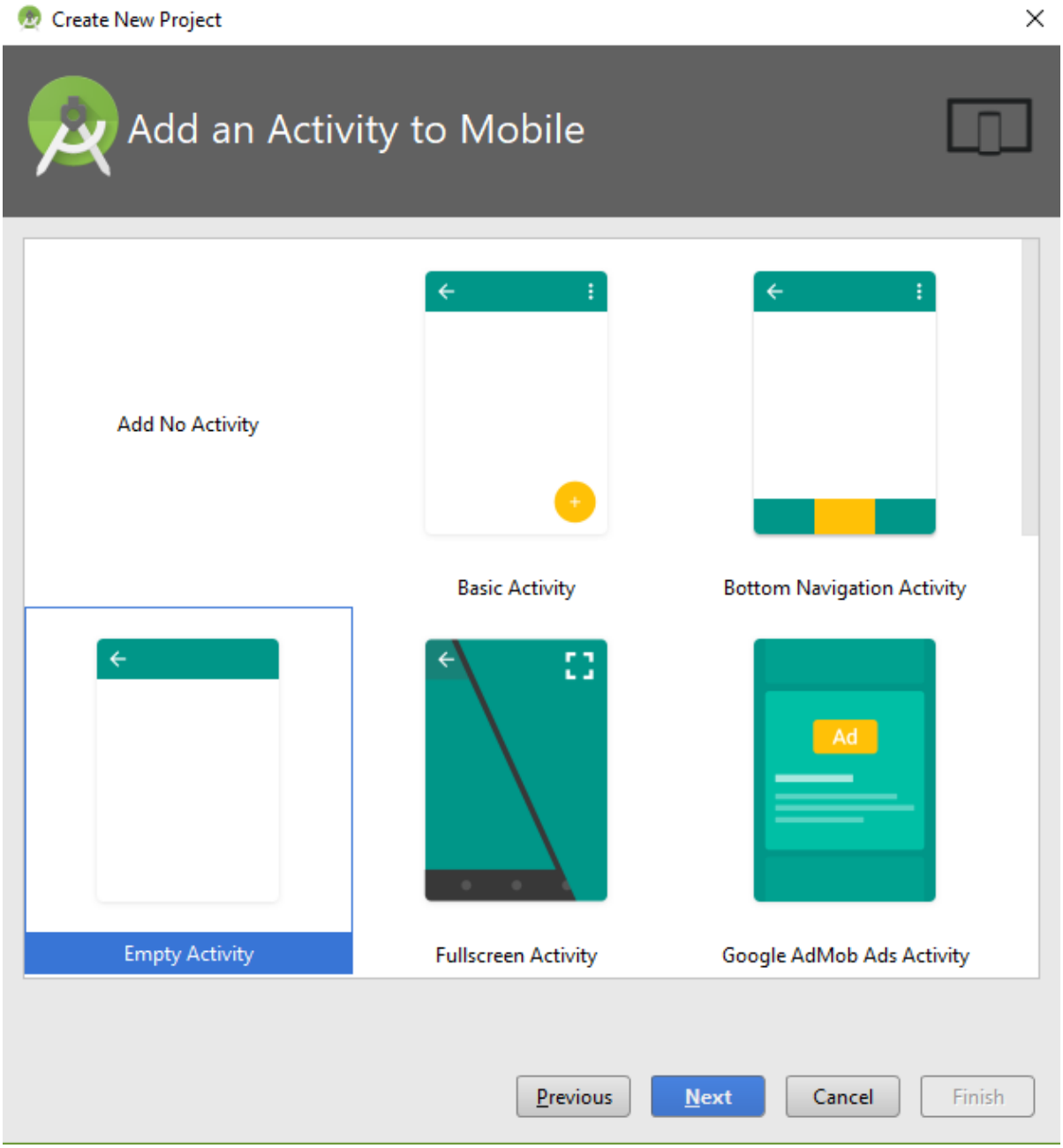
3.3.3 Arquivos de Configuração de Compilação

Para iniciar um novo projeto com base em padrões razoáveis que são utilizados nas mais diversas aplicações desenvolvidas hoje em dia, o Android Studio cria automaticamente alguns desses arquivos. São configurações iniciais para a aplicação, de compilação e configurações Java e um layout que é escolhido no momento da criação do projeto de alguns que a plataforma disponibiliza. Cada layout representa uma Activity que é gerada no projeto e cada uma possui suas características distintas e que podem ser utilizadas para diferentes tipos de aplicativos a serem desenvolvidos.

Algumas configurações personalizadas de compilação também são possíveis de serem utilizadas na IDE, porém necessitam maior conhecimento seguindo o nível de programação. Pode-se fazer alterações em um ou mais arquivos de configuração de compilação ou em arquivos *build.gradle*. São arquivos de texto sem formatação que usam a Domain Specific Language (DSL) para descrever e manipular a lógica de compilação que usa Groovy, que é uma linguagem dinâmica para a máquina virtual Java (JVM). A página oficial do Android Studio informa que não é necessário conhecimento em Groovy para começar a configurar a compilação, pois o Android Plugin for Gradle introduz a maioria dos elementos DSL de que a aplicação precisará.

A página Create New Project inicializa a criação da nova aplicação utilizando a configuração padrão do Android Studio, criando os arquivos necessários para o desenvolvimento inicial da aplicação. Uma das etapas do processo de criação é a escolha da nova Activity dentre algumas opções que vão desde o layout vazio, passando por alguns básicos com botão e até telas de login ou de utilização de mapas do Google. A *Figura 15* ilustra a etapa de seleção da Activity inicial do Projeto de Aplicação no Android Studio.

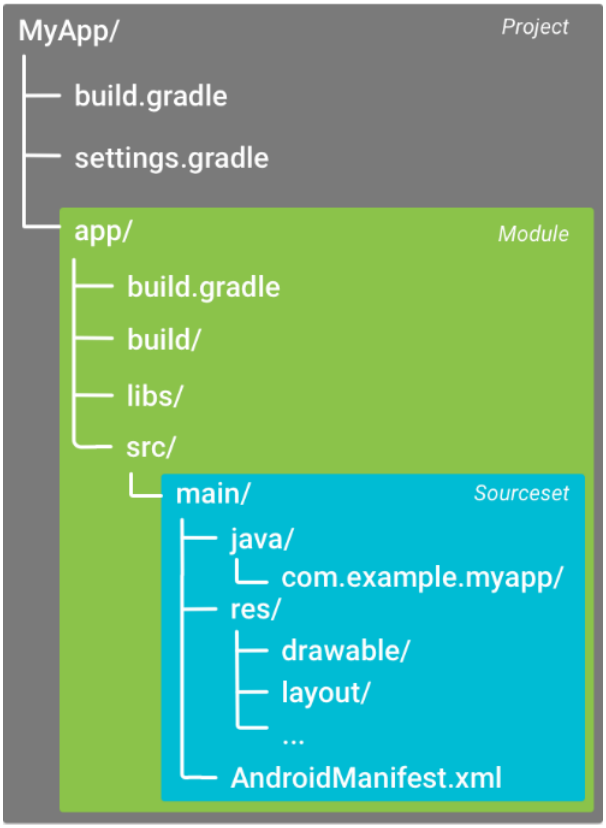
Figura 15: Criação de Novo Projeto - Escolha da Primeira Activity



Fonte: Elaborada Pelo Autor

Após criado o projeto, uma árvore de arquivos chamada Project é criada e passa a ser visualizada no canto direito da IDE Android Studio. Nele estão todos os arquivos do projeto. A *Figura 16* mostra a estrutura padrão para um módulo de aplicativo Android.

Figura 16: Estrutura de Projeto Padrão Módulo de Aplicativo Android



Fonte: (ANDROID DEVELOPERS, 2018)

O Android Studio ainda possibilita o desenvolvimento de aplicativos visando diversos fatores de formato com um único projeto para compartilhar facilmente código entre diferentes versões do aplicativo. O Android Studio oferece um ambiente unificado para o desenvolvimento de aplicativos para telefones e tablets Android e dispositivos Android Wear, Android TV e Android Auto como mostrado na *Figura 17*.

Figura 17: Dispositivos Suportados no Android Studio

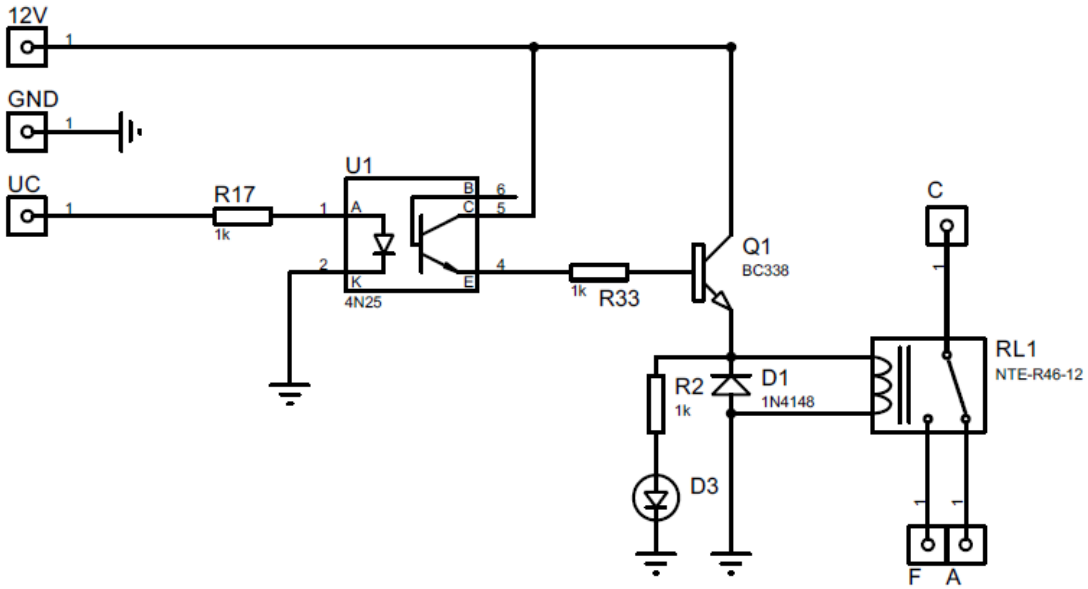
Fonte: Site Oficial Android Studio

3.4 Placa de Controle Veicular

Para que o sistema de controle possa funcionar no veículo, foram desenvolvidos circuitos de interface entre o sistema de controle e o sistema existente no veículo.

Inicialmente foram feitos levantamentos de quais eram as cargas e suas correntes de acionamento dos diversos sistemas de sinalização/ iluminação do veiculo. Neste passo, foi desenvolvido um drive para instrumentar os circuitos de acionamento existente no veículo e coleta de dados das correntes para cada circuito. O drive baseia-se em um sistema com optoacoplador 4N25 (VISHAY SEMICONDUCTORS, 2010) utilizado para garantir isolação galvânica do circuito que, ao receber o sinal de acionamento vindo do microcontrolador, realiza a polarização de um transistor BC338 (FAIRCHILD, 2015) que chaveia um relé 12V (HONGFA, 2018) como mostrado na *Figura 18*.

Figura 18: Drive para Testes de Corrente

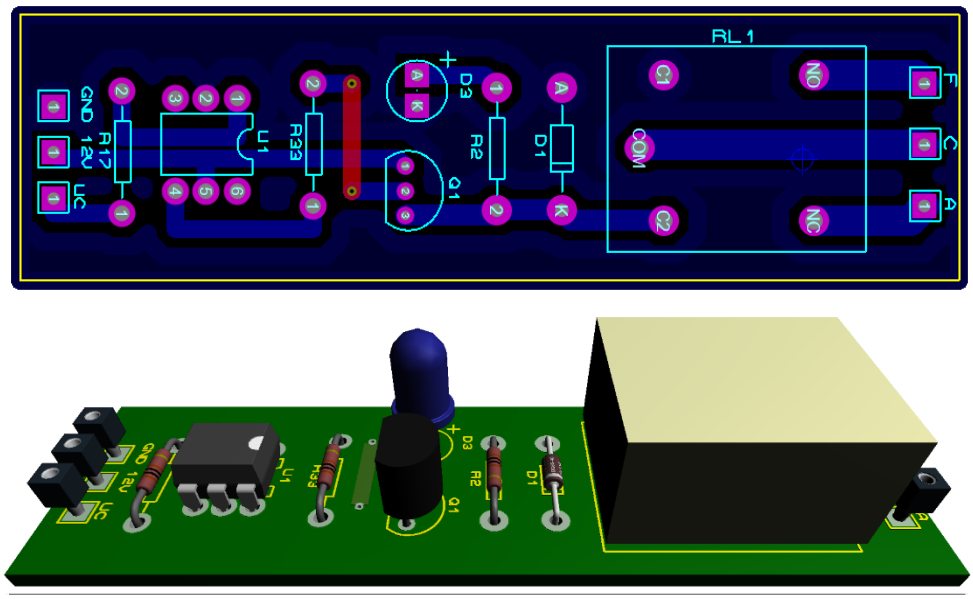


Fonte: Elaborada pelo Autor

A alimentação para o funcionamento básico de chaveamento do drive vem da saída de 12V da CPU FATEC, sendo que a corrente a ser utilizada para o acionamento passe direto da bateria passando pelo relé e indo direto ao ponto de conexão no veículo.

Este drive foi construído em uma placa de fenolite simples com o layout desenvolvido no software Ares Proteus. O esquema elétrico, o layout da placa e a demonstração em 3D estão ilustrado na *Figura 19*.

Figura 19: Layout e Representação 3D Drive Para Testes de Corrente



Fonte: Elaborada pelo Autor

Na *Tabela 1* estão os valores de corrente obtidos para cada um dos circuitos do veículo.

Tabela 1: Relação de Acionamentos e Correntes

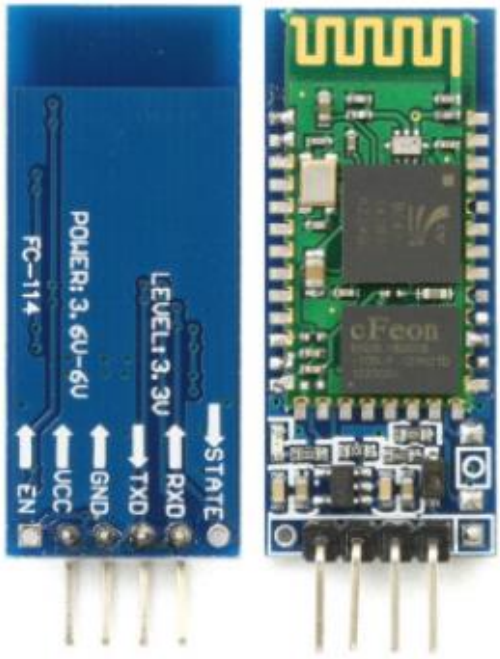
Pino PIC	Pino Módulo	Descrição	Corrente (mA)
RB0	1	SETA_DIREITA	230
RB1	2	SETA_ESQUERDA	230
RB2	3	LANTERNA	220
RB3	4	FAROL_BAIXO	540
RB4	5	FAROL_ALTO	560
RB5	6	FAROL_NEBLINA	430
RB6	7	LAV_DIANTEIRO	600
RB7	8	LAV_TRASEIRO	600
RA0	9	LIMP_TRASEIRO	650
RA1	10	LIMP_DIANTEIRO	650
RA2	11	INTERMITENTE	650
RA3	12	DESEMBACADOR	480
RA4	13	EMERGENCIA	300
RA5	14	LIVRE	0

Fonte: Elaborada pelo Autor

Após a fase de medição de todos os circuitos do veículo, 14 no total. Os circuitos com relè foram substituídos por dois circuitos integrados ULN2003 (DIODES INCORPORATED, 2017) para chaveamento dos relés. O ULN2003 possui 7 pares Darlington capazes de controlar cargas indutivas com correntes de até 500mA.

Para o sistema de comunicação sem fios Bluetooth, foi empregado o módulo Bluetooth HC-06 do fabricante Guangzhou HC Information Technology, *Figura 20*.

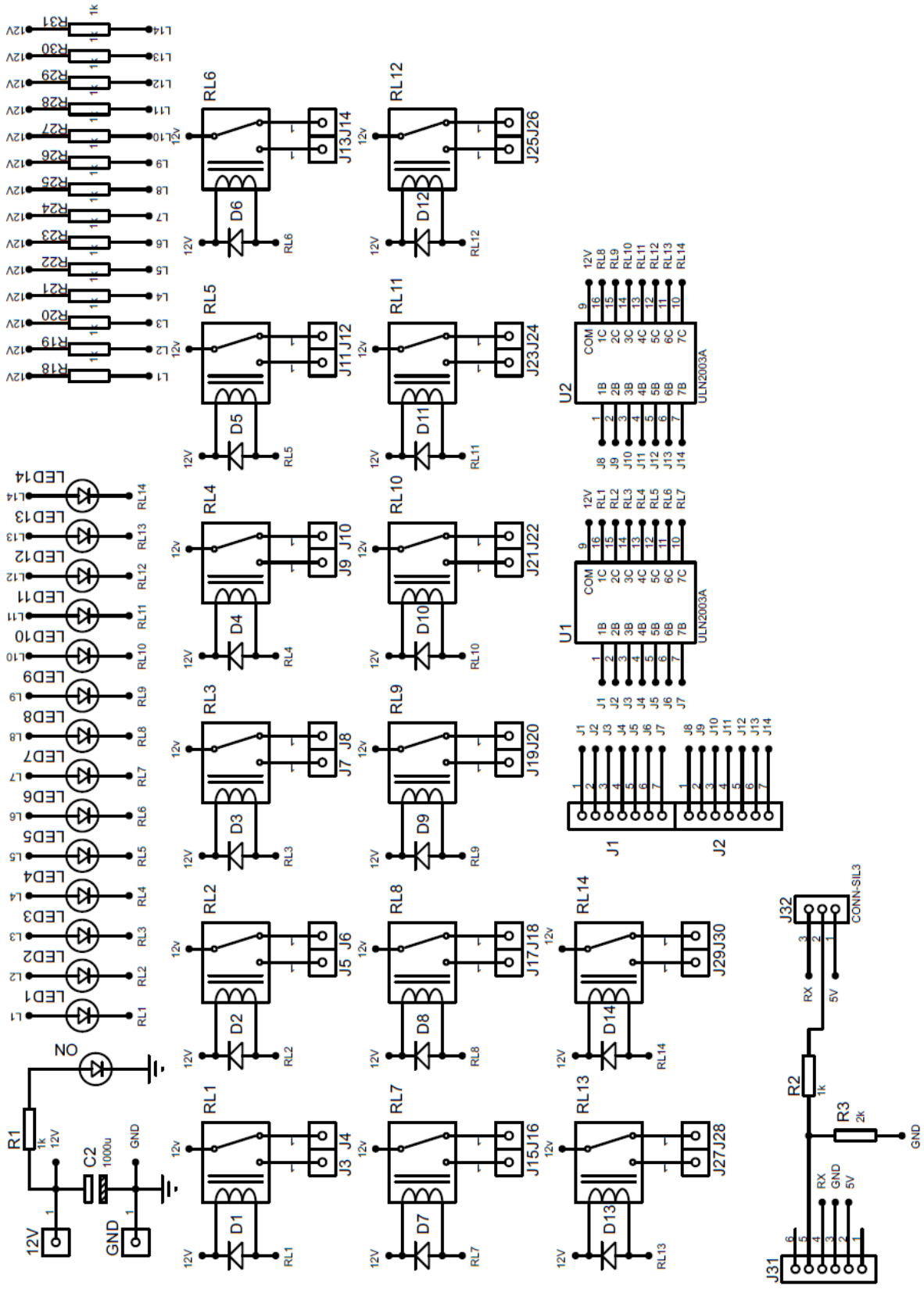
Figura 20: Módulo Bluetooth HC-06



Fonte: Elaborado pelo Autor

O módulo Bluetooth HC-06 está conectado ao conector J31, Figura 21 e a sua alimentação é feita através da saída de 5V vinda da CPU FATEC.

