INSTITUTO FEDERAL DA PARAÍBA
CURSO TÉCNICO EM ELETRÔNICA INTEGRADO AO ENSINO MÉDIO

Denis Ivinis Sousa Gomes
Jean Carlos Silva Ribeiro

SISTEMA DE ALARME VEICULAR CONTROLADO POR WI-FI

João Pessoa-PB
2019

Denis Ivinis Sousa Gomes

Jean Carlos Silva Ribeiro

Sistema de Alarme Veicular Controlado por Wi-Fi

Trabalho de conclusão de curso técnico integrado ao médio em Eletrônica apresentado ao Instituto Federal da Paraíba como requisito parcial para a obtenção do título de Técnico em Eletrônica.

Orientador: Prof. Dr. Adaildo G D'Assunção Junior

João Pessoa-PB

2019

Denis Ivinis Sousa Gomes

Jean Carlos Silva Ribeiro

Sistema de Alarme Veicular controlado por Wi-Fi

Trabalho de conclusão de curso Técnico Integrado ao Médio apresentado ao Instituto Federal da Paraíba requisito parcial para a obtenção do título de Técnico em Eletrônica.

Aprovado em: 27 de março de 2019

BANCA EXAMINADORA

_____

Lincoln Machado De Araújo, Prof. Dr. – IFPB
(Avaliador)

_____

Patric Lacouth Da Silva, Prof. Dr. - IFPB
(Avaliador)

_____

Adaildo Gomes D'Assunção Jr, Prof. Dr. - IFPB
(Orientador)

Dedicamos este trabalho a todos aqueles que nos apoiaram ao longo dessa jornada de aprendizagem, principalmente nossos pais e professores.

AGRADECIMENTOS

Agradecemos primeiramente as nossas famílias por todo apoio, incentivo e suporte, características essenciais no decorrer na nossa vida e graduação.

Aos nossos professores por todos os conhecimentos passados.

Ao Prof. Adaildo Assunção Jr pela orientação, suporte e oportunidade no decorrer do desenvolvimento desse trabalho.

Aos nossos colegas de classe pela convivência e conhecimentos compartilhados ao longo dos anos de graduação.

A todos que direta ou indiretamente fizeram parte de nossa formação, o nosso muito obrigado.

“A persistência é o menor caminho do êxito”.

(Charles Chaplin)

RESUMO

Neste trabalho é apresentado o desenvolvimento e aplicação de um sistema de alarme que permite maior controle e segurança aos condutores e donos de veículos.
Componentes utilizados: Placa de desenvolvimento WeMosD1, com módulo ESP8266 e um *smartphone*. Esse projeto tem como finalidade proporcionar maior interação entre o condutor e seu veículo além de fornecer maior segurança em suas devidas funções. Este dispositivo busca disponibilizar o controle da injeção de combustível, sistema de alarme, controle de travas e faróis. Todo esse aparato de funções controladas remotamente, através de um *smartphone* Android.

Palavras-chave: Android. Alarme veicular. Alarme Wi-Fi. ESP8266. WeMosD1.

ABSTRACT

This work presents the development and application of an alarm system that allows greater control and a sense of safety to drivers and vehicle owners. The work created used: WeMosD1 development board, with ESP8266 module and a smartphone. This project was created with the purpose of providing greater interaction between the driver and his vehicle, in addition to providing greater security in his duties. This device seeks to provide control of the fuel injection, visual signal from where the vehicle is parked, alarm system, control of locks and headlights. All this apparatus of functions controlled remotely, through an Android *smartphone*.

Keywords: Android. Wi-fi alarm. Vehicular Alarm. WeMosD1. ESP8266.

Lista de Siglas e Abreviaturas

TCC Trabalho de Conclusão de Curso.

APP Aplicativo.

VCC Tensão de Corrente Continua.

IDE "*Integrated Development Environment*" – Ambiente de Desenvolvimento Integrado.

WI-FI *"Wireless Fidelity".*

GND "*Graduated Neutral Density Filter*" – Filtro de Densidade Neutra Graduada.

USB "*Universal Serial Bus*" – Barramento Serial Universal.

PWM "*Pulse Width Modulation*" – Modulação de Largura de Pulso.

CPU "*Central Processing Unit*" – Unidade central de processamento.

IOT "*Internet Of Things*" – Internet das Coisas.

M2M "*Machine-to-Machine*" – Máquina a Máquina.

I/O "*In Out*" – Entrada ou Saída.

GPIO "*General Purpose Input/Output*" – Entrada / Saída para Uso Geral.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

A procura por veículos eficientes aumenta dia após dia, essa demanda faz com que as montadoras produzam veículos mais sofisticados e tecnológicos, com os avanços em carros autônomos, sendo o primeiro da General Motors (GM), no ano de 1956, o Firebird II aos modelos de carro *Model S, Model X e Model 3* da Tesla. As montadoras buscam melhorar o conforto dos condutores e passageiros, já existem veículos com sistemas que se conectam a dispositivos móveis permitindo atender chamadas, e controlar o rádio e outras funções.