Figura 21: Esquemático Módulo de Interface

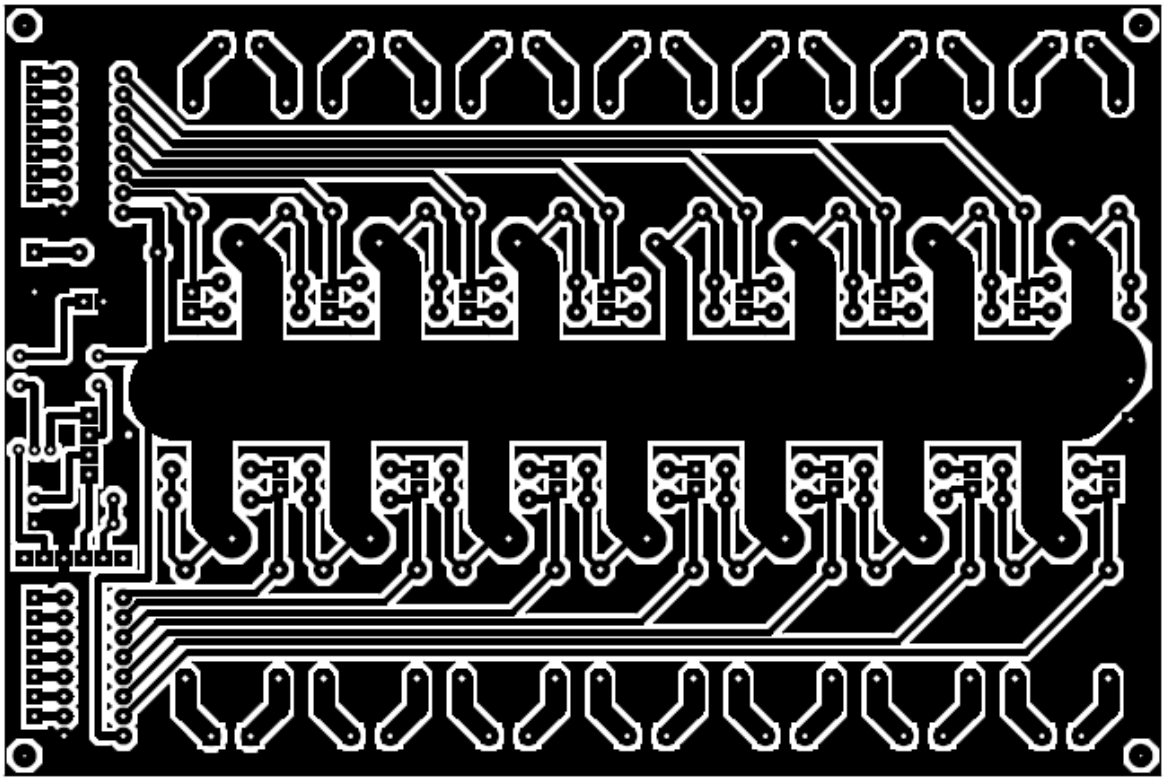


Fonte: Elaborado pelo Autor

3.4.1 Montagem do Hardware

A montagem do módulo desenvolvido para o sistema foi executada em uma placa de fenolite de face simples. O módulo possui dois barramentos com sete entradas cada onde são conectados os pinos de IO do microcontrolador. Estes barramentos levam o sinal para dois circuitos integrados ULN2003 que são amplamente utilizados em drivers de motores de passo que chaveiam os 14 relés 12V contidos na placa. O layout para impressão também pode ser visto na *Figura 22*.

Figura 22: Layout para Impressão e Transferência Térmica

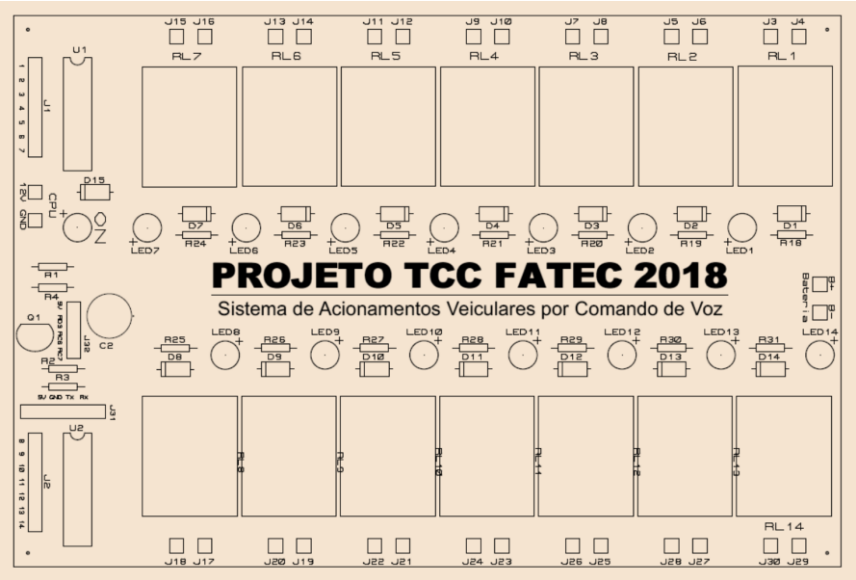


Fonte: Elaborada pelo Autor

Após a finalização do layout do módulo de interface, foram utilizadas folhas de papel Transfer próprias para impressora laser para a transferência para a placa por transferência térmica. Este tipo de folha, amplamente utilizadas em processos de sublimação, é um papel especial que possui uma camada de resina de poliéster na sua camada principal onde é impressa a arte que será transferida. Esta transferência normalmente é feita com prensas que trabalham a temperaturas de aproximadamente

200°C. A *Figura 23* a seguir também mostra a máscara de componentes impressa na parte superior não condutiva da placa.

Figura 23: Máscara de Componentes



Fonte: Elaborada pelo Autor

Já na *Figura 24*, temos a placa após ter passado pelo processo de corrosão no Percloreto de Ferro, pronta para furação e solda dos componentes.

Figura 24: Placa Após Processo de Corrosão

Fonte: Elaborada pelo Autor

Como o módulo foi desenvolvido para funcionar a partir de relés 12V chaveados por CIs ULN2003 que são amplamente utilizados para a produção de drivers de motores de passo. Nele também existem Leds que demonstram qual relé está acionado no momento. A relação de componentes utilizados para o módulo está listada na *Tabela 2*.

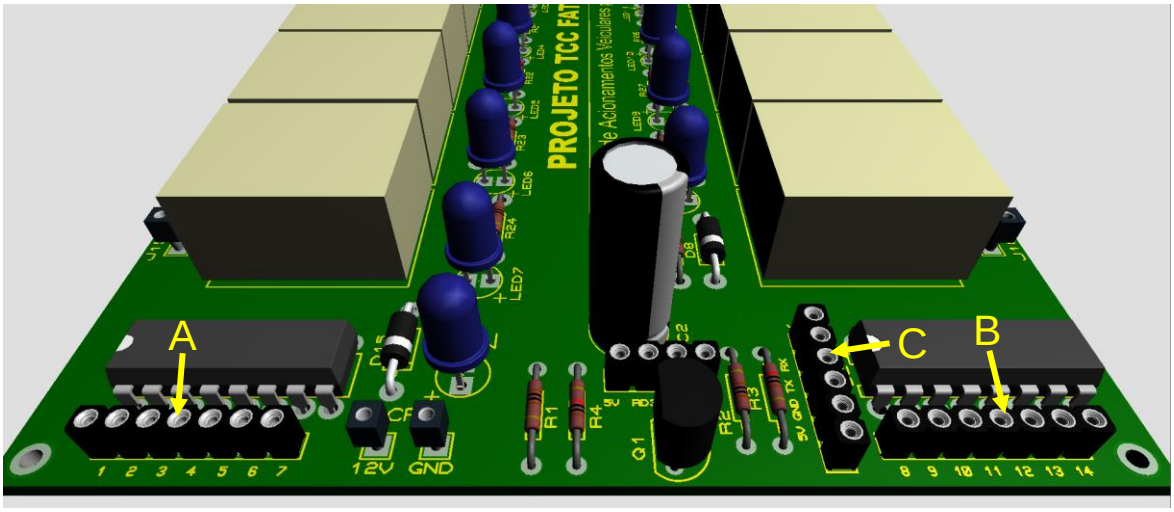
Tabela 2: Lista de Componentes Módulo Interface

Qtde	Componente	Características
14	Relé	12V
14	Diodo	1n4001
15	Led	5mm
16	Resistor	1k
1	Resistor	2k
2	Circuito Integrado	ULN2003
2	Soquete Estampado	16 pinos
1	Barra de Pinos	Fêmea
1	Capacitor Eletrolítico	1000uF 16V
16	Borne KRE	2 Vias
1	Placa de Fenolite	10x15mm

Fonte: Elaborado pelo Autor

A *Figura 25* ilustra os barramentos de comunicação do módulo. Foram utilizadas barras de pinos fêmea onde são conectados os jumpers ligados à CPU Fatec (itens A e B) e outro para instalação do módulo Bluetooth HC-06.

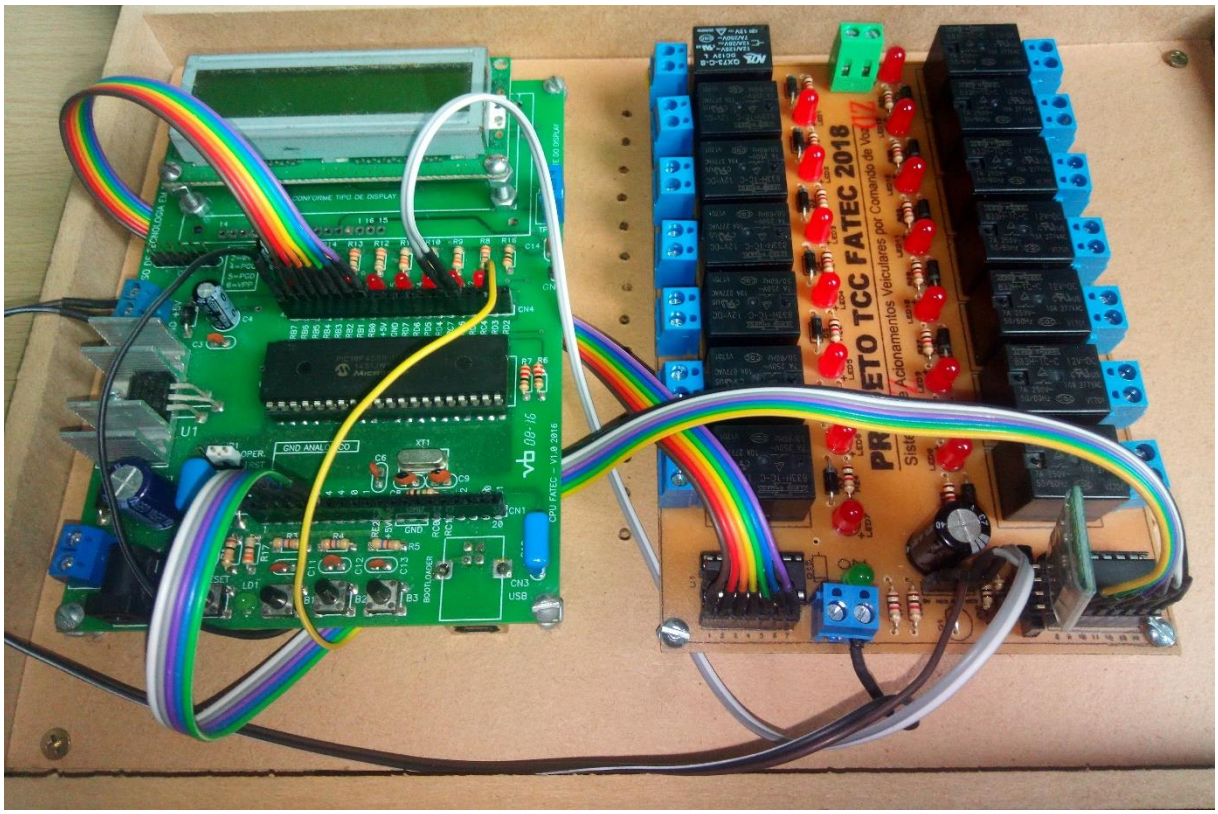
Figura 25: Barramentos de Comunicação



Fonte: Elaborada pelo Autor

As conexões são feitas com cabos flat entre a CPU Fatec e o módulo, com a montagem final ilustrada na *Figura 26*. A alimentação para a CPU Fatec é feita da mesma forma que é usada normalmente, através de uma fonte 12V. A CPU possui uma saída de 5V, onde está sendo alimentado o Módulo Bluetooth HC-05. Já o módulo de interface foi desenvolvido para ser alimentado através da bateria do veículo.

Figura 26: Montagem Final



Fonte: Elaborado pelo Autor

Como a corrente máxima suportada pelo ULN2003 é de 500mA, foram feitos testes com todos os relés acionados para verificação da corrente máxima utilizada nesta situação, encontrando valores de aproximadamente $400mA$ para a placa inteira, isto é, menos da metade da corrente máxima para cada CI.

3.5 Módulo Bluetooth HC-06

A forma de comunicação a ser utilizada é uma que já vem nativa em todos os smartphones da atualidade, o Bluetooth. Esta tecnologia permite a transferência de informações entre dois dispositivos sem a necessidade de conexões fixas por cabos e fios, por exemplo. Esta tecnologia permite a conexão e troca de informações entre dispositivos como telefones celulares, notebooks, computadores, impressoras, câmeras digitais e consoles de videogames digitais através de uma frequência de rádio de curto alcance globalmente licenciada e segura (ROBOCORE TECNOLOGIA, 2016).

De acordo com Siqueira (2006, p. 2), o Bluetooth consiste basicamente de um transceiver (hardware) e uma pilha de protocolos (software) que possibilitam a conexão de dispositivos e a troca dados entre eles, formando durante a sua operação um canal de rádio sincronizados a partir de um clock comum e a um padrão de saltos de frequência que servem para combater interferência e enfraquecimento do sinal.

Existem três protocolos de comunicação disponíveis para o Bluetooth. Cada protocolo depende principalmente de dois fatores que devem ser a princípio previamente estudados com o intuito de determinar o melhor hardware a ser utilizado, que são a Potência Máxima Permitida e o Alcance da Comunicação entre os dispositivos da rede.

Tabela 3: Potência Máxima Permitida e Alcance de Cada Classe Bluetooth

Classe	Potência Máxima Permitida	Alcance (Aproximadamente)
1	100mW (20dBm)	Até 100 Metros
2	2,5mW (4dBm)	Até 10 Metros
3	1mW (0dBm)	~1 Metro

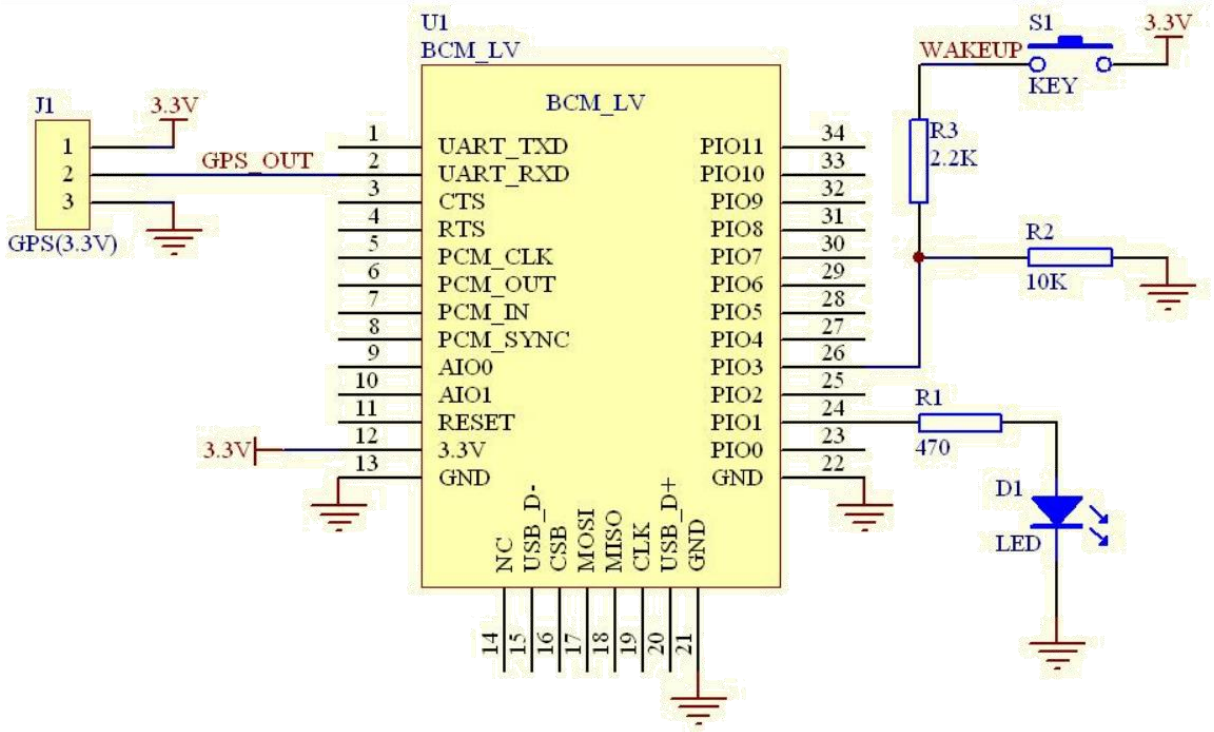
Fonte: (INFO WESTER, 2018)

O módulo HC-06 é um módulo Bluetooth Serial Port Protocol (SPP) extremamente fácil de ser utilizado com 4 terminais. Ele foi projetado para a tornar a configuração de conexão serial sem fio entre dois dispositivos da forma mais simplificada possível com relação à configuração do sistema a ser implementado.

De acordo com o Datasheet do dispositivo, ele é utilizado a partir da saída serial de um microcontrolador qualquer, sendo que o seu *baud rate* pode ser configurado de acordo com a frequência a ser utilizada no sistema. Ele usa uma tecnologia chamada

CSR Bluecore 04 - Sistema Bluetooth de chip único externo com tecnologia CMOS e com AFH (Função de frequência adaptativa). Estas duas tecnologias disponíveis no módulo permitem que haja uma simplificação no processo de desenvolvimento dos projetos. O diagrama de blocos do módulo Bluetooth HC-06 pode ser visto a seguir na *Figura 27*.

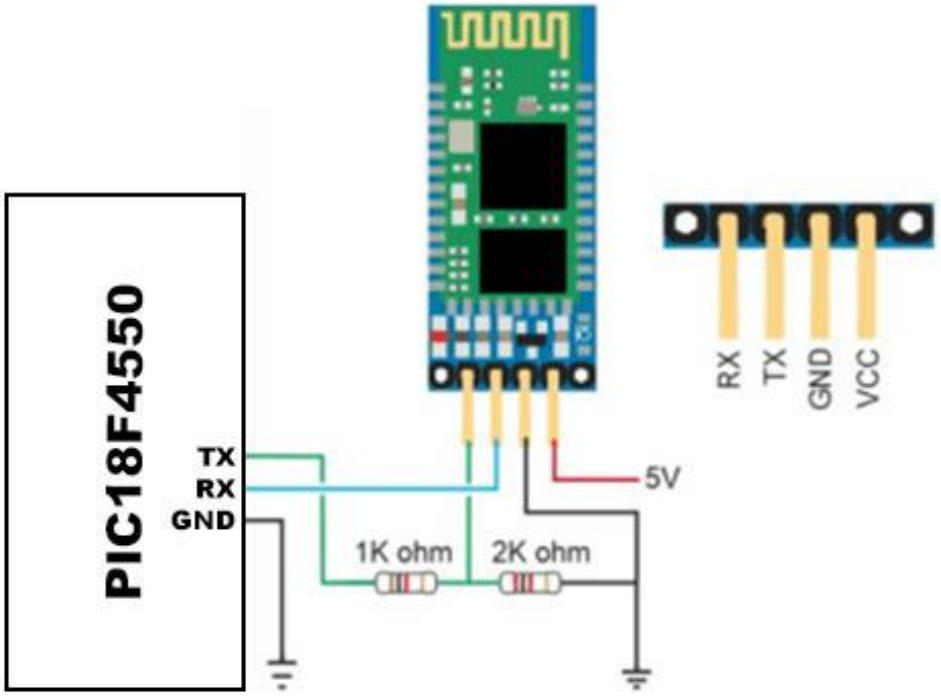
Figura 27: Diagrama de Blocos Módulo HC-06



Fonte: Guangzhou HC Information Technology

A tensão de entrada no pino RX do módulo bluetooth aceita uma tensão máxima de 3,3V nos pinos de transmissão e de recepção de dados. Como a tensão de saída dos pinos TX do microcontrolador PIC18F4550 (MICROCHIP TECHNOLOGY, 2006) é de 5V, torna-se necessária a redução da tensão utilizada com a intenção de manter a integridade elétrica do componente. Para este caso, foi utilizado um divisor de tensão de acordo com a *Figura 28* para manter a tensão máxima de entrada em 3,3V.

Figura 28: Divisor de Tensão Entrada RX do HC-06



Fonte: Elaborado pelo Autor

3.7 Arquitetura de Software Microcontrolador PIC18F4550

A arquitetura de software empregada foi um escalonador de funções desenvolvido para Microcontroladores PIC de 8 bits. A utilização desta arquitetura foi escolhida com a intenção de possibilitar a implementação do código sem a utilização de delay para as rotinas que precisariam de temporização, utilizando apenas a base de tempo programada de 1ms, proporcionando tempos mais precisos para a execução de cada uma das 4 tarefas listadas a seguir:

- **Tarefa Task1:** Utilizado uma base de tempo de 10ms, a tarefa Task1 foi utilizada para monitoração do botão de acionamento do comando de voz no pino digital RE0 programado como entrada. Para a leitura do botão, foi implementado um debounce de 50ms para impedir acionamentos indevidos do comando de voz ou duplo acionamentos.
- **Tarefa Task2:** Esta tarefa foi utilizada para controle de tempo de acionamento dos lavadores do para-brisa. Com a utilização de uma base de tempo de 500ms, a tarefa controla por uma máquina de estados o momento de inicialização da bomba do

esguicho do lavador dianteiro por um tempo determinado e após este tempo é acionado o limpador correspondente.

- **Tarefa Task3:** Idêntica à tarefa anterior, a tarefa Task3 faz exatamente a mesma função para o lavador traseiro.
- **Tarefa Task4:** A tarefa Task4 fica responsável por controlar os comandos de indicação de direção e emergência do veículo para sistemas que necessitam que o sinal já seja pulsado. Utilizando um período de 400ms conseguimos controlar a intermitência das lâmpadas de seta. Para veículos que utilizam relés para a seta e para a emergência, o sinal a ser enviado é um simples nível alto.

Além das funções controladas pelas tarefas do escalonador, temos a comunicação e tratamento das mensagens trocadas com o módulo bluetooth. Esta comunicação é feita através de comunicação serial USART com mensagens de 1 byte (1 caractere).

Pensando em evitar qualquer tipo de acionamento involuntário por ruído que porventura possam surgir no sistema, utilizamos um debounce de 50ms. O envio de solicitação para inicialização do comando de voz é feito através do caractere ‘@’ no momento em que o botão RE0 é pressionado. Este caractere é enviado para o módulo bluetooth que o transmite para o aplicativo.

Para os demais acionamentos que estão listados na Tabela 4, a inicialização do comando é feita através do aplicativo instalado no smartphone. O microcontrolador, para este processo, fica responsável por receber o caractere correspondente ao comando enviado pelo aplicativo e faz o tratamento do mesmo para o chaveamento no módulo para o veículo.

Tabela 4: Comandos Reconhecidos e Envios Bluetooth

Comando de Voz	Hexadecimal	ASCII
Liga Seta Direita	0x41	A
Desliga Seta Direita	0x61	a
Liga Seta Esquerda	0x42	B
Desliga Seta Esquerda	0x62	b
Liga Lanterna	0x43	C
Desliga Lanterna	0x63	c
Liga Farol Baixo	0x44	D
Desliga Farol Baixo	0x64	d
Liga Farol Alto	0x45	E
Desliga Farol Alto	0x65	e
Liga Farol de Neblina	0x46	F
Desliga Farol de Neblina	0x66	f
Liga Lavador Dianteiro	0x47	G
Desliga Lavador Dianteiro	0x67	g
Liga Lavador Traseiro	0x48	H
Desliga Lavador Traseiro	0x68	h
Liga Limpador Traseiro	0x49	I
Desliga Limpador Traseiro	0x69	i
Liga Limpador Dianteiro	0x50	J
Desliga Limpador Dianteiro	0x70	j
Liga Intermitente	0x51	K
Desliga Intermitente	0x71	k
Liga Desembaçador	0x52	L
Desliga Desembaçador	0x72	l
Liga Emergência	0x53	M
Desliga Emergência	0x73	m

Fonte: Elaborada pelo Autor

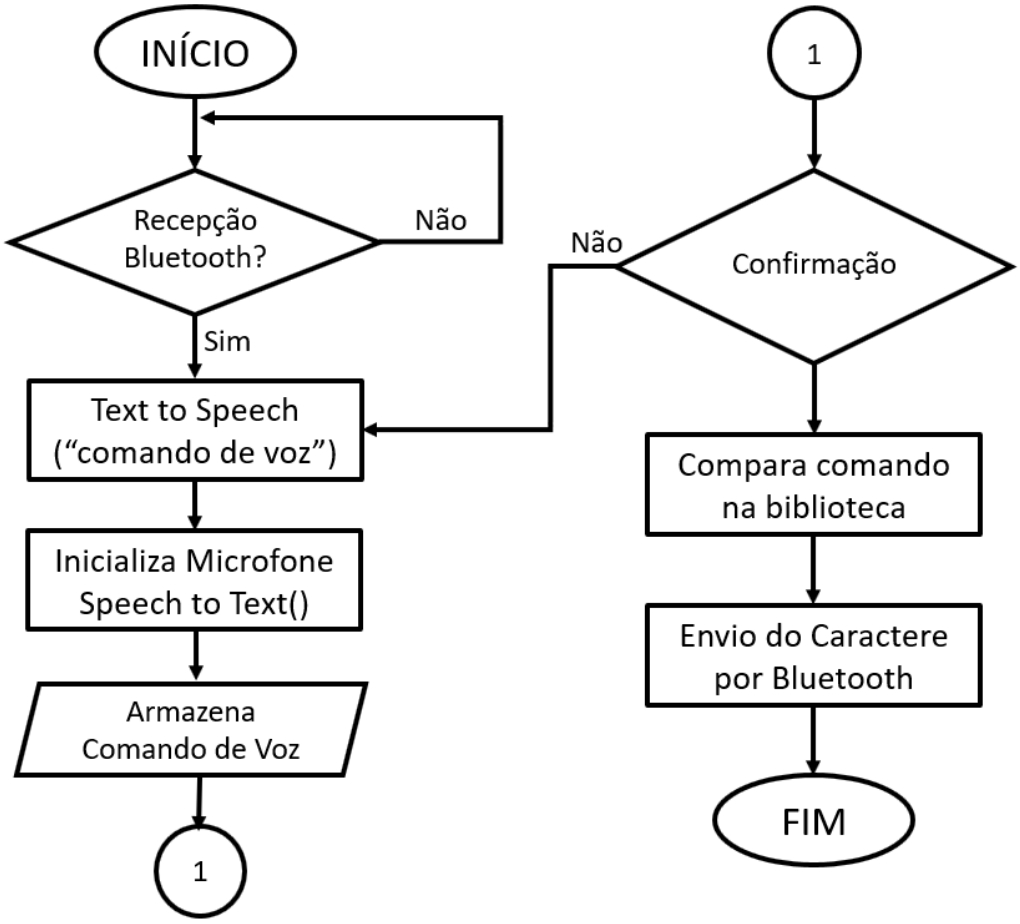
3.8 Aplicativo para Acionamentos por Comando de Voz

Buscando o desenvolvimento de um aplicativo mais leve e totalmente funcional, optou-se pela utilização do Android Studio por ser a IDE padrão do Android, além de proporcionar uma programação mais limpa devido à utilização de funções específicas do Java, sua linguagem de programação principal.

O aplicativo foi desenvolvido com a intenção de fazer a comunicação bluetooth com o módulo HC-06 da forma mais limpa possível. Para isto foram utilizadas as funções utilizadas como padrão para desenvolvimento deste tipo de aplicação de acordo com instruções disponíveis no site da IDE.

A maior parte do tempo o aplicativo fica em modo de espera, isto é, com a porta de recepção bluetooth aberta aguardando a recepção de dados pelo bluetooth, que normalmente é o dado correspondente à inicialização do comando de voz. Uma vez o dado recebido, inicializa-se o processo de reconhecimento de voz passando pela etapa de confirmação e finalizando com o envio do caractere correspondente ao comando para o microcontrolador. O processo de recepção e processamento dos dados pelo aplicativo está ilustrado na *Figura 29*.

Figura 29: Diagrama de Funcionamento do Aplicativo



Fonte: Elaborado pelo Autor

3.9 Inicialização da Aplicação Para Smartphone Android

Buscando o desenvolvimento de um aplicativo mais leve e totalmente funcional, optou-se pela utilização do Android Studio por ser a IDE padrão do Android, além de proporcionar uma programação mais limpa devido à utilização de funções específicas do Java, sua linguagem de programação principal.

Uma vez instalado o aplicativo no smartphone, o ícone configurado para apresentação no aparelho já pode ser clicado para inicialização do aplicativo. A *Figura 30* ilustra o ícone do aplicativo utilizado.

Figura 30: Ícone do Aplicativo

Fonte: Elaborado pelo Autor

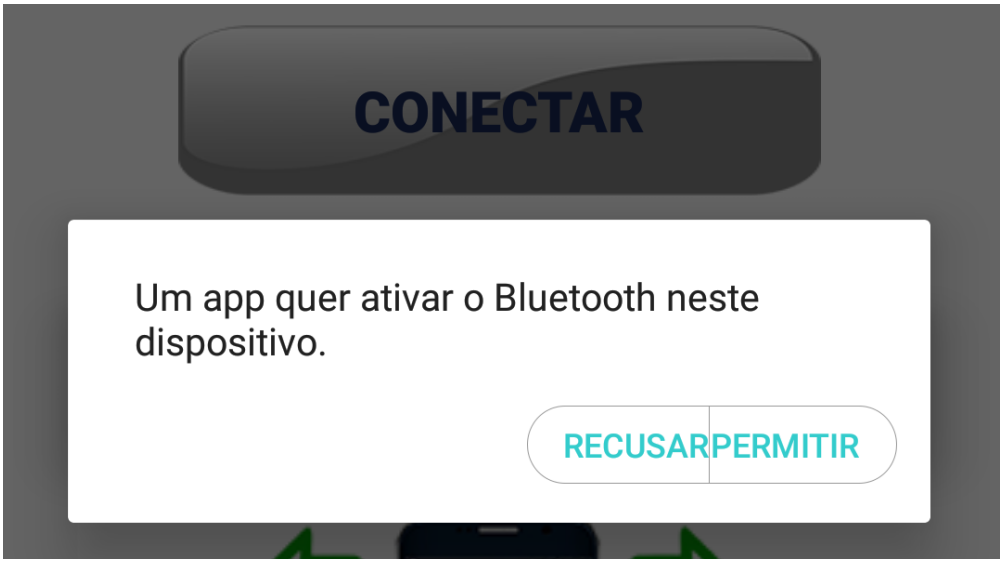
O aplicativo contém, apenas, duas páginas visíveis. A primeira página é uma tela de Splash que possui o logo e uma animação que permanece em execução enquanto o sistema se inicializa. A tela de Splash está representada na *Figura 31*.

Figura 31: Tela de Splash

Fonte: Elaborada pelo Autor

Inicializado o sistema, a aplicação verifica se o Bluetooth do aparelho se encontra ativado. Se não estiver ele solicita a ativação para o usuário e lhe apresenta dois botões de ação que, se negada a ativação o aplicativo é encerrado. Caso a solicitação seja aceita, o aplicativo ativa remotamente o Bluetooth do aparelho e inicializa a tela de conexão do aplicativo. A *Figura 32* ilustra a solicitação para a ativação da função Bluetooth no smartphone pelo aplicativo.

Figura 32: Solicitação de Ativação do Bluetooth



Fonte: Elaborada pelo Autor

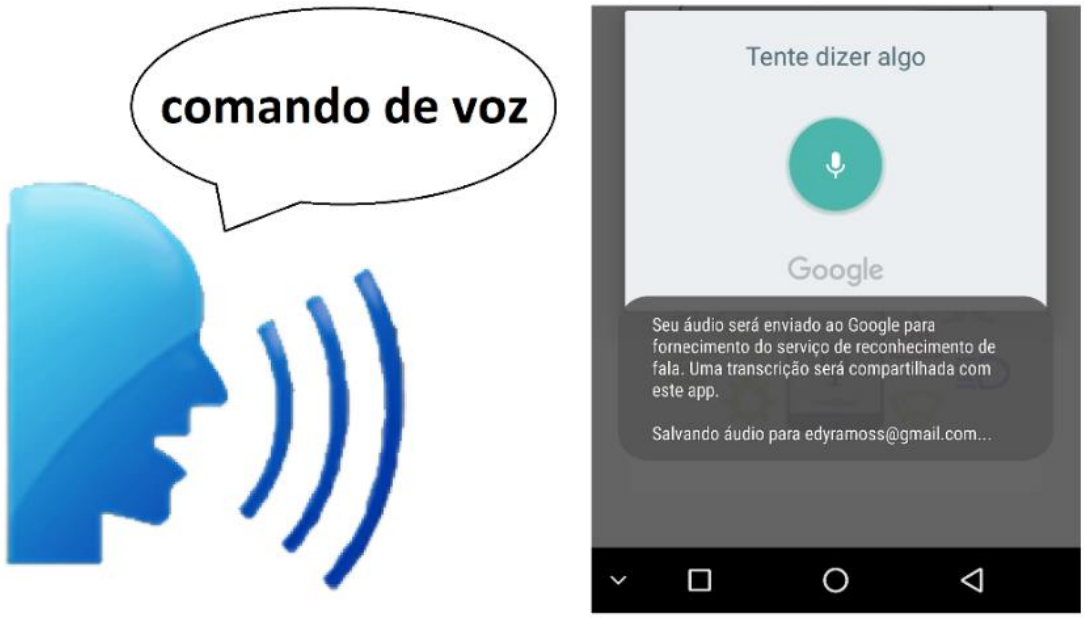
O módulo Bluetooth HC-06 possui somente a função de se conectar a uma rede disponível em modo Slave, isto é, não é capaz de gerar a conexão Bluetooth, apenas se conecta a um dispositivo que possui esta capacidade. Desta forma, a conexão necessariamente precisa ser feita, neste caso, pelo aplicativo no smartphone. O aplicativo desenvolvido utiliza em sua página principal um botão que, ao ser utilizado, mostra na tela os dispositivos pareados com o smartphone. O módulo Bluetooth aparece sob o nome do seu modelo e, ao ser selecionado, estabelece a conexão entre os dois dispositivos.

O microcontrolador é o responsável por verificar o status do botão que o usuário do veículo tem à disposição para inicialização do comando de voz. Ao ser pressionado, o microcontrolador envia pela porta serial o caractere correspondente a este processo ao módulo HC-06 que o transmite para o aplicativo. Uma vez recebido via Bluetooth o caractere “@”, o aplicativo inicializa o processo de reconhecimento inteligente de voz do Google. O microfone do aparelho fica ativo por 5 segundos

aguardando o comando de voz e desativa automaticamente caso não receba comando algum. Se o usuário dá início ao seu comando, o processo permanece ativo enquanto o mesmo estiver falando, encerrando o processo com o fim da frase.

O microfone do Android é ativado para que o usuário possa dar o comando desejado. Neste momento o comando é convertido para texto e é armazenado em uma variável do tipo string que será utilizada para a confirmação do comando pelo aplicativo para o usuário como mostrado na *Figura 33*.

Figura 33: Ativação do Comando de Voz



Fonte: Elaborada pelo Autor

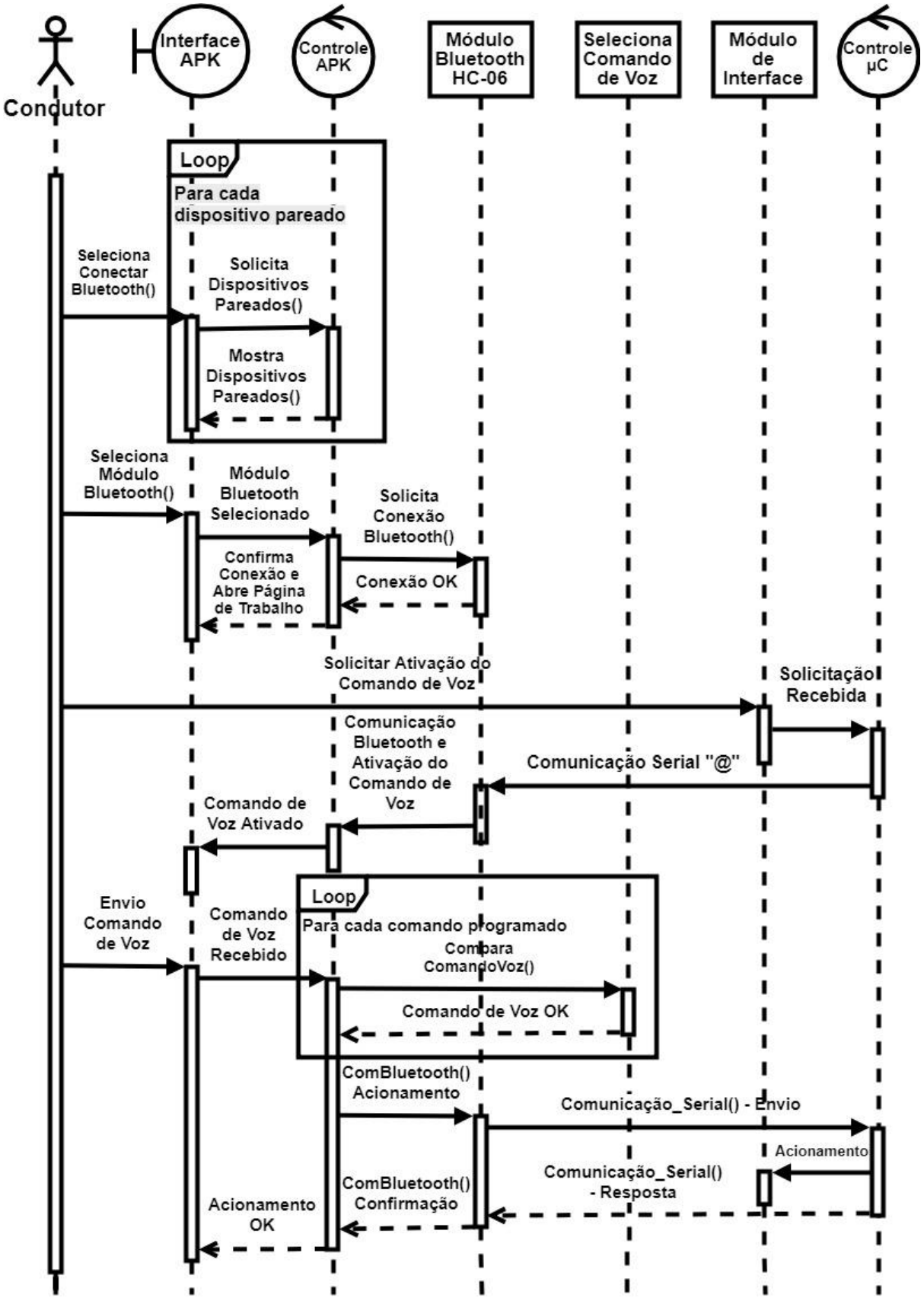
A frase recebida é tratada pela função de conversão de voz em texto (*speech-to-text*), função esta que também disponibiliza a frase convertida ao final do processo e a disponibiliza para tratamento da aplicação que a solicitou. O aplicativo armazena este texto em uma variável para ser posteriormente comparada com uma relação de comandos listados na sua programação para verificação do comando a ser enviado para o microcontrolador para que a devida função seja executada pelo mesmo. Também é possível a verificação na tela do dispositivo a frase convertida referente ao comando dado como mostrado na *Figura 34*.

Figura 34: Função Speech-to-Text - Conversão de voz em texto

Fonte: Elaborada pelo Autor

A sequência de funcionamento do sistema parte da leitura do botão de acionamento do comando de voz para inicialização, armazenamento, confirmação do comando e acionamento da função no veículo está ilustrado no Diagrama de Sequência da *Figura 35* a seguir.