A conectividade faz parte do dia a dia da população, atualmente a conexão não se restringe a conectar pessoas a pessoas em todo mundo. Para Evans, D. (2011) a Internet das Coisas surgiu quando o número de coisas e objetos conectados à internet ultrapassou a quantidade de pessoas conectadas. Hoje é possível a conexão de pessoas com máquinas, como também, de máquina com máquina, tal tecnologia é conhecida como M2M que se refere à comunicação de máquina para máquina, fornecendo conectividade para todos e tudo. (Peter Waher, 2015).

Atualmente, existe uma tendência de utilização de dispositivos móveis para controlar outros objetos, tendência essa que já está sendo muito utilizado na indústria. Na domótica, por exemplo, é possível acessar o controle de lâmpadas, televisores e até um sistema de segurança via rede Wi-Fi. Tal tecnologia tem o potencial de causar um grande impacto no cotidiano dos usuários. Desde monitoramento remoto de pacientes, *e-health*, possibilidade de criar ambientes inteligentes, até tornar uma residência mais segura e também economizar energia, como mostrado em Pinto (2010).

Essa tecnologia está migrando para os veículos, pois o acumulo de bens com alto custo monetário (transportadoras), e a necessidade de locomoção mais rápida. Com isso, é possível permitir uma maior interação com os veículos, identificando a localização, e controlando funções em um veículo por um *smartphone* local ou remotamente.

De acordo com um estudo, em 2015, a produção anual de veículos foi de 2,499 milhões, enquanto 2016 foram produzidos 2,176 milhões. Se compararmos a produção do mês de dezembro de 2017 com a de dezembro de 2018, houve queda de 16,8%. No último mês de 2018, a produção de veículos atingiu 177,7 mil unidades. Já em 2017 foi de 213,7 mil.

De acordo com dados do Portal G1 (http://g1.globo.com/brasil/noticia/2014/03/com-aumento-da-frota-pais-tem-1-automovel-para-cada-4-habitantes.html, 2019) o crescimento da circulação carros no Brasil, sendo um automóvel para cada 4,4 habitantes. O investimento tecnológico para proteção desses bens é algo frequente nos dias atuais. No Brasil, segundo matéria do Bom dia Brasil, da rede Globo exibida no dia 01/10/2015, a cada 3 minutos um carro é roubado, sendo mais de 210 mil por ano. O que leva a uma grande demanda na busca de meios de segurança no mercado, os alarmes bloqueadores são uma boa pedida por ter um baixo custo e alta facilidade de manuseamento.

Um alarme bloqueador é um conjunto de sensores unidos a algum tipo de sirene, com uma função a mais de poder interromper a injeção de combustível no veículo, impedindo à partida antes que o alarme seja desativado. Assim, um sistema de alarme é definido como sendo todo sistema para a geração e visualização de alarmes, incluindo equipamentos como *hardware* e *software*, equipamentos de campo, transmissão de sinais, processamento de alarme e visualização. (LEITÃO, 2008). O alarme mais simples seria um interruptor na porta do motorista e seria conectado a uma sirene, e quando a porta fosse aberta, a sirene começa a tocar.

O projeto tem como proposta desenvolver um sistema de alarme veicular utilizando uma plataforma WEMOS D1 com acesso via *smartphone*, que gerencia o acionamento do alarme, controla a sirene, tal como a abertura e fechamento das portas e movimento interno do veículo com um sensor ultrassônico.

O alarme desenvolvido é uma união dessa necessidade de segurança e de aumento da conectividade como o veículo, permitindo até mesmo acender os faróis e sinaleira como ferramenta de localização do veículo num estacionamento, que já existe em carros mais sofisticados (Mini Cooper, por exemplo).

O sistema desenvolvido é controlado por um dispositivo móvel com sistema operacional Android, que utiliza de um *smartphone* como um controle remoto.

2. COMPONENTES E MATERIAS

Umas das preocupações iniciais para o desenvolvimento do alarme foram às condições que um veículo convencional enfrenta durante o dia a dia, com a alta temperatura podendo causar superaquecimento dos circuitos ou desconectar as ligações devido às trepidações do veículo. Então, foram pesquisados componentes resistentes a trepidações e ao calor interno do veículo.

Para esse projeto foram utilizados diversos componentes (Tabela 1), entre eles, uma placa de desenvolvimento WEMOS D1 como controlador do sistema, um sensor ultrassônico (Modulo HC-05) para detecção de movimento interno e relés para acionamento de cargas do veículo.