Figura 35: Diagrama de Sequência

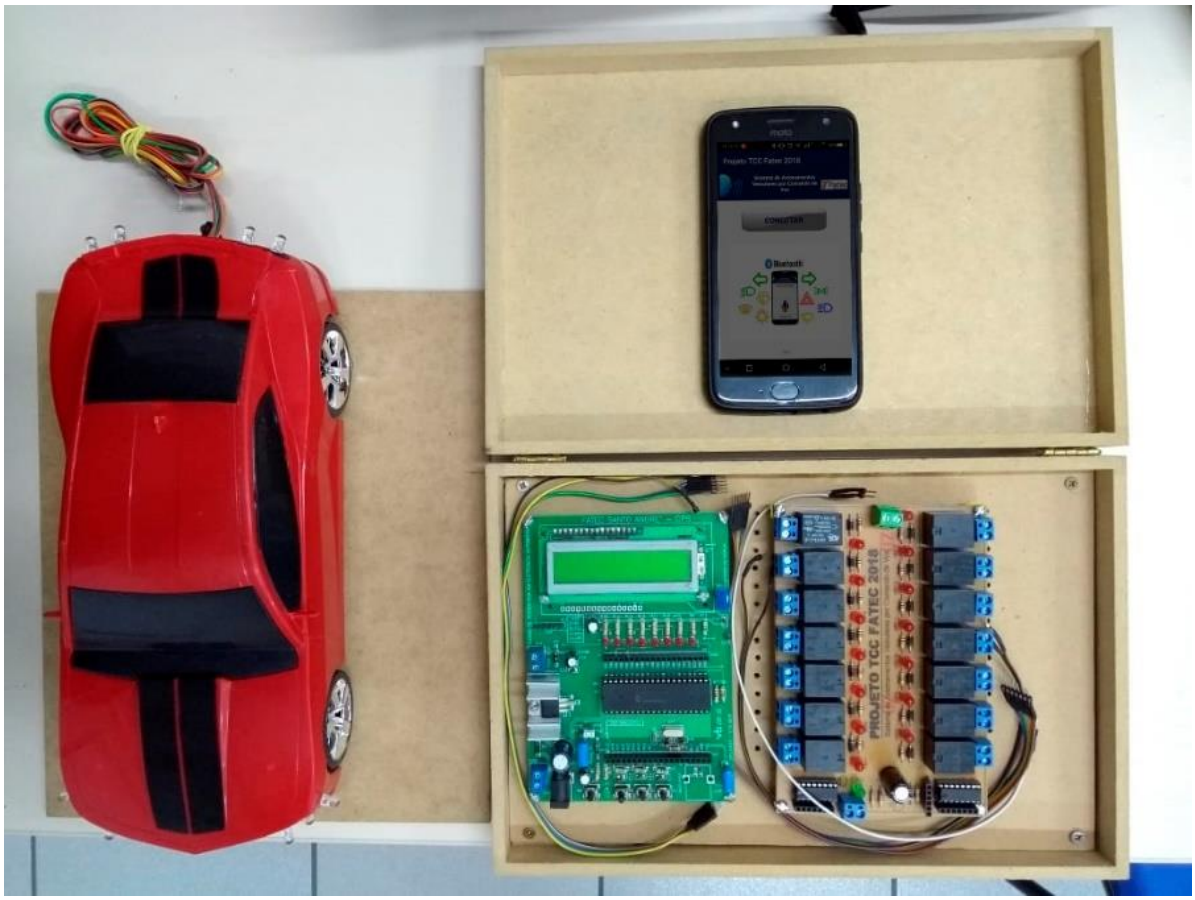


Fonte: Elaborado pelo Autor

3.10 Protótipo

Para testes e demonstração do funcionamento do sistema foi desenvolvido um pequeno protótipo onde seria possível a demonstração de alguns comandos de iluminação. Este protótipo é composto por um carrinho de brinquedo onde foram adicionados leds que se iluminariam de acordo com o comando dado e a placa de controle veicular como mostrado na *Figura 36.*

Figura 36: Protótipo para Simulação do Sistema



Fonte: Elaborado pelo Autor

4 RESULTADOS

Com a intenção de desenvolvermos um sistema capaz de fazer os acionamentos de um painel de instrumentos de um veículo através de comandos de voz, dois sistemas que trabalhassem em conjunto neccessáriamente precisariam ser desenvolvidos. Para isto, precisaríamos que o sistema fosse estável, que pudesse ser nutilizado por todas as pessoas, independente do seu timbre de voz ou sotaques, leve e que funcionasse utilizando o mínimo de dados móveis do smartphone.

Dois testes foram feitos diretamente na tela do smartphone. Para verificação do tamanho do aplicativo instalado no aparelho e a quantidade de dados móveis consumidos pela aplicação, as configurações de aplicativos do Android nos formneceu estas duas informações como mostrado na *Figura 37.* Ao serem recebidos os comandos pelo aplicativo, a nova API de reconhecimento de voz do Android permite o reconhecimento de voz de forma rápida e sem a necessidade de utilização de conexão à internet, facilitando a sua utilização em locais em que o sinal de dados móveis utilizados em telefonia celular é fraco ou inexistente. Da mesma forma, o armazenamento interno para a aplicação foi de apenas 744kb, demonstrando que o aplicativo é bem leve e que não irá consumir tanto espaço com a sua instalação.

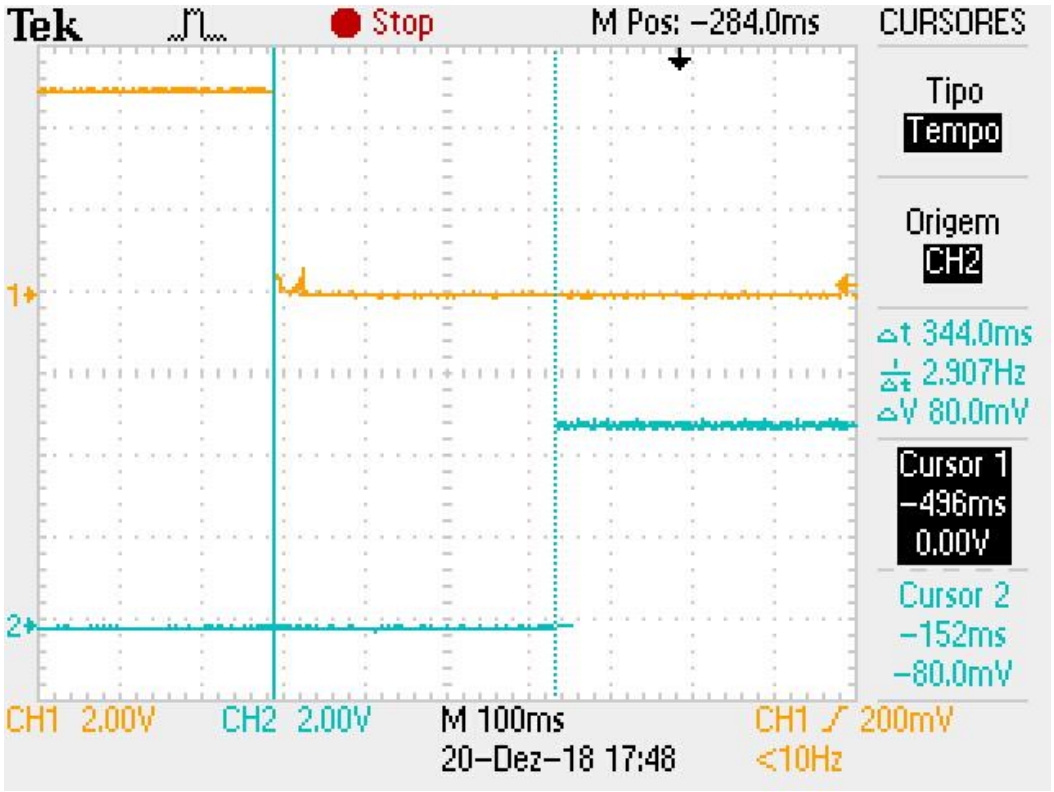
Figura 37: Armazenamento Interno e Uso de Dados pelo Aplicativo

Fonte: Elaborado pelo Autor

Para o teste de funcionalidade do sistema com a sua utilização por qualquer pessoa, foram feitos testes com usuários de idades e sexos diferentes. Todos os comandos foram recebidos corretamente, sendo mínima a ocorrência de erros de recepção dos mesmos, somente nos casos em que o condutor não fala a frase corretamente, sendo perfeitamente utilizável por qualquer pessoa.

Para a medição do tempo de acionamento do comando de voz, foi feito uma pequena modificação nos softwares do sistema. Esta modificação baseia-se em, ao ser clicado em um botão na CPU FATEC, um comando era enviado para o smartphone, ele respondia com o mesmo comando para o microcontrolador que, ao receber a resposta, invertia o estado lógico de um determinado pino. Este tempo foi medido através de um osciloscópio desde o acionamento do botão até a mudança de estado do pino, com o tempo total foi de 344ms como mostrado na *Figura 38*.

Figura 38: Tempo de Acionamento do Comando de Voz



Fonte: Elaborada pelo Autor

Uma vez recebido o csaractere pelo aplicativo no smartphone, ele reenvia o caractere que, ao ser novamente recebido pelo microcontrolador, seta um pino livre no microcontrolador para confirmação da recepção. Como o tempo medido foi o tempo de ida e volta da informação, com um total de 344ms, o tempo de acionamento do comando de voz pode ser definida pela metade deste tempo, isto é, aproximadamente 172ms.

Também foram feitos testes de funcionamento do aplicativo em locais com níveis de ruído. Este teste foi feito com o auxílio de um aplicativo instalado em um smartphone, pois não tínhamos um fonômetro a disposição. Os ruídos testados foram na ordem de aproximadamente 70dB com pessoas conversando, reprodução de música e ruídos de outros veículos sendo obtidos bons resultados. Para ruídos acima do informado, o processo de captura de voz fica ativo por mais tempo que o esperado. Nesta situação, a aplicação se comporta de duas formas diferentes:

1. *Para ruídos de motor de carro, moto e buzina, o aplicativo mantém ativo o reconhecimento de voz enquanto não forem reduzidos os níveis de ruído, mas se o comando dado pelo usuário for nítido o aplicativo reconhece o que foi dito e efetua o comando, lembrando que se utilizado com os vidros do carro fechado o nível de ruído no interior do mesmo reduz consideravelmente;*
2. *Para música o aplicativo passa a não conseguir o mesmo desempenho da opção anterior, porque além do som ser constante, também contém voz, que dificulta o reconhecimento do comando dado.*

A comunicação entre o smartphone e o módulo Bluetooth permite que os dados sejam trocados de uma forma que não se percam dados durante o seu uso. O módulo de relés está funcionando bem, possibilitando a troca de dados entre o HC-06, o aplicativo e o microcontrolador, sendo que o chaveamento para acionamentos dos comandos do veículo é feito totalmente pelos relés que são acionados pelo PIC em conjunto com os circuitos integrados ULN2003, CIs que são utilizados com frequência em drives de motor de passo.

5 CONCLUSÃO

Pensando em acrescentar uma opção a mais entre as existentes no mercado de adaptações veiculares voltado para pessoas com deficiência física ou mobilidade reduzida, foi desenvolvido um sistema capaz de receber comandos de voz do usuário do veículo para os mais diversos acionamentos internos do mesmo. O estudo demonstrou que o Sistema Operacional Android, por ser de código aberto, foi utilizado no desenvolvimento do sistema por ser uma ferramenta de fácil utilização e que disponibiliza diversos recursos embarcados em seu sistema nativo, bastando apenas o desenvolvimento da aplicação pretendida para que seja feita a comunicação entre o dispositivo onde está instalada com o mundo exterior, no caso o veículo, através de um módulo embarcado capaz de fazer os acionamentos no veículo.

De um modo geral, a aplicação desenvolvida para o smartphone desempenha o seu papel de forma correta, sendo que a possibilidade de ocorrência de erros de recepção do comando dado pelo usuário se resume apenas à existência constante de níveis de ruído acima de 70dB ou múltiplas pessoas conversando ao mesmo tempo. Além destas opções, a recepção dos comandos e a comunicação com o módulo não apresenta erros.

A comunicação sem fio entre o smartphone e o módulo está estável e totalmente funcional, não apresentando falhas de comunicação ou mensagens perdidas, fazendo com que todos os comandos fossem recebidos pelo aplicativo e transmitidos ao módulo corretamente. Desta forma, fica evidente que os objetivos relacionados ao desenvolvimento do sistema foram alcançados com êxito por demonstrar total usabilidade do sistema, podendo ser utilizado com total segurança e confiabilidade.

6 PROPOSTAS FUTURAS

Para continuidade desse trabalho são apresentadas as seguintes propostas:

- Substituição do Módulo Bluetooth HC-06 pelo HC-05 para possibilitar a conexão automática entre o sistema e o smartphone, retirando do processo a etapa de conexão com o módulo;
- Otimização do aplicativo para inicialização e conexão automática da aplicação com o HC-05;
- Instalação no veículo para verificação funcional.

7 BIBLIOGRAFIA

ALSELECTRO. Voice Recognition board HM2007 – How to control a motor using voice commands. **Alselectro**, 2014. Disponivel em: <https://alselectro.wordpress.com/2014/01/31/voice-recognition-board-hm2007-how-to-control-a-motor-using-voice-commands/>. Acesso em: 11 Set 2018.

ANDROID. História do Android - Donut. **Android**, 2018. Disponivel em: <https://www.android.com/history/#/donut>. Acesso em: 11 Set 2018.

ANDROID. História do Android - Eclair. **Android**, 2018. Disponivel em: <https://www.android.com/history/#/eclair>. Acesso em: 11 Set 2018.

ANDROID. História do Android - Jelly Bean. **Android**, 2018. Disponivel em: <https://www.android.com/history/#/jellybean>. Acesso em: 11 Set 2018.

ANDROID DEVELOPER. Android Studio. **Android Developer**, 2018. Disponivel em: <https://developer.android.com/studio/?hl=pt-br>. Acesso em: 11 Set 2018.

ANDROID DEVELOPERS. Configure sua compilação. **Android Developers**, 2018. Disponivel em: <https://developer.android.com/studio/build/>. Acesso em: 12 Set 2018.

ANDROID DEVELOPERS. Zipalign. **Android Developers**, 2018. Disponivel em: <https://developer.android.com/studio/command-line/zipalign>. Acesso em: 12 Set 2018.

ANDROID STUDIO. Executar aplicativos no Android Emulator. **Developers Android**, 2018. Disponivel em: <https://developer.android.com/studio/run/emulator?hl=pt-br>. Acesso em: 12 Set 2018.

BRASIL. CONSTITUIÇÃO DA REPÚBLICA FEDERATIVA DO BRASIL, Brasília, DF, 05 Out 1988.

BRASIL. Decreto nº 5.296 de 2 de Dezembro de 2004. **critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida**, Brasília, 02 Dez 2004.

BRASIL. Estatuto da Cidade, Brasília, n. 3ª, 30 Junho 2008. Disponivel em: <https://www2.senado.leg.br/bdsf/bitstream/handle/id/70317/000070317.pdf?sequence=6>. Acesso em: 22 Julho 2018.

BRASIL. Lei nº 13.146, de 6 de Julho de 2015. **Estatuto da Pessoa com Deficiência**, Brasília, 06 Jul 2015.

CÂMARA DOS DEPUTADOS. Estudos Estratégicos. **O Desafio da Mobilidade Urbana**, Brasília, 5º vol, 2015.

CANALTECH. Google cria sistema de reconhecimento de voz que dispensa conexão com a internet. **Canaltech**, 2016. Disponivel em: <https://canaltech.com.br/mercado/google-cria-sistema-de-reconhecimento-de-voz-que-dispensa-conexao-com-a-internet-59787/>. Acesso em: 11 Set 2018.

CIRIACO, D. Play Store aumentou 30% e App Store diminuiu 5% em 2017. **Tecmundo**, 2018. Disponivel em: <https://www.tecmundo.com.br/software/128979-pesquisa-play-store-aumentou-30-app-store-diminuiu-5-2017.htm>. Acesso em: 24 Set 2018.

COMPANHIA DE ENGENHARIA DE TRÁFEGO. Manual de Sinalização Urbana. **Regulamentação de Estacionamento e Parada**, 10, Novembro 2016. Disponivel em: <http://www.cetsp.com.br/media/392055/msu-vol-10-parte-5-deficiente-fisico-rev-05.pdf>. Acesso em: 09 Set 2018.

CONVENÇÃO INTERNACIONAL DOS DIREITOS DA PESSOA COM DEFICIÊNCIA. Protocolo Facultativo à Convenção sobre os Direitos das Pessoas com Deficiência: Decreto Legislativo nº 186, de 09 de julho de 2008: Decreto nº 6.949, de 25 de agosto de 2009, Brasília, n. 4ª Edição, 2012. Disponivel em: <http://www.pessoacomdeficiencia.gov.br/app/sites/default/files/publicacoes/convencaopessoascomdeficiencia.pdf>. Acesso em: 09 set. 2018.

DETRAN - ES. CNH para Deficientes. **Detran ES**, 2015. Disponivel em: <https://detran.es.gov.br/cnh-para-deficientes>. Acesso em: 09 Set 2018.

DIODES INCORPORATED. High Voltage, High Current ULN2004A, Jan 2017. Disponivel em: <https://www.diodes.com/assets/Datasheets/ULN200xA.pdf>. Acesso em: 23 Fev 2019.

DO BIT AO BYTE. Bluetooth HC-06 no Arduino. **Do Bit ao Byte**, 2016. Disponivel em: <https://www.dobitaobyte.com.br/bluetooth-hc-06-no-arduino/>. Acesso em: 24 Set 2018.

FAIRCHILD. BC337/BC338 NPN Epitaxial Silicon Transistor, Set 2015. Disponivel em: <http://www.mouser.com/ds/2/149/BC337-193546.pdf>. Acesso em: 23 Fev 2019.

FARIA, M. D. A PERSPECTIVA TRANSFORMATIVA NA ANÁLISE DE SIGNIFICADOS DE PRODUTOS DE TECNOLOGIA ASSISTIVA, Rio de Janeiro, 21 Mar 2015. Disponivel em:

<http://200.229.32.55/index.php/economiaegestao/article/view/P.1984-6606.2015v15n40p172/8715>. Acesso em: 09 set. 2018.

FUNDAÇÃO GETÚLIO VARGAS. Mercado Brasileiro de TI e Uso Nas Empresas. **Resumo de Notícias**, São Paulo, 29, 19 Abr 2018. 23. Disponivel em: <https://eaesp.fgv.br/sites/eaesp.fgv.br/files/pesti2018gvciappt.pdf>. Acesso em: 11 Set 2018.

GOMES, A. E. G.; REZENDE, L. K.; TORTORELLI, M. F. P. Acessibilidade e Deficiência. **Análise de Documentos Normativos**, São Paulo, 2016. Disponivel em: <http://editorarevistas.mackenzie.br/index.php/cpgdd/article/download/11197/6928>. Acesso em: 21 Abr 2017.

GOOGLE. Conheça o seu Google Assistente. **Google Assistente**, 2017. Disponivel em: <https://assistant.google.com/intl/pt_br/>. Acesso em: 11 Set 2018.

HONGFA. Subminiature High Power Relay, 2018. Disponivel em: <http://www.hongfa.com/pro/pdf/HF3FA_en.pdf>. Acesso em: 23 Fev 2019.

INFO WESTER. Tecnologia Bluetooth: o que é e como funciona? **Info Wester**, 2018. Disponivel em: <https://www.infowester.com/bluetooth.php>. Acesso em: 14 Set 2018.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Censo Demográfico 2010. **Características Gerais da População, Religião e Pessoas com Deficiência**, Rio de Janeiro, 2012. Disponivel em: <https://biblioteca.ibge.gov.br/visualizacao/periodicos/94/cd_2010_religiao_deficiencia.pdf>. Acesso em: 05 Set 2018.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Relatórios Metodológicos. **Metodologia do Censo Demográfico 2010**, Rio de Janeiro, 41, n. 2ª ed, 2016. 18. Disponivel em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv95987.pdf>. Acesso em: 08 Set 2018.

KIVI BRASIL. Acelerador de aro sob o volante K5 easy-fit. **KIVI Brasil**, 2016. Disponivel em: <http://www.kivi.com.br/produtos/acelerador_de_aro_sob_o_volante_k5_easy-fit>. Acesso em: 09 Set 2018.

KIVI BRASIL. Banco de transferência manual RM205. **KIVI Brasil**, 2016. Disponivel em: <http://www.kivi.com.br/produtos/banco_de_transferencia_manual_rm205>. Acesso em: 09 Set 2018.

KIVI BRASIL. Banco de Transferência Manual RM205. **KIVI Brasil**, 2016. Disponivel em: <http://www.kivi.com.br/produtos/banco_de_transferencia_manual_rm205>. Acesso em: 10 Set 2018.

KIVI BRASIL. Central de comandos auxiliares PV3000. **KIVI Brasil**, 2016. Disponivel em: <http://www.kivi.com.br/produtos/central_de_comandos_auxiliares_pv3000>. Acesso em: 09 Set 2018.

KIVI BRASIL. Controles auxiliares via cabo 197/10. **KIVI Brasil**, 2016. Disponivel em: <http://www.kivi.com.br/produtos/controles_auxiliares_via_cabo_19710>. Acesso em: 10 Set 2018.

LAUREANO, M. **Máquinas Virtuais e Emuladores - Conceitos, Técnicas e Aplicações**. Paraná: Novatec Editora, 2006. 18 p. ISBN ISBN: 85-7522-098-5. Disponivel em: <http://www.mlaureano.org/aulas_material/so/livro_vm_laureano.pdf>. Acesso em: 11 Set 2018.

MICROCHIP. MPLAB® X Integrated Development Environment (IDE). **Microchip**, 2018. Disponivel em: <http://www.microchip.com/mplab/mplab-x-ide>. Acesso em: 14 Set 2018.

MICROCHIP TECHNOLOGY. PIC18F2455/2550/4455/4550 Datasheet, 2006. Disponivel em: <https://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf>. Acesso em: 23 Fev 2019.

MOVIMENTO POPULAR INCLUA-SE. Sobre a Acessibilidade. **Inclua-se**, 2011. Disponivel em: <http://incluase.blogspot.com/2008/10/simbolo-internacional-de-acesso-sai.html>. Acesso em: 09 set. 2018.

OLHAR DIGITAL. Entenda o que é o Android puro, suas diferenças e vantagens. **Olhar Digital**, 2018. Disponivel em: <https://olhardigital.com.br/video/entenda-o-que-e-o-android-puro-suas-diferencas-e-vantagens/76137>. Acesso em: 11 Set 2018.

OLIVEIRA, W. Comunicação entre Módulos Bluetooth HC-05 e HC-06. **Embarcados**, 2016. Disponivel em: <https://www.embarcados.com.br/modulos-bluetooth-hc-05-e-hc-06/>. Acesso em: 24 Set 2018.

PAPPA, M. F.; CHIROLI, D. M. D. G. Mobilidade Urbana Sustentável, Maringá, 25-28 Out 2011. Disponivel em: <https://www.unicesumar.edu.br/epcc-2011/wp-content/uploads/sites/86/2016/07/marcia_fernanda_pappa2.pdf>. Acesso em: 21 Abr 2017.

PCDEF. Quatorze modelos exclusivos para pessoas com deficiência (PCD). **PCDef**, 2017. Disponivel em: <https://pcdef.com.br/servicos/veiculos-adaptados/quatorze-modelos-exclusivos-para-pessoas-com-deficiencia-pcd/>. Acesso em: 09 Set 2018.

QUATRO RODAS. Como funciona a isenção de impostos para PCD? **Quatro Rodas**, 2018. Disponivel em: <https://quatrorodas.abril.com.br/auto-servico/como-funciona-a-isencao-de-impostos-para-deficientes/>. Acesso em: 09 Set 2018.

RICARTE, I. L. M. Compiladores. **Unicamp**, 2003. Disponivel em: <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node37.html>. Acesso em: 14 Set 2018.

ROBOCORE TECNOLOGIA. Bluetooth HC-05 com Arduino: Comunicando com PC. **Robocore Tecnologia**, 2016. Disponivel em: <https://www.robocore.net/tutorials/42>. Acesso em: 14 Set 2018.

ROBOCORE TECNOLOGIA. Comparação Entre Protocolos de Comunicação Serial. **Robocore Tecnologia**, 2016. Disponivel em: <https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html>. Acesso em: 14 Set 2018.

SANTOS, N. D. S. F.; NOIA, A. C. Mobilidade Urbana e Política Pública. **Uma Análise das Ações Realizadas Pelo Poder Público na Cidade de Itabuna, Bahia**, Ilhéus, 2015. Disponivel em: <http://www.uesc.br/eventos/vsemeconomista/anais/gt3-6.pdf>. Acesso em: 21 Abr 2017.

SEABRA, L. O.; TACO, P. W. G. Mobilidade Urbana no Brasil, Brasília, 2014. 1-16. Disponivel em: <http://redpgv.coppe.ufrj.br/index.php/es/produccion/articulos-cientificos/2014-1/807-mobilidade-urbana-no-brasil-antecedentes-e-perspectivas-a-luz-dos-mecanismos-de-gestao/file>. Acesso em: 19 abr. 2017.

SIQUEIRA, T. S. D. Bluetooth – Características, Protocolos e Funcionamento, Campinas, 15 Jun 2006. Disponivel em: <http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf>. Acesso em: 14 Set 2018.

STATISTA. Number of available applications in the Google Play Store from December 2009 to June 2018. **Statista**, 2018. Disponivel em: <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>. Acesso em: 11 Set 2018.

TECMUNDO. Google Assistente ganha integração mais profunda com apps de terceiros. **Tecmundo**, 2017. Disponivel em:

<https://www.tecmundo.com.br/software/127616-google-assistente-ganha-integracao-profunda-apps-terceiros.htm>. Acesso em: 11 Set 2018.

VISHAY SEMICONDUCTORS. Optocoupler, Phototransistor Output, with Base Connection, 10 Jan 2010. Disponivel em: <https://www.vishay.com/docs/83725/4n25.pdf>. Acesso em: 23 Fev 2019.

WAGNER, L. C. et al. Ciência em Movimento. **Acessibilidade de Pessoas com Deficiência: O Olhar de uma Comunidade da Periferia de Porto Alegre**, Porto Alegre, 2010. Disponivel em: <https://www.metodista.br/revistas/revistas-ipa/index.php/RS/article/view/94/58>. Acesso em: 21 Abr 2017.

WCED. **World Commission on Environment and Development**, 2015. Disponivel em: <https://sustainabledevelopment.un.org/milestones/wced>. Acesso em: 08 Set 2018.

APÊNDICE A – PROGRAMAÇÃO PIC18F4550 EM LINGUAGEM C

```
/*
 * MainTaskMng.c
 * Author: Edson Ramos
 * Author: Bruno Majores
 * Author: Guilherme Alencar
 * Data: Mar/2018
 * Description:
 * This software is a task manager for 8-bit PIC microcontrollers.
 * WARNING: be careful when programming the tasks. Tasks must be as
fast as possible (due to real-time features of this task manager)
 */

/* Changed by Weslley Torres in order to fit into hardware developed
on FATEC Santo André    */
/* Author: Weslley Torres
                                                                    */
/* Date: 08/May/ 2016
                                                                    */



#include "displayLCD.h"
#include "Config.h"
#include "Config_EUSART.h"
#include "Serial_EUSART.h"
#include "displayLCD.h"
#include <xc.h>
/*****************************************************************
***************************/
/*
Variables
*/
/*****************************************************************
***************************/
/*Store function pointers of Task.*/
void (*ScheduledTasks[NUMBER_OF_TASKS])(void);

/*Store task´s times (time period to execute)*/
int TimeScheduledTasks[NUMBER_OF_TASKS];

/*Stores recent task´s times ("time to execute" each task) */
int RecentTasksTimeToExecute[NUMBER_OF_TASKS];

/*Tells if TImer0 interrupt has been generated*/
char Timer0IntGeneraed;

/*  Tells  if  a  task  in  executing  (used  to  check  task  timeout
validation)*/
volatile char TaskIsExecuting;

int TaskTimeout;

unsigned char TimerBreathingLight = 100;

/*****************************************************************
***************************/
```

```
      /*************************************************************************
**************************/
      /*
      Function Prototipe
      */
      /*************************************************************************
**************************/

      void ConfigTimer0(void);
      void InitTasks(void);
      void ExecuteTask(void);
      void ConfigUC(void);

      void Task1(void);
      void Task2(void);
      void Task3(void);
      void Task4(void);
      void Task5(void);
      void Task6(void);

      /*************************************************************************
**************************/

      /**
       *  \brief Initialize and schedule tasks
       *
       *  \return none
       *
       *  \details Initialize the scheduller
       */
      void InitTasks(void)
      {
            //init function pointers of tasks
            ScheduledTasks[INDEX_TAREFA_1] = Task1;
            ScheduledTasks[INDEX_TAREFA_2] = Task2;
          ScheduledTasks[INDEX_TAREFA_3] = Task3;
          ScheduledTasks[INDEX_TAREFA_4] = Task4;
            ScheduledTasks[INDEX_TAREFA_5] = Task5;
          ScheduledTasks[INDEX_TAREFA_6] = Task6;

            /*init temporization values of each task. These values do no
            change during execution*/
            TimeScheduledTasks[INDEX_TAREFA_1] = TIME_TO_EXECUTE_TASK1;
            TimeScheduledTasks[INDEX_TAREFA_2] = TIME_TO_EXECUTE_TASK2;
          TimeScheduledTasks[INDEX_TAREFA_3] = TIME_TO_EXECUTE_TASK3;
          TimeScheduledTasks[INDEX_TAREFA_4] = TIME_TO_EXECUTE_TASK4;
            TimeScheduledTasks[INDEX_TAREFA_5] = TIME_TO_EXECUTE_TASK5;
          TimeScheduledTasks[INDEX_TAREFA_6] = TIME_TO_EXECUTE_TASK6;

            /*init recent temporization values of each task. These values
will
            change during execution (they´re used to decide which task must
be executed)*/

            RecentTasksTimeToExecute[INDEX_TAREFA_1]                      =
TIME_TO_EXECUTE_TASK1;
            RecentTasksTimeToExecute[INDEX_TAREFA_2]                      =
TIME_TO_EXECUTE_TASK2;
          RecentTasksTimeToExecute[INDEX_TAREFA_3] = TIME_TO_EXECUTE_TASK3;
          RecentTasksTimeToExecute[INDEX_TAREFA_4] = TIME_TO_EXECUTE_TASK4;
            RecentTasksTimeToExecute[INDEX_TAREFA_5]                      =
```

```
TIME_TO_EXECUTE_TASK5;
          RecentTasksTimeToExecute[INDEX_TAREFA_6] = TIME_TO_EXECUTE_TASK6;

          //It indicates that there´s no task executing
          TaskIsExecuting = NO;
      }


      /**
       *  \brief ExecuteTask
       *
       *  \return none
       *
       *  \details Execute tasks assigned to the system
       */
      void ExecuteTask(void)
      {
          char i;
            for (i=0; i<NUMBER_OF_TASKS; i++)
            {
                  //Check if it´s time to execute a task
                  if        ((ScheduledTasks[i]        !=        0)        &&
(RecentTasksTimeToExecute[i] == 0))
              {
                        TaskIsExecuting = YES;
                  TaskTimeout = TASK_TIMEOUT;
                  ScheduledTasks[i]();  //executes the task
                  TaskIsExecuting = NO;
                        RecentTasksTimeToExecute[i] = TimeScheduledTasks[i];
//reagendamento da tarefa
                  }
            }
      }


      /**
       *  \brief Task1
       *
       *  \return none
       *
       *  \details Executes task 1
       */
      void Task1(void)
      {
          //LATBbits.LATB0 = ~LATBbits.LATB0;
          if(!BOTAO1 && BT1<5 && !EstadoBotao){
              BT1++;
          }
          if(!BOTAO1 && BT1>=5 && !EstadoBotao){
              EstadoBotao = 1;
              escreve_EUSART('@');
          }
          if(BOTAO1 && EstadoBotao){
              BT1 = 0;
              EstadoBotao = 0;
          }
      }


      /**
       *  \brief Task2
```

```
         *
         *  \return none
         *
         *  \details Executes task 2
         */
        void Task2(void)
        {   //Lavador Dianteiro
            if(FlagLavDiant && EstadoLavDiant == 0)
                EstadoLavDiant++;
            if(EstadoLavDiant > 0 && EstadoLavDiant <= 6 && FlagLavDiant == 1){
                LAV_DIANTEIRO = ON;
                LIMP_DIANTEIRO = OFF;
                EstadoLavDiant++;
            }
            if(EstadoLavDiant > 6 && EstadoLavDiant <= 20 && FlagLavDiant ==
1){
                LAV_DIANTEIRO = OFF;
                LIMP_DIANTEIRO = ON;
                EstadoLavDiant++;
            }
            if(EstadoLavDiant >= 20){
                LAV_DIANTEIRO = OFF;
                LIMP_DIANTEIRO = OFF;
                EstadoLavDiant=0;
                FlagLavDiant=0;
            }
        }

        /**
         *  \brief Task2
         *
         *  \return none
         *
         *  \details Executes task 2
         */
        void Task3(void)
        {   //Lavador Traseiro
            if(FlagLavTras && EstadoLavTras == 0)
                EstadoLavTras++;
            if(EstadoLavTras > 0 && EstadoLavTras <= 6 && FlagLavTras == 1){
                LAV_TRASEIRO = ON;
                LIMP_TRASEIRO = OFF;
                EstadoLavTras++;
            }
            if(EstadoLavTras > 6 && EstadoLavTras <= 20 && FlagLavTras == 1){
                LAV_TRASEIRO = OFF;
                LIMP_TRASEIRO = ON;
                EstadoLavTras++;
            }
            if(EstadoLavTras >= 20){
                LAV_TRASEIRO = OFF;
                LIMP_TRASEIRO = OFF;
                EstadoLavTras=0;
                FlagLavTras=0;
            }
        }

        /**
         *  \brief Task1
         *
         *  \return none
```

```
 *
 *  \details Executes task 1
 */
void Task4(void)
{
    if(EstadoSetas == 0){
        SETA_DIREITA =  OFF;
        SETA_ESQUERDA =  OFF;
        EstadoSetaDireita = 0;
        EstadoSetaEsquerda = 0;
    }
    if(EstadoSetas == 1){
        SETA_ESQUERDA =  OFF;
        EstadoSetaEsquerda = 0;
        EstadoSetaDireita = ~EstadoSetaDireita;
        if(EstadoSetaDireita){
            SETA_DIREITA =  ON;
        }
        if(!EstadoSetaDireita){
            SETA_DIREITA =  OFF;
        }
    }
    if(EstadoSetas == 2){
        SETA_DIREITA =  OFF;
        EstadoSetaDireita = 0;
        EstadoSetaEsquerda = ~EstadoSetaEsquerda;
        if(EstadoSetaEsquerda){
            SETA_ESQUERDA =  ON;
        }
        if(!EstadoSetaEsquerda){
            SETA_ESQUERDA =  OFF;
        }
    }
    if(EstadoSetas == 3){
        EstadoEmergencia = ~EstadoEmergencia;
        if(EstadoEmergencia){
            SETA_DIREITA =  ON;
            SETA_ESQUERDA =  ON;
        }
        if(!EstadoEmergencia){
            SETA_DIREITA =  OFF;
            SETA_ESQUERDA =  OFF;
        }
    }
    PosicaoCursorLCD(2,19);
    EscreveInteiroLCD(EstadoSetas);
}

/**
 *  \brief Task1
 *
 *  \return none
 *
 *  \details Executes task 1
 */
void Task5(void)
{

}

/**
```

```
        *   \brief Task1
        *
        *   \return none
        *
        *   \details Executes task 1
        */
       void Task6(void)
       {

       }


       /**
        *   \brief ISR
        *
        *   \return none
        *
        *   \details Interrupt service routine (be careful snce PIC has only
one interrupt vector, based on that, the way you check the       interrupt
flag will be the priority)
        */
       void interrupt isr(void)
       {
           char  i;

           //check if Timer0 interupt was triggered
             if (TMR0IF && T0IF)
           {
               T0IF = 0;                   //set trigger for Timer0 interrupt
(so it can be generated again)
               INTCONbits.TMR0IF = 0;
               TMR0 = TIMER0_INIT_VALUE; //Set initial value of TMR0 register
(for counting 1ms, as we need)

               //LATBbits.LATB1 = ~LATBbits.LATB1;

               if(TimerBreathingLight)     TimerBreathingLight--;
               if(TimerBreathingLight <= 0)
               {
                    //LATBbits.LATB4 = ~LATBbits.LATB4;
                    TimerBreathingLight = 100;
               }
               Timer0IntGeneraed = YES;

                   //Refresh "time to execute" of each task
                   for (i=0; i<NUMBER_OF_TASKS; i++)
                   {
                         if (RecentTasksTimeToExecute[i] > 0)
                               RecentTasksTimeToExecute[i]--;

                   }
           }

           //check if RX interupt was triggered
           if(PIR1bits.RCIF == 1){
               PIR1bits.RCIF = 0;

               Rec = LerEUSART();
               if(Rec == 'A'){
                   EstadoSetas = 1;
                   escreve_EUSART('A');
```

```
}
if(Rec == 'a'){
    EstadoSetas = 0;
    escreve_EUSART('a');
}
if(Rec == 'B'){
    EstadoSetas = 2;
    escreve_EUSART('B');
}
if(Rec == 'b'){
    EstadoSetas = 0;
    escreve_EUSART('b');
}
if(Rec == 'C'){
    LANTERNA =  ON;
    escreve_EUSART('C');
}
if(Rec == 'c'){
    LANTERNA =  OFF;
    escreve_EUSART('c');
}
if(Rec == 'D'){
    LANTERNA =  ON;
    FAROL_BAIXO = ON;
    escreve_EUSART('D');
}
if(Rec == 'd'){
    FAROL_BAIXO = OFF;
    escreve_EUSART('d');
}
if(Rec == 'E'){
    LANTERNA =  ON;
    FAROL_BAIXO = ON;
    FAROL_ALTO =  ON;
    escreve_EUSART('E');
}
if(Rec == 'e'){
    FAROL_ALTO =  OFF;
    escreve_EUSART('e');
}
if(Rec == 'F'){
    LANTERNA =  ON;
    FAROL_NEBLINA = ON;
    escreve_EUSART('F');
}
if(Rec == 'f'){
    FAROL_NEBLINA = OFF;
    escreve_EUSART('f');
}
if(Rec == 'G'){
    FlagLavDiant = 1;
    EstadoLavDiant = 1;
    escreve_EUSART('G');
}
if(Rec == 'g'){
    escreve_EUSART('g');
}
if(Rec == 'H'){
    FlagLavTras = 1;
    EstadoLavTras = 1;
    escreve_EUSART('H');
```

```
}
if(Rec == 'h'){
    LAV_TRASEIRO = OFF;
    escreve_EUSART('h');
}
if(Rec == 'I'){
    LIMP_TRASEIRO =  ON;
    escreve_EUSART('I');
}
if(Rec == 'i'){
    LIMP_TRASEIRO =  OFF;
    escreve_EUSART('i');
}
if(Rec == 'J'){
    LIMP_DIANTEIRO = ON;
    escreve_EUSART('J');
}
if(Rec == 'j'){
    LIMP_DIANTEIRO = OFF;
    escreve_EUSART('j');
}
if(Rec == 'K'){
    INTERMITENTE = ON;
    escreve_EUSART('K');
}
if(Rec == 'k'){
    INTERMITENTE = OFF;
    escreve_EUSART('k');
}
if(Rec == 'L'){
    DESEMBACADOR =  ON;
    escreve_EUSART('L');
}
if(Rec == 'l'){
    DESEMBACADOR =  OFF;
    escreve_EUSART('l');
}
if(Rec == 'M'){
    EstadoSetas = 3;
    escreve_EUSART('M');
}
if(Rec == 'm'){
    EstadoSetas = 0;
    escreve_EUSART('m');
}
if(Rec == 'N'){
    PINO_RESERVA =  ON;
    escreve_EUSART('M');
}
if(Rec == 'n'){
    PINO_RESERVA =  OFF;
    escreve_EUSART('m');
}
if(Rec == 'Z'){
    EstadoSetas = 0;
    SETA_DIREITA = OFF;
    SETA_ESQUERDA = OFF;
    LANTERNA = OFF;
    FAROL_BAIXO = OFF;
    FAROL_ALTO = OFF;
    FAROL_NEBLINA = OFF;
```

```
            LAV_DIANTEIRO = OFF;
            LAV_TRASEIRO = OFF;
            LIMP_TRASEIRO = OFF;
            LIMP_DIANTEIRO = OFF;
            INTERMITENTE = OFF;
            DESEMBACADOR = OFF;
            EMERGENCIA1 = OFF;
            EMERGENCIA2 = OFF;
            PINO_RESERVA = OFF;
            escreve_EUSART('Z');
        }
        if(Rec == '@'){
            escreve_EUSART('@');
        }
    }
}


 /**
 *  \brief main
 *
 *  \return none
 *
 *  \details Main function of the system
 */
void main(void){
    char Frase1[17] = "PROJETO TCC 2018";
    char Frase2[17] = "Inicializacao OK";
      ConfiguraLCD();         //configure LCD display
    DesligaCursor();
    PosicaoCursorLCD(1,1);
    EscreveFraseRamLCD(Frase1);
    PosicaoCursorLCD(2,1);
    EscreveFraseRamLCD(Frase2);
    __delay_ms(1000);
    InitTasks();            //Initialize tasks
      ConfigTimer0();         //Configure Timer0

      ConfigUC();             //Configure Microcontroller

    ConfiguraEUSART();      //Configure EUSART

    //main loop
    while(1)
    {
        //Verification: check if there´s a task to be executed
        if ((Timer0IntGeneraed == YES)  && (NUMBER_OF_TASKS))
        {
            Timer0IntGeneraed = NO;
            ExecuteTask();
        }
    }
}

/**
 *  \brief ConfigTimer0
 *
 *  \return none
 *
 *  \details Configure timer0 registers
 */
```

```
      void ConfigTimer0(void){
          T0CON    = 0b11000101;
          INTCON   = 0b11100000;
          INTCON2  = 0b10000000;
          TMR0     = TIMER0_INIT_VALUE;  //Set initial value of TMR0 register
(for counting 1ms, as we need)
          Timer0IntGeneraed = NO;
      }

      /**
       *  \brief Init_Ports
       *
       *  \return none
       *
       *  \details Initialize ports
       */
      void ConfigUC(void)
      {
          ADCON0bits.ADON = 0;                  //Desabilita conversor AD
          CMCON  = 0x07;                        //Desabilita os comparadores
          ADCON1 = 0x0F;                           //Todos os pinos como I/O
Digital
          TRISB  = 0x00;                        //Todo PORTB como Saída
          LATB   = 0x00;                        //Inicializa o PORTB em HIGH
          TRISA  = 0x00;                        //Todo PORTA como Saída
          LATA   = 0x00;                        //Inicializa o PORTA em HIGH
          TRISDbits.TRISD3 = 0;
          LATDbits.LATD3 = 0;
      }
```