Tabela 1: Componentes de *Hardware* e custos

Componentes	Quantidade	Valor R$
WeMos D1 R2	1	R$ 40,00
Transistor BD139	4	R$ 4,00
Relé	4	R$ 11,00
Sensor ultrassônico HC05	1	R$ 12,00
Carregador veicular	1	R$ 20,00
Fios Jumper	20	R$ 2,00
Diodo 1N4148	4	R$ 0,60
Resistor 1KΩ	4	R$ 0,40
	Total:	R$ 90,00

Fonte: Elaborado por autores

2.1. PLACA DE DESENVOLVIMENTO WEMOSD1

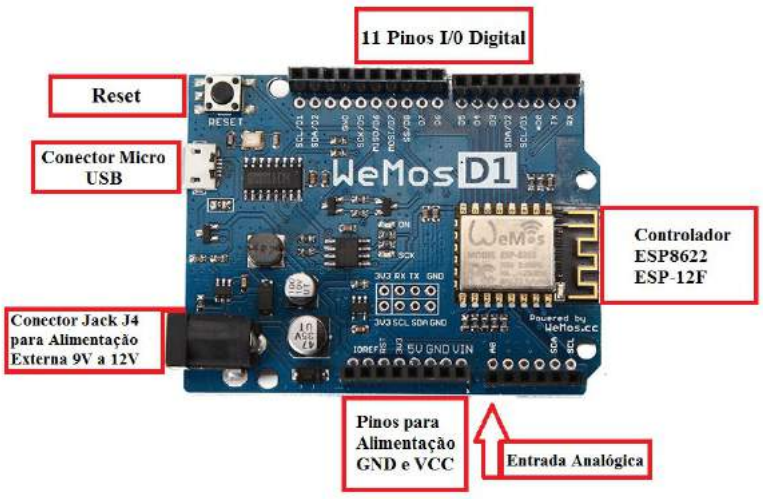
A placa de desenvolvimento WeMosD1 R2 (https://www.wemos.cc/ – site do fabricante) é uma placa de código aberto, controlada pelo módulo ESP8266EX, proporciona conectividade Wi-Fi nativa pelo modulo ESP12F, contém um conector micro USB que permite a ligação com um computador para programação ou alimentação, além de diversos pinos que permitem a conexão com circuitos eletrônicos externos, motores, relés, sensores luminosos, diodos laser, alto-falantes, microfones e etc.

A placa WeMosD1 pode ser alimentada através do conector micro USB a partir de um computador ou de bateria 9V a 12V. Ela pode ser programada através da entrada micro USB ou via WIFI. Podendo ser controlada diretamente por um computador, ou podem primeiro ser programados pelo computador e, a seguir, desconectados para trabalharem de forma autônoma.

A placa WeMosD1 é uma placa robusta e foi desenvolvida para aplicações em campo, com uma variedade grande de recursos ela permite projetar ferramentas de baixo custo, sendo uma ótima plataforma para criação de equipamentos utilizando linguagem C++ para programação. Para desenvolvimento do código fonte, utilizamos o software: "Arduino IDE (*Integrated Developmente Environment)".*

A WeMos D1 (figura 1), possui 11 pinos para entrada e saída de dados digitais, 1 para dados analógicos, 5 para alimentação (VCC e GND), 1 reset e outros 2 vin.

Figura 3: Pinos WeMos D1 R2

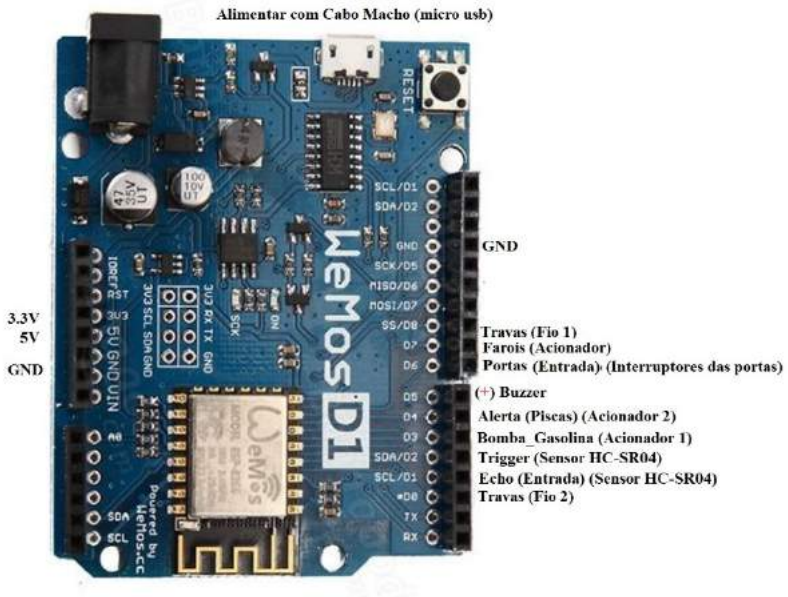


Fonte: Adaptado por autores.