APÊNDICE B – PROGRAMAÇÃO APLICATIVO ANDROID STUDIO

```
package edsonramos.com.tccfatec2018;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.os.Handler;
import android.os.Message;
import android.speech.RecognizerIntent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    //Declaração dos objetos
    private ImageButton btn_Conectar;
    private TextView tv_Conectar;
    private ImageView img_Produto;
    private Button button;
    //End Objetos

    //Variáveis utilizadas para controle do RequestCode
    private static final int ATIVA_BLUETOOTH = 1;
    private static final int SOLICITA_CONEXAO = 2;
    private static final int MESSAGE_READ = 3;
    private static final int RECONHECIMENTO_DE_VOZ = 4;

    //Objetos e Variáveis para o bluetooth
    String mensagemBT;
    ConnectedThread connectedThread;
    Handler mHandler;
    boolean conexao = false;
    private static String MAC = null;
    BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
    BluetoothDevice btDevice = null;
    BluetoothSocket btSocket = null;
    UUID    btUUID    =    UUID.fromString("00001101-0000-1000-8000-
00805f9b34fb");

    //Método onCreate
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
            //Associação das variáveis com os objetos
            btn_Conectar                                                =
(ImageButton)findViewById(R.id.btn_Conectar);
            button                  = (Button)findViewById(R.id.button);
            tv_Conectar                                                 =
(TextView)findViewById(R.id.tv_Conectar);
            img_Produto                                                 =
(ImageView)findViewById(R.id.img_Produto);

            //Bluetooth disponível?
            if (btAdapter == null) {
                Toast.makeText(this, "Que pena! Hardware Bluetooth Não Está
Funcionando.", Toast.LENGTH_SHORT).show();
            }//end Bluetooth disponível

            //Bluetooth está ativado?
            if(!btAdapter.isEnabled()) {//Sim
                Intent          enableBtIntent          =          new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, ATIVA_BLUETOOTH);
                Toast.makeText(this,     "Solicitando     Ativação     do
Bluetooth...", Toast.LENGTH_SHORT).show();
            } else {//Não
                Toast.makeText(this,     "Bluetooth     Já     Ativado.",
Toast.LENGTH_SHORT).show();
            }//End Bluetooth está ativado?

            //Conexão
            btn_Conectar.setOnClickListener(new View.OnClickListener(){
                @Override
                public void onClick(View view){
                    if(conexao){
                        //desconectar
                        try{
                            btSocket.close();
                            conexao = false;
                            tv_Conectar.setText("CONECTAR");

img_Produto.setImageResource(R.mipmap.produto);

                            Toast.makeText(MainActivity.this,   "Bluetooth
Desconectado.", Toast.LENGTH_SHORT).show();
                        }catch(IOException erro){
                            Toast.makeText(MainActivity.this, "Ocorreu um
Erro.", Toast.LENGTH_SHORT).show();
                        }
                    }
                    else{
                        //Conectar
                        Intent  abreLista  =  new  Intent(MainActivity.this,
ListaDispositivos.class);
                        startActivityForResult(abreLista,
SOLICITA_CONEXAO);
                    }
                }
            });


            //Recepção de string por bluetooth
            mHandler = new Handler(){
```

```
                @Override
                public void handleMessage(Message msg) {
                    if(msg.what == MESSAGE_READ){
                        String recepcao = (String) msg.obj;
                        mensagemBT = recepcao;
                        //Compara os valores de recepção para acionamentos
na aplicação
                        if(recepcao.equals("@")){
                            button.callOnClick();
                        }
                    }

                }
            };

        }


        @Override
        protected void onActivityResult(int requestCode, int resultCode,
Intent data) {

            switch(requestCode){

                case ATIVA_BLUETOOTH:
                    if(resultCode == Activity.RESULT_OK) {
                        Toast.makeText(this,     "Bluetooth     Ativado.",
Toast.LENGTH_SHORT).show();
                    }
                    else {
                        Toast.makeText(this, "Bluetooth Não Ativado.\n O
App Será Encerrado.", Toast.LENGTH_SHORT).show();
                        finish();
                    }
                    break;

                case SOLICITA_CONEXAO:
                    if(resultCode == Activity.RESULT_OK) {
                        MAC                                                =
data.getExtras().getString(ListaDispositivos.ENDERECO_MAC);
                        btDevice = btAdapter.getRemoteDevice(MAC);

                        try{
                            btSocket                                       =
btDevice.createRfcommSocketToServiceRecord(btUUID);
                            btSocket.connect();
                            conexao = true;
                            connectedThread             =             new
ConnectedThread(btSocket);
                            connectedThread.start();
                            Toast.makeText(this,    "Bluetooth    Conectado
com:\n" + MAC, Toast.LENGTH_SHORT).show();
                            tv_Conectar.setText("DESCONECTAR");

img_Produto.setImageResource(R.drawable.microfone);

                        }catch(IOException erro){
                            conexao = false;
                            Toast.makeText(this,    "Erro    de    Conexão.",
Toast.LENGTH_SHORT).show();
                        }
```

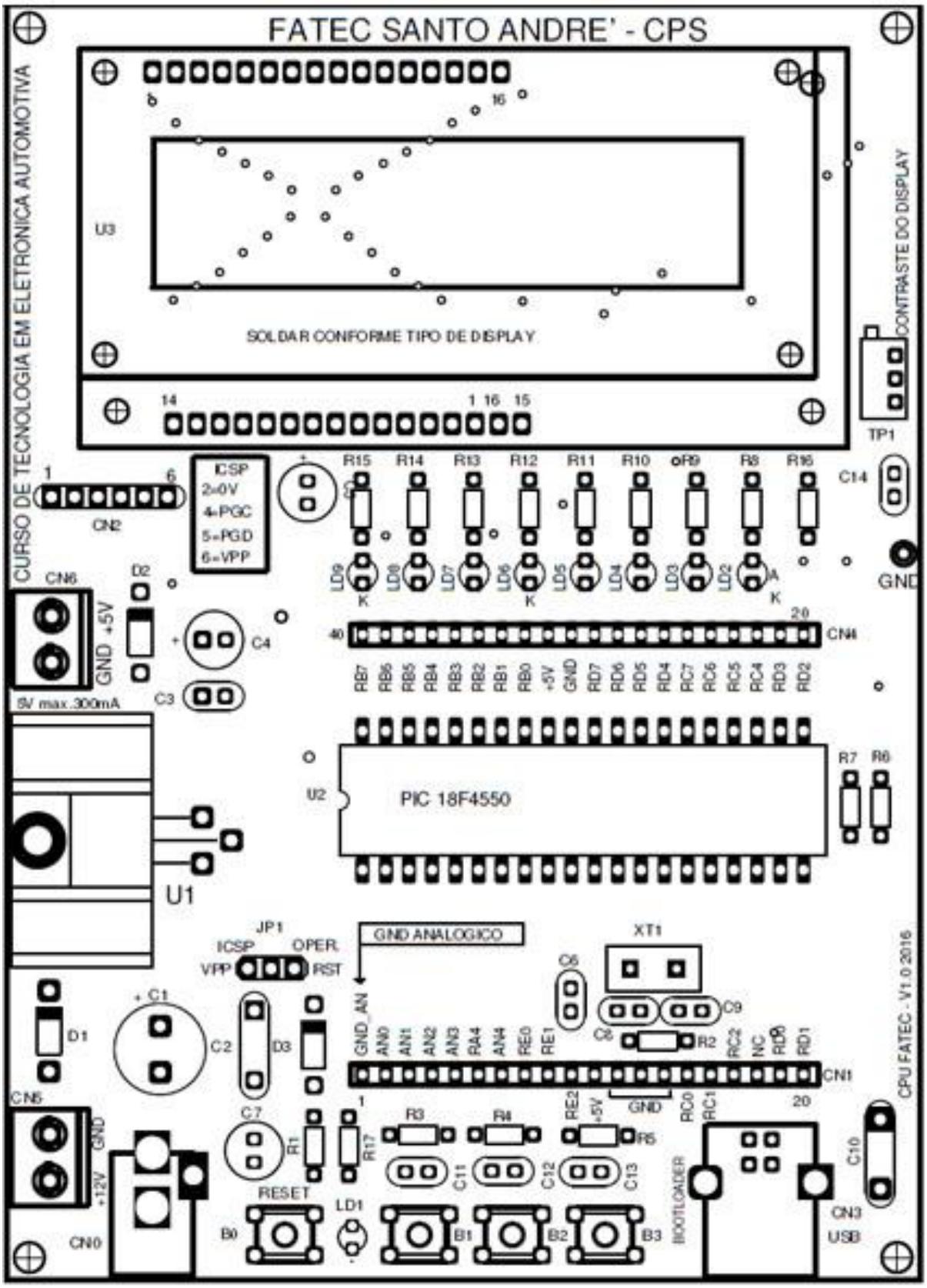
```
                    }
                    else{
                        Toast.makeText(this, "Falha ao Obter o MAC.",
Toast.LENGTH_SHORT).show();
                    }
                    break;

                case RECONHECIMENTO_DE_VOZ:
                    if (resultCode == RESULT_OK) {
                        ArrayList<String> speech =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                        String strSpeech2Text = speech.get(0);
                        if(strSpeech2Text.equals("liga seta direita")){
                            connectedThread.enviar("A");
                            Toast.makeText(this, "Seta Direita Ligada",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("desliga seta direita")){
                            connectedThread.enviar("a");
                            Toast.makeText(this, "Seta Direita Desligada",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("liga seta esquerda")){
                            connectedThread.enviar("B");
                            Toast.makeText(this, "Seta Esquerda Ligada",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("desliga seta
esquerda")){
                            connectedThread.enviar("b");
                            Toast.makeText(this, "Seta Esquerda
Desligada", Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("lanterna")){
                            connectedThread.enviar("C");
                            Toast.makeText(this, "Lanterna Ligada",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("desliga lanterna")){
                            connectedThread.enviar("c");
                            Toast.makeText(this, "Lanterna Desligada",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("liga farol baixo")){
                            connectedThread.enviar("D");
                            Toast.makeText(this, "Farol Baixo Ligado",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("desliga farol baixo")){
                            connectedThread.enviar("d");
                            Toast.makeText(this, "Farol Baixo Desligado",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("liga farol alto")){
                            connectedThread.enviar("E");
                            Toast.makeText(this, "Farol Alto Ligado",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("desliga farol alto")){
                            connectedThread.enviar("e");
                            Toast.makeText(this, "Farol Alto Desligado",
```

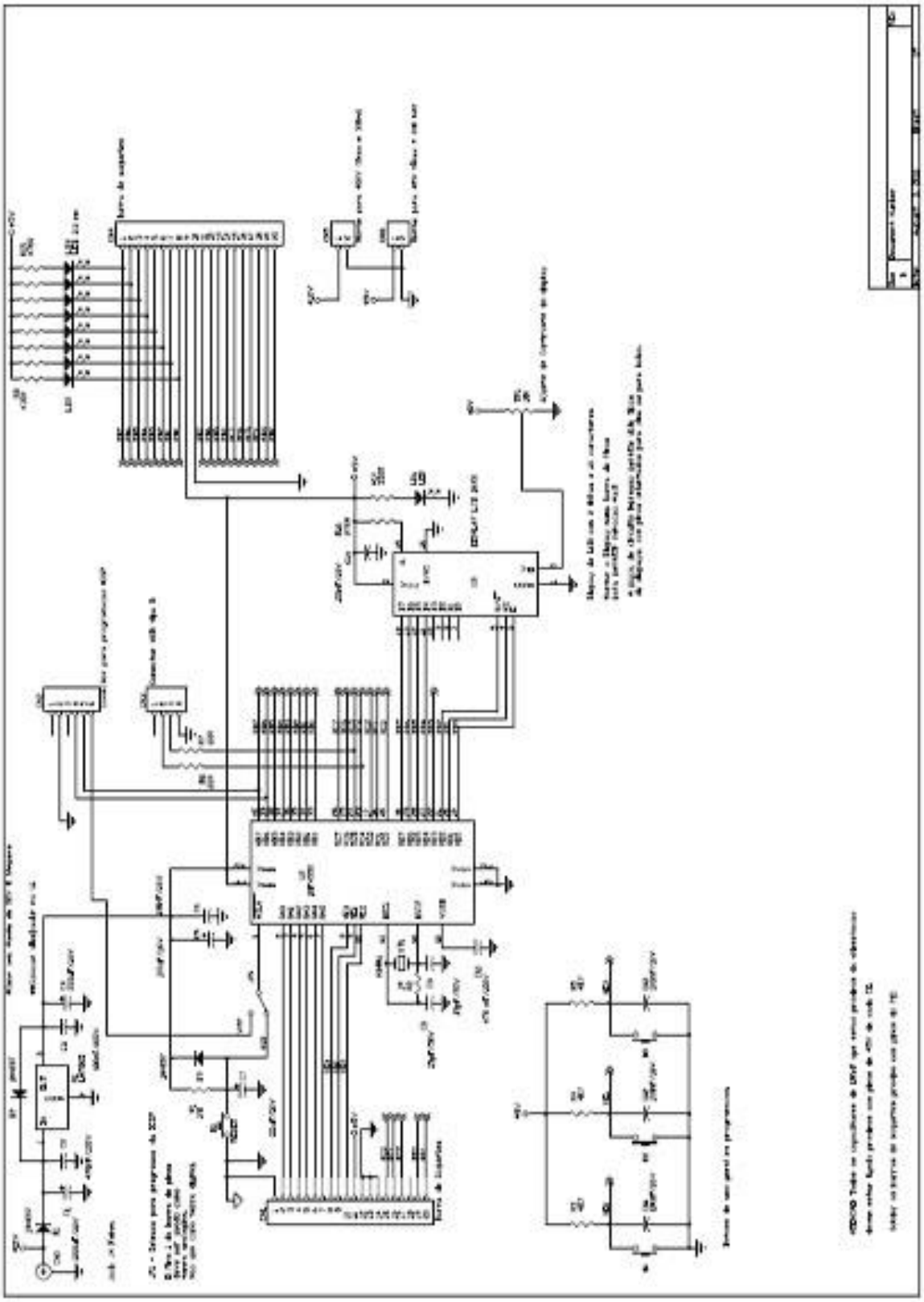
```
Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("liga      farol      de
neblina")){
                                connectedThread.enviar("F");
                                Toast.makeText(this,    "Farol    de    Neblina
Ligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("desliga     farol     de
neblina")){
                                connectedThread.enviar("f");
                                Toast.makeText(this,    "Farol    de    Neblina
Desligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("liga             lavador
dianteiro")){
                                connectedThread.enviar("G");
                                Toast.makeText(this,     "Lavador     Dianteiro
Ligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("liga             lavador
traseiro")){
                                connectedThread.enviar("H");
                                Toast.makeText(this,     "Lavador     Traseiro
Ligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("liga             limpador
traseiro")){
                                connectedThread.enviar("I");
                                Toast.makeText(this,     "Limpador     Traseiro
Ligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("desliga         limpador
traseiro")){
                                connectedThread.enviar("i");
                                Toast.makeText(this,     "Limpador     Traseiro
Desligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("liga             limpador
dianteiro")){
                                connectedThread.enviar("J");
                                Toast.makeText(this,    "Limpador    Dianteiro
Ligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("desliga         limpador
dianteiro")){
                                connectedThread.enviar("j");
                                Toast.makeText(this,    "Limpador    Dianteiro
Desligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("liga intermitente")){
                                connectedThread.enviar("K");
                                Toast.makeText(this,  "Limpador  Intermitente
Ligado", Toast.LENGTH_SHORT).show();
                            }
                            if(strSpeech2Text.equals("desliga
intermitente")){
                                connectedThread.enviar("k");
                                Toast.makeText(this,  "Limpador  Intermitente
Desligado", Toast.LENGTH_SHORT).show();
                            }
```

```
                        if(strSpeech2Text.equals("liga desembaçador")){
                            connectedThread.enviar("L");
                            Toast.makeText(this,  "Desembaçador  Ligado",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("desliga
desembaçador")){
                            connectedThread.enviar("l");
                            Toast.makeText(this, "Desembaçador Desligado",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("liga emergência")){
                            connectedThread.enviar("M");
                            Toast.makeText(this,   "Emergência   Ligada",
Toast.LENGTH_SHORT).show();
                        }
                        if(strSpeech2Text.equals("desliga
desembaçador")){
                            connectedThread.enviar("m");
                            Toast.makeText(this,  "Emergência  Desligada",
Toast.LENGTH_SHORT).show();
                        }
                    }
                    break;
            }
        }

        //Classe que controla a linha de conexão para a recepção via
bluetooth
        private class ConnectedThread extends Thread {
            private final InputStream mmInStream;
            private final OutputStream mmOutStream;

            public ConnectedThread(BluetoothSocket socket) {
                InputStream tmpIn = null;
                OutputStream tmpOut = null;

                // Get the input and output streams, using temp objects
because
                // member streams are final
                try {
                    tmpIn = socket.getInputStream();
                    tmpOut = socket.getOutputStream();
                } catch (IOException e) { }

                mmInStream = tmpIn;
                mmOutStream = tmpOut;
            }

            //Liga o processo bluetooth
            public void run() {
                byte[] buffer = new byte[1024];  // buffer store for the
stream
                int bytes; // bytes returned from read()

                // Mantém o bluetooth ativo sempre aguardando o recebimento
de dados
                while (true) {
                    try {
                        // Read from the InputStream
                        bytes = mmInStream.read(buffer);
```

```
                            String dadosBT_string = new String(buffer, 0,
bytes);
                            // Send the obtained bytes to the UI activity
                            mHandler.obtainMessage(MESSAGE_READ, bytes, -1,
dadosBT_string).sendToTarget();
                            //textView.setText(dadosBT_string);
                        } catch (IOException e) {
                            break;
                        }
                    }
                }

                //Função que envia dados pelo Bluetooth
                public void enviar(String dadosEnviar) {
                    byte[] msgBuffer = dadosEnviar.getBytes();
                    try {
                        mmOutStream.write(msgBuffer);
                    } catch (IOException e) { }
                }
            }

            //Configuração do comando de voz
            public void ComandoVoz(View view) {
                Intent intentActionRecognizeSpeech = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
                // Configura Linguagem (Português-Brasil)

intentActionRecognizeSpeech.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
"pt-BR");
                try {
                    startActivityForResult(intentActionRecognizeSpeech,
RECONHECIMENTO_DE_VOZ);
                } catch (ActivityNotFoundException a) {
                    Toast.makeText(this, "Dispositivo não suporta
recohecimento de voz.", Toast.LENGTH_SHORT).show();
                }
            }
        }
```

ANEXO A – PLACA ELETRÔNICA FATEC





ANEXO B – DATASHEET HC-06

Guangzhou HC Information Technology Co., Ltd.

Product Data Sheet

Module Data Sheet

Rev 1

1. 0	2.0	2.1	2.2				
2006/6/18	2006/9/6	2010/4/22	2011/4/6				

DRAWN BY :	Ling Xin		**MODEL :** HC-06	
CHECKED BY :	Eric Huang		**Description::** BC04 has external 8M Flash and EDR module HC-06 is industrial, and compatible with civil HC-04	
APPD. BY:	Simon Mok		**REV: 2.0**	**Page :**
Former version introduction	HC-06 is the higher version of LV_BC_2.0. Linvor is the former of wavesen.			

Contents

2. Feature

3. Pins description

4. The parameters and mode of product

5. Block diagram

6. Debugging device

7. Characteristic of test

8. Test diagram

9. AT command set

1. Product’s picture

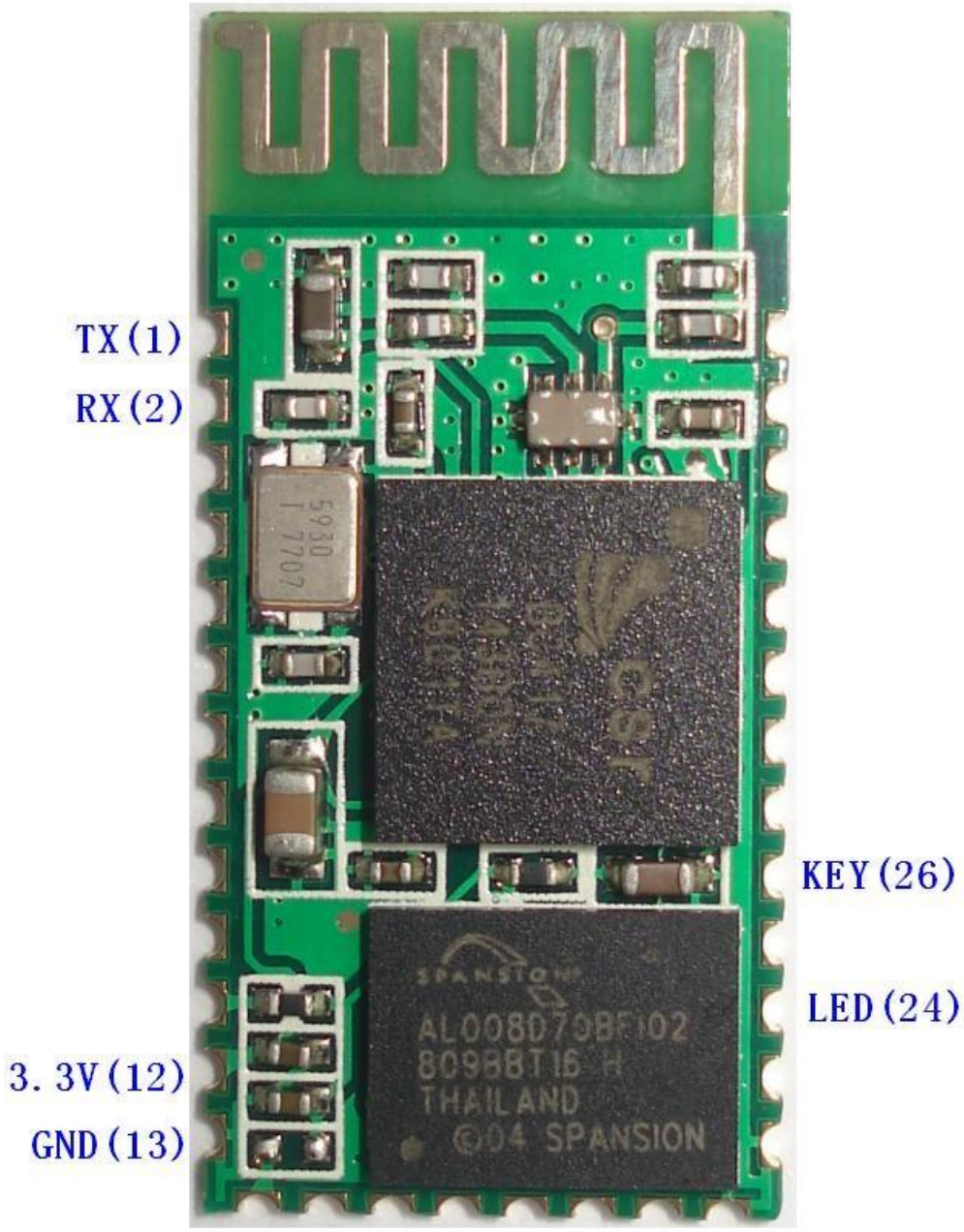


Figure 1 A Bluetooth module

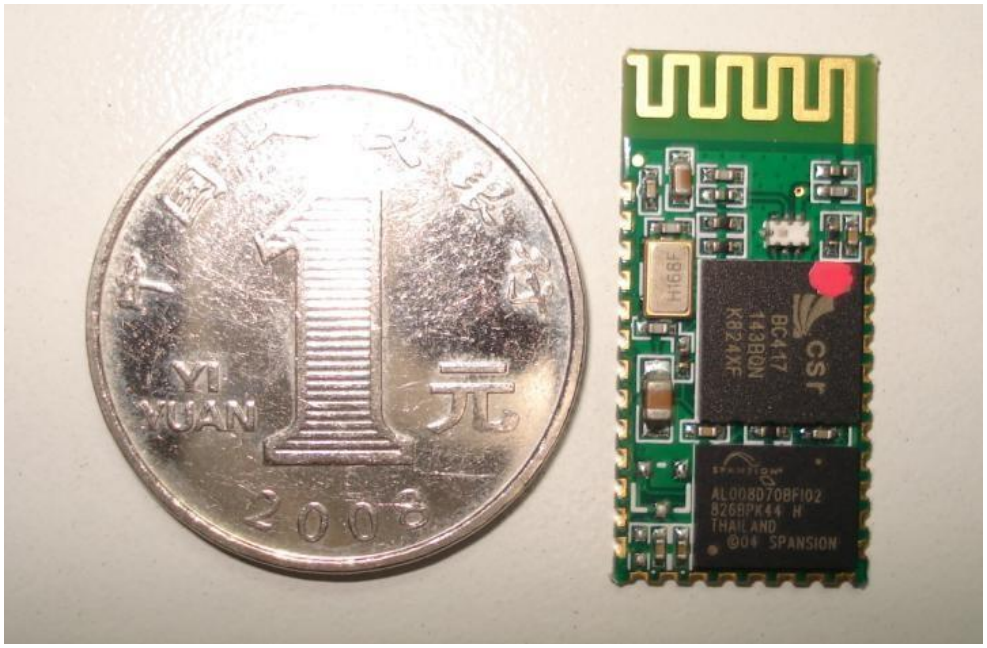


Figure 2. A Bluetooth module size

Figure 3 50 pieces chips in an anti-static blister package.

2. Feature

- Wireless transceiver
  - Sensitivity (Bit error rate) can reach -80dBm.

- The change range of output's power: -4 - +6dBm.

- Function description (perfect Bluetooth solution)
  - Has an EDR module; and the change range of modulation depth: 2Mbps - 3Mbps.
  - Has a build-in 2.4GHz antenna; user needn't test antenna.
  - Has the external 8Mbit FLASH
  - Can work at the low voltage (3.1V~4.2V). The current in pairing is in the range of 30～40mA.

    The current in communication is 8mA.
  - Standard HCI Port (UART or USB)
  - USB Protocol: Full Speed USB1.1, Compliant With 2.0 ➢ This module can be used in the SMD.
  - It's made through RoHS process.
  - The board PIN is half hole size.
  - Has a 2.4GHz digital wireless transceiver.
  - Bases at CSR BC04 Bluetooth technology.
  - Has the function of adaptive frequency hopping.
  - Small (27mm×13mm×2mm)
  - Peripherals circuit is simple.
  - It's at the Bluetooth class 2 power level.
  - Storage temperature range: -40 ℃ - 85℃，work temperature range: -25 ℃ - +75℃ ➢ Any wave inter Interference: 2.4MHz，the power of emitting: 3 dBm.
  - Bit error rate: 0. Only the signal decays at the transmission link, bit error may be produced. For example, when RS232 or TTL is being processed, some signals may decay.

- Low power consumption
- Has high-performance wireless transceiver system
- Low Cost
- Application fields:
  - Bluetooth Car Handsfree Device

- Bluetooth GPS
- Bluetooth PCMCIA , USB Dongle
- Bluetooth Data Transfer ● Software
- CSR

3. PINs description

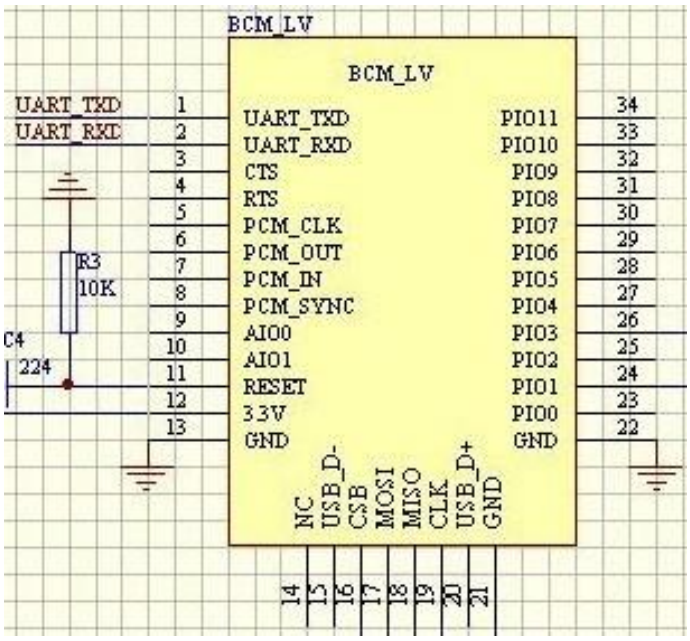


Figure 3 PIN configuration

The PINs at this block diagram is as same as the physical one.

PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pot	
1V8	14	VDD	Integrated 1.8V (+) supply with On-chip linear regulator output within 1.7-1.9V	
VCC	12	3.3V		
AIO0	9	Bi-Directional	Programmable input/output line	

AIO1	10	Bi-Directional	Programmable input/output line	

PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	CLK_REQ
PIO7	30	Bi-Directional	Programmable input/output line	CLK_OUT
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	
RESETB	11	CMOS Input with weak intemal pull-down		
UART_RTS	4	CMOS output, tri-stable with weak internal pull-up	UART request to send, active low	
UART_CTS	3	CMOS input with weak internal pull-down	UART clear to send, active low	
UART_RX	2	CMOS input with weak internal pull-down	UART Data input	

UART_TX	1	CMOS output, Tri-stable with weak internal pull-up	UART Data output	
SPI_MOSI	17	CMOS input with weak internal pull-down	Serial peripheral interface data input	
SPI_CSB	16	CMOS input with weak internal	Chip select for serial peripheral interface, active low	
		pull-up		
SPI_CLK	19	CMOS input with weak internal pull-down	Serial peripheral interface clock	
SPI_MISO	18	CMOS input with weak internal pull-down	Serial peripheral interface data Output	
USB_-	15	Bi-Directional		
USB_+	20	Bi-Directional		
1.8V	14		1.8V external power supply input	Default : 1.8V internal powe r supply.
PCM_CLK	5	Bi-Directional		
PCM_OUT	6	CMOS output		
PCM_IN	7	CMOS Input		

PCM_SYNC	8	Bi-Directional		

The parameters and mode of product

LINVOR BLUE T

www.linvor.com

Bluetooth Module

Bluetooth

CSR,BC417143B

V 2.0

2006/09/6

蓝牙 **RF** 模块

1. 采用 CSR BC4 +8M FLASH 方案
2. 具有 PIO0-PIO11、AIO0、AIO1、USB、PCM、UART 及 SPI 接口，模块内置 8MFLASH，功能强大，用户可定制软件,适用于各种蓝牙设备，内置 RF 天线,便于调试。

蓝牙协议版本	Bluetooth Specifcation V2.0 With EDR
USB 协议 USB Protocol	Full Speed USB V1.1 Compliant With USB V2.0
频率	2.4Ghz ISM band
调制方式	GFSK(Gaussian Frequency Shift Keying)
发射功率	-4 ->4 dBm, Class 2
灵敏度	≤-80dBm at 0.1% BER
通讯速率	Asynchronous:2Mbps(Max)
供电电源	3.3V
工作温度	-20~+55 Centigrade
封装尺寸	27mmX13mmX2mm

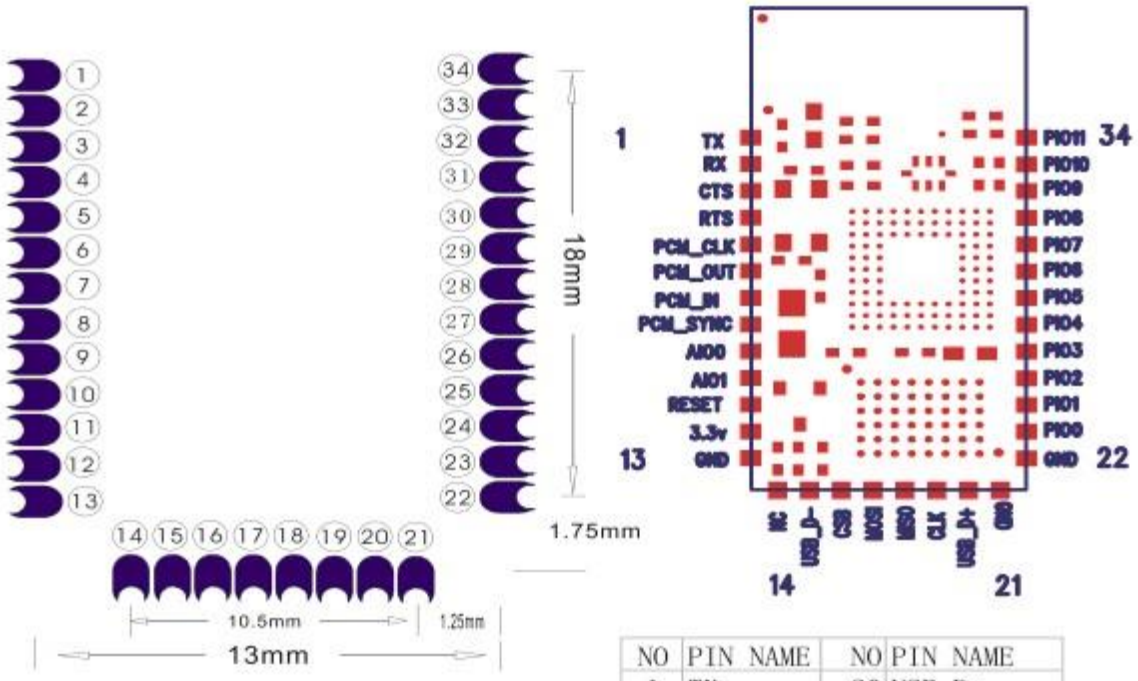
If you want more information, please visit www.wavesen.com.

LINVOR BLUE T

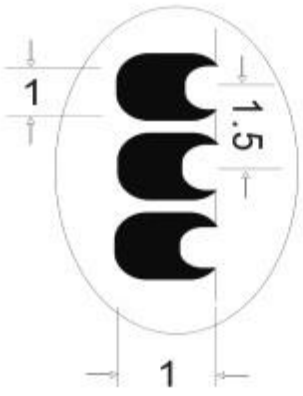
www.linvor.com

LV-BC-2.0

单位： mm



NO	PIN NAME	NO	PIN NAME
1	TX	20	USB D+
2	RX	21	GND
3	CTS	22	GND
4	RTS	23	PIO0
5	PCM CLK	24	PIO1
6	PCM OUT	25	PIO2
7	PCM IN	26	PIO3
8	PCM SYNC	27	PIO4
9	AIO0	28	PIO5
10	AIO1	29	PIO6
11	RESET	30	PIO7
12	3.3V	31	PIO8
13	GND	32	PIO9
14	NC	33	PIO10
15	USB D-	34	PIO11
16	CSB		
17	MOSI		
18	MISO		
19	CLK		



PCB Layout 请参考实物

5. Block diagram

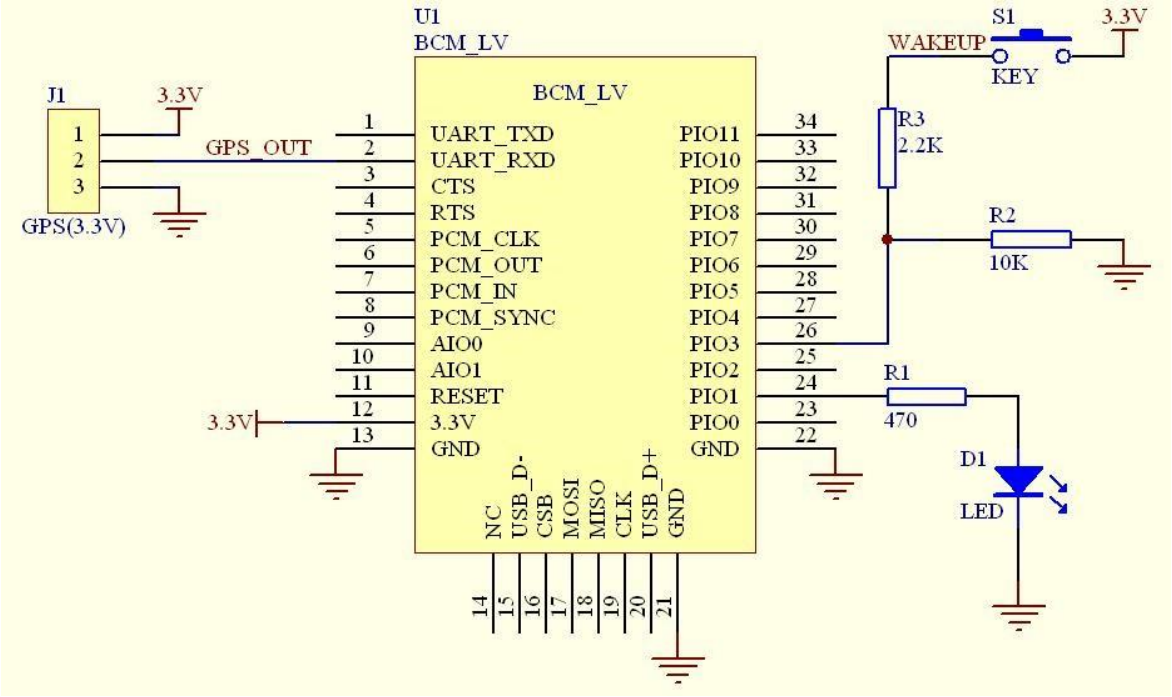


Figure 5 Block diagram 1

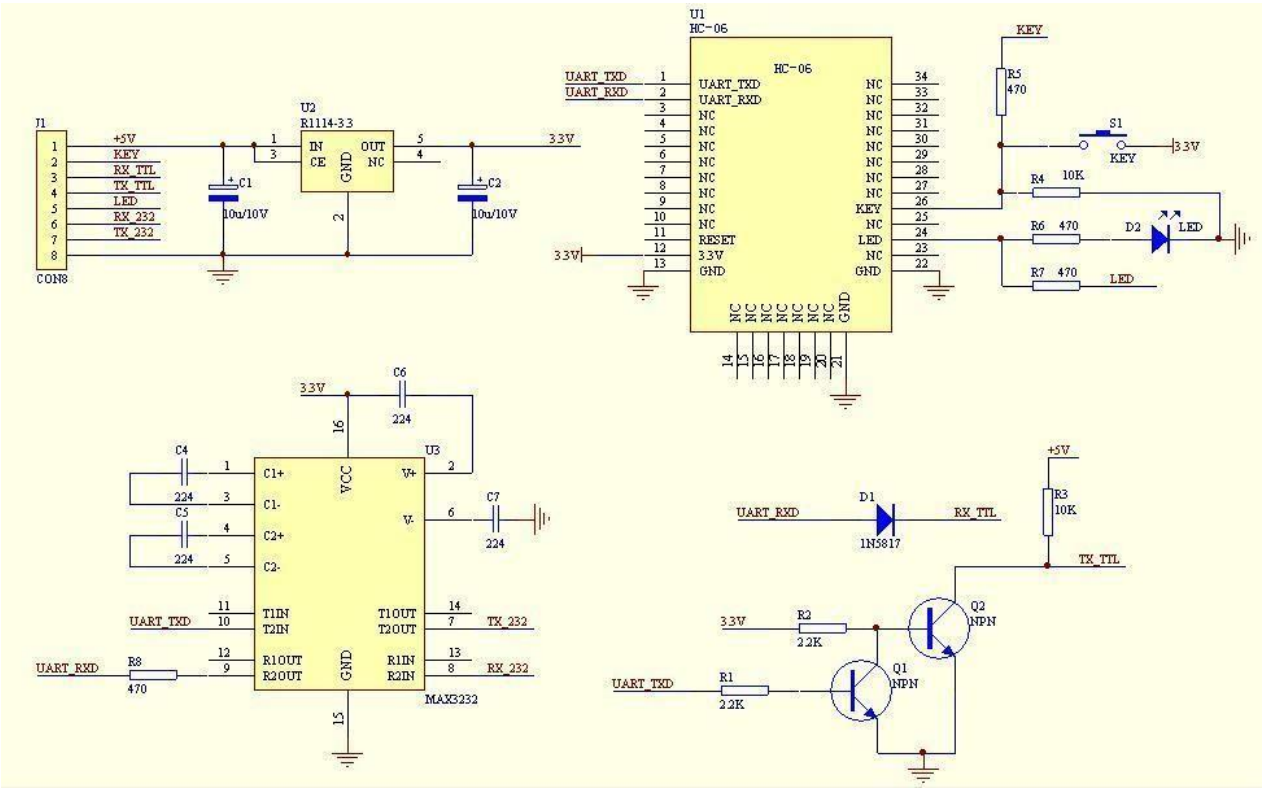


Figure 5 Block diagram 2

HC-04/06 master device has a function of remembering the last paired slave device. As a master device, it will search the last paired salve device until the connection is built. But if the WAKEUP bottom is pressed, HC-04/06 will lose the memory and research the new slave device.

6. Debugging device

6.1 Device

PC, hardware, 3G，3G Frequency Counter (SP3386), 3.15V DC power supply, Shielding，Bluetooth Test box.

6.2 Software

7. Characteristic of test

Test Condition 25℃ RH 65%

	Min	Typ	Max	Unit
1. Carrier Freq. (*ISM Band*)	2.4		2.4835	MHz
2. RF O/P Power	-6	2	4	dBm
3. Step size of Power control	2		8	dB
4. Freq. Offset (*Typical Carrier freq.*)	-75		75	KHz
5. Carrier Freq. drift (*Hopping on, drift rate/50uS*)	-20		20	KHz
1 slot packet	-25		25	KHz
3 slot packet	-40		-40	KHz
6. Average Freq. Deviations (*Hopping off, modulation*)	140		175	KHz
Freq. Deviation	115			KHz
Ratio of Freq. Deviation	0.8			
7 . Receive Sensitivity @< 0.1% BER*(Bit error rate)*	-83			dBm

8. Test diagram

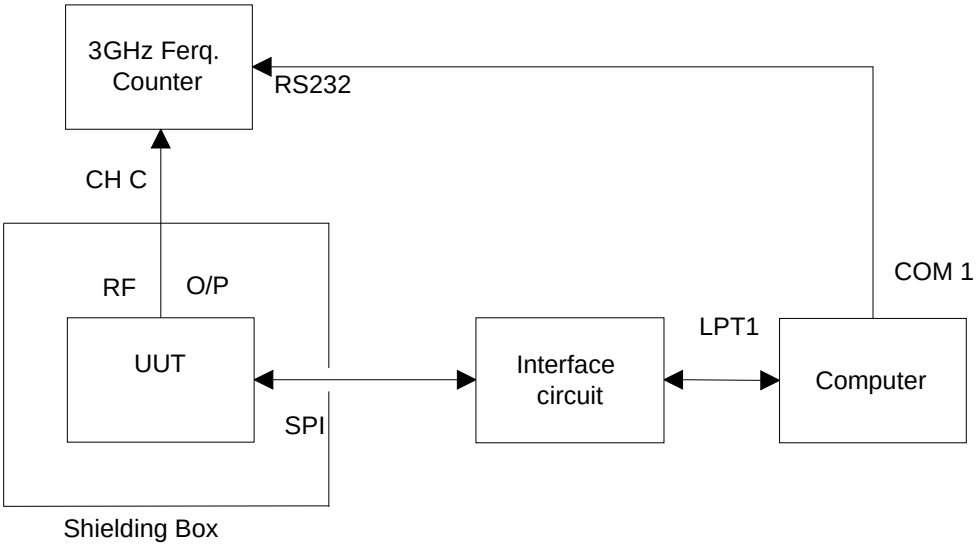


Fig 1. Programming and Freq. Alignment

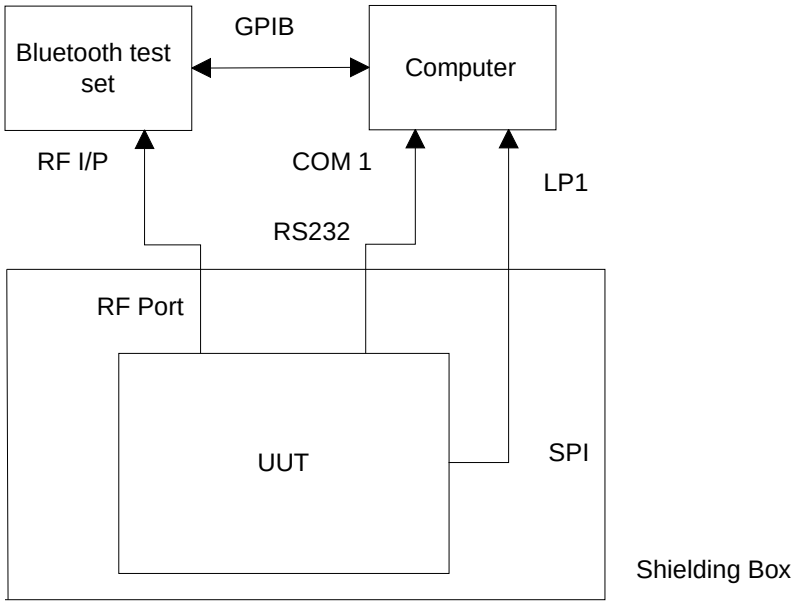


Fig 2 RF parameter Test Procedure

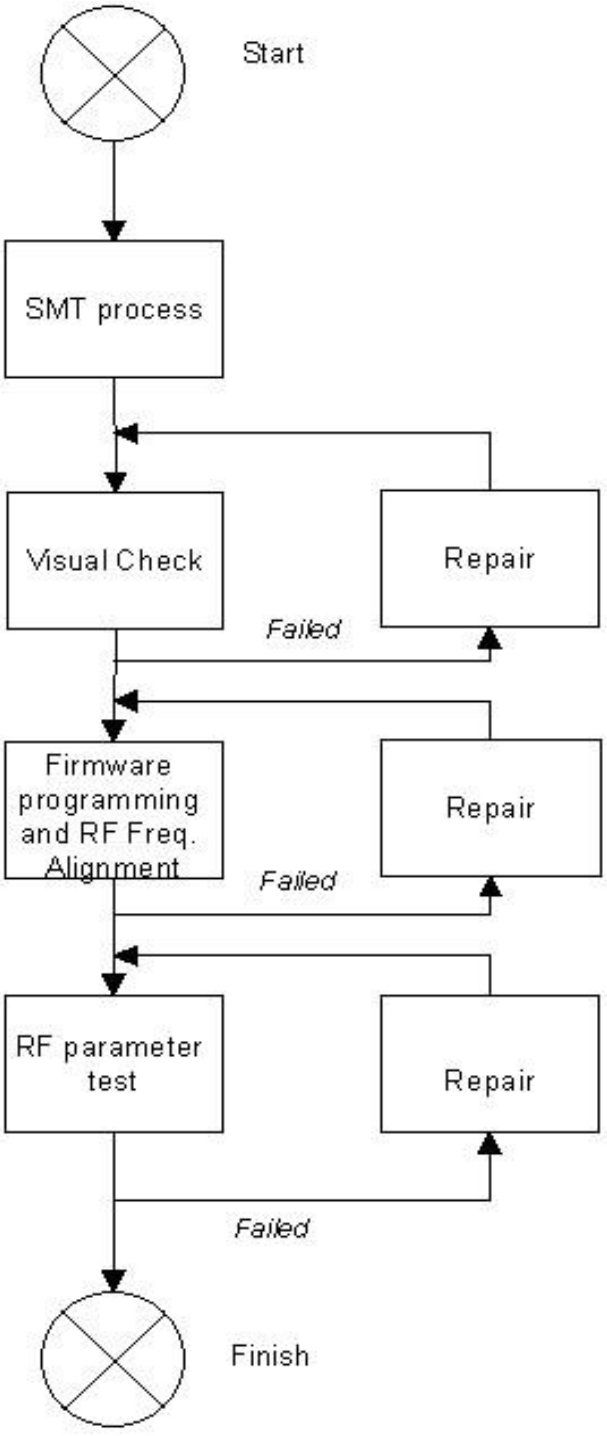


Fig 3 Assemble/Alignment/Testing Flow Chart

9. AT command set

The way to the AT command mode: supply power to the module, it will enter to the AT mode if it needn't pair. The interval of command is about 1 second.

Default parameter: Baud rate:9600N81, ID: linvor, Password:1234 1. Test communication

Send: AT (please send it every second)

Back: OK

2. Reset the Bluetooth serial baud rate

Send:

AT+BAUD1

Back: OK1200

Send: AT+BAUD2

Back: OK2400

……

1---------1200

2---------2400

3---------4800

4---------9600 (Default)

5---------19200

6---------38400

7---------57600

8---------115200

9---------230400

A---------460800

B---------921600

C---------1382400

PC can't support the baud rate lager than 115200. The solution is: make the MCU have higher baud rate (lager than 115200) through programming, and reset the baud rate to low level through the AT command.

The baud rate reset by the AT command can be kept for the next time even though the power is cut off.

3. Reset the Bluetooth name

Send: AT+NAMEname

Back: OKname

Parameter name: Name needed to be set (20 characters limited) Example:

Send: AT+NAMEbill_gates

Back: OKname

Now, the Bluetooth name is reset to be “bill_gates”

The parameter can be kept even though the power is cut off. User can see the new Bluetooth name in PDA refresh service. (Note: The name is limited in 20 characters.)

4. change the Bluetooth pair password

Send: AT+PINxxxx

Back:OKsetpin

Parameter xxxx: The pair password needed to be set, is a 4-bits number. This command can be used in the master and slave module. At some occasions, the master module may be asked to enter the password when the master module tries to connect the slave module (adapter or cell-phone). Only if the password is entered, the successful connection can be built. At the other occasions, the pair can be finish automatically if the master module can search the proper slave module and the password is correct. Besides the paired slave module, the master can connect the other devices who have slave module, such as Bluetooth digital camera, Bluetooth GPS, Bluetooth serial printer etc.

Example:

Send: AT+PIN8888

Back: OKsetpin

Then the password is changed to be 8888, while the default is 1234. This parameter can be kept even though the power is cut off.

5. No parity check (The version, higher than V1.5, can use this command)

Send: AT+PN (This is the default value)

Back: OK NONE

6. Set odd parity check (The version, higher than V1.5, can use this command)

Send: AT+PO

Back: OK ODD

7. Set even parity check(The version, higher than V1.5, can use this command)

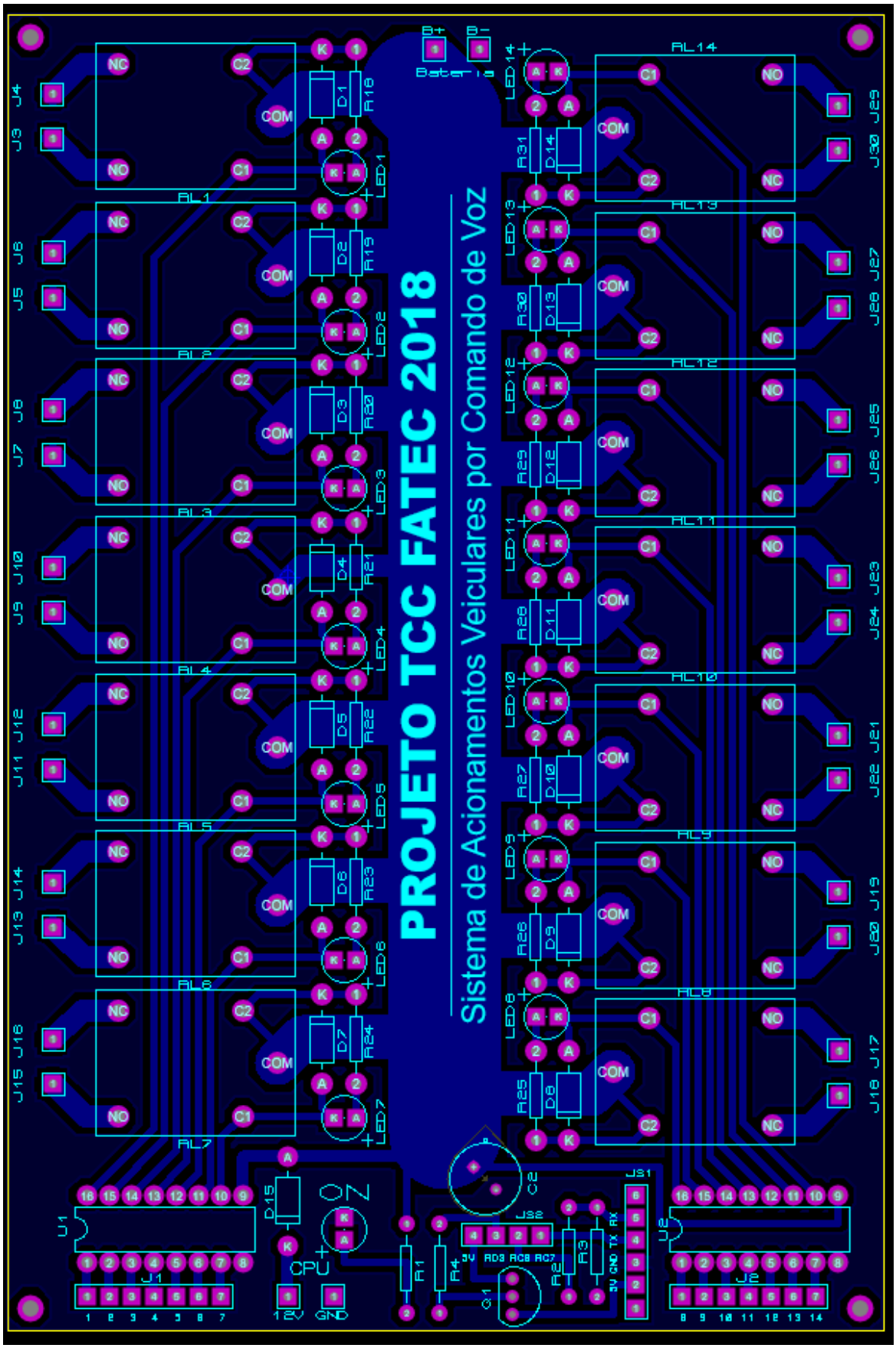
Send: AT+PE

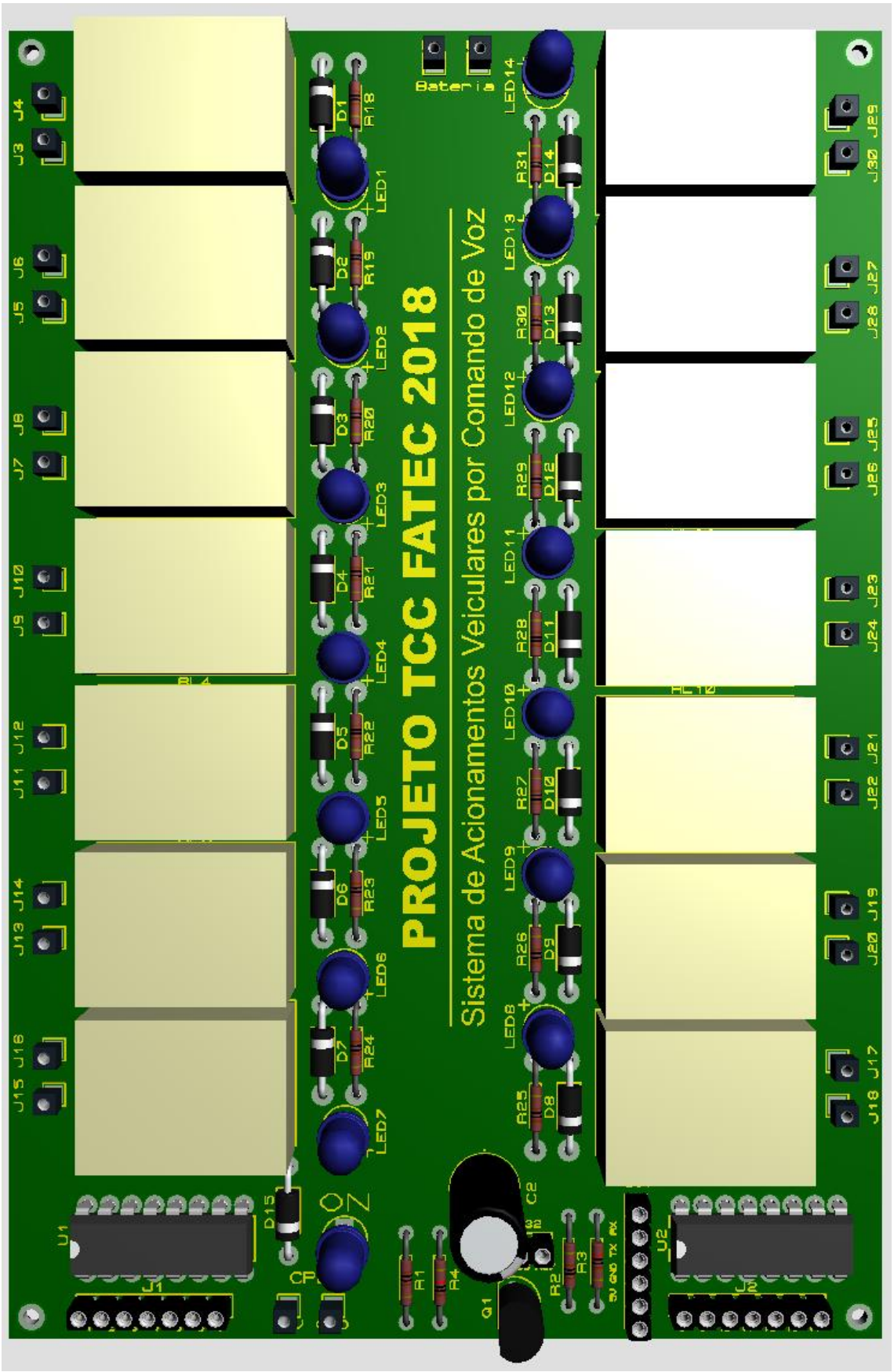
Back: OK EVEN

8. Get the AT version Send: AT+VERSION

Back: LinvorV1.n

ANEXO C – LAYOUT PLACA DE INTERFACE PROTÓTIPO




PROJETO TCC FATEC 2018
Sistema de Acionamentos Veiculares por Comando de Voz
Bateria