A porta A0 (analógica) pode ser usada como entrada/saída PWM de 8 bits através da função analogWrite(). Os pinos Digitais D1 ao D5 são pinos I/0 (*In Out*), ou seja, são pinos de saída (acionamentos) ou entradas (leituras) digitais. O conector Micro USB é utilizado para comunicação e/ou alimentação com o computador. A alimentação externa é utilizada para alimentar a placa, por uma bateria ou fonte. Quando a placa é alimentada através do conector tipo Jack J4, a tensão da fonte estará no pino VIN. Os pinos para alimentação são pinos para alimentação de circuitos ou *shields*. O botão conectado a pino de RESET do controlador pode ser utilizado para um reset externo da placa WeMos.

A alimentação pode ser realizada através do conector micro USB com tensão 5V ou do conector Jack (positivo no centro), onde a tensão da fonte externa deve estar entre 9V a 12V, porém, se alimentada com uma tensão abaixo de 7V, a tensão de funcionamento da placa, que no WeMos é 3.3V, pode ficar instável e quando alimentada com tensão acima de 12V, o regulador de tensão da placa pode sobreaquecer e danificar a placa. Dessa forma, é recomendado para tensões de fonte externa valores de 9V a 12V (Rodrigues,2018).

Figura 4: Esquema de ligação para a Wemos D1 R2



Fonte: Adaptado por autores.

2.2 CONTROLADOR ESP8622EX

Este é um controlador de criação chinesa da Espressif (https://www.espressif.com/ – site do fabricante), que se tornou referência no desenvolvimento de plataforma micro controlada, que transmite e recebe dados via WiFi. A empresa Espressif criou uma gama de ESP para atender diversas necessidades, todas utilizando o controlador **ESP8266**, cada uma com sua particularidade, variando o tamanho, quantidade de pornas e podendo ter pinos para conexão ou não, sendo da ESP-01 a ESP-12, como observado na Figura 5.

Figura 5: Imagem dos controladores ESP's

Fonte: Adaptado por autores.

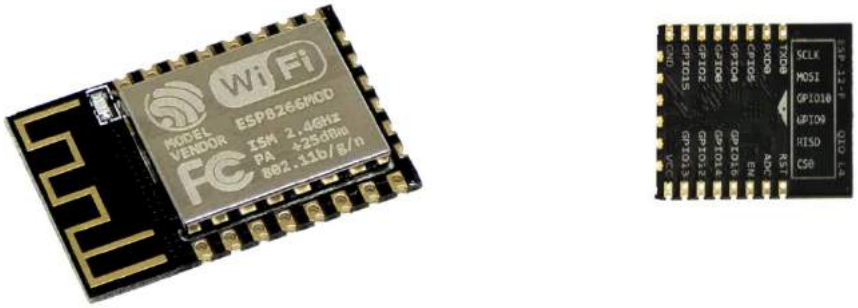
Embora visualmente os controladores sejam parecidos, existem grande diferenças entre eles, tanto em tamanho quanto a parte da especificação técnica. A placa escolhida, Wemos D1, conta com o controlador ESP-12F, sendo o mais robusto em poder de processamento e transferência de dados em relação aos outros controladores.

A placas de desenvolvimento ESP, ver Figura 6, são dispositivos de código aberto, que possui processador interno bastante poderoso, sendo capaz de processar sensores e aplicações em seus GPIO (*General Purpose Input/Output*), com o objetivo auxiliar projetos de IoT, sua

programação sendo feita através no Arduino IDE, torna de fácil programação e entendimento, além de, proporcionar conexão WiFi nativa.

Figura 6: Módulo ESP8622 ESP-12F. (a) Vista frontal. (b) Vista traseira

(a) (b)



Fonte: https://www.eletrogate.com/modulo-wifi-esp8266-esp-12f-nova-versao

Seu processador integrado de 32 bits de baixo consumo e sua memoria de 4 Mbyte, tem o poder de executar programas diretamente no módulo, dispensando uso de muitos componentes externos.

Seus pinos de digitais GPIO são ligados aos pinos digitas da WeMos D1, assim como o VCC e GND, sua programação é feita através dos pinos RXD0 e TXD0. A tensão de operação dos pinos digitais do módulo ESP-12F trabalham em cerca de 3.3V, a maioria dos *Shields* de Arduino trabalham com 5V, o que pode danificar as saídas do módulo, caso sejam conectadas diretamente.

Outras caracteristicas são relacionadas abaixo

- Padrão: 802.11 B/G/N
- Wi-Fi Direct (P2P), Soft-AP
- Stack TCP/IP integrada
- SDIO 1.1/2.0, SPI, UART
- Modos: Estação / Ponto de Acesso
- Segurança: WPA, WPA2
- ADC 10 bits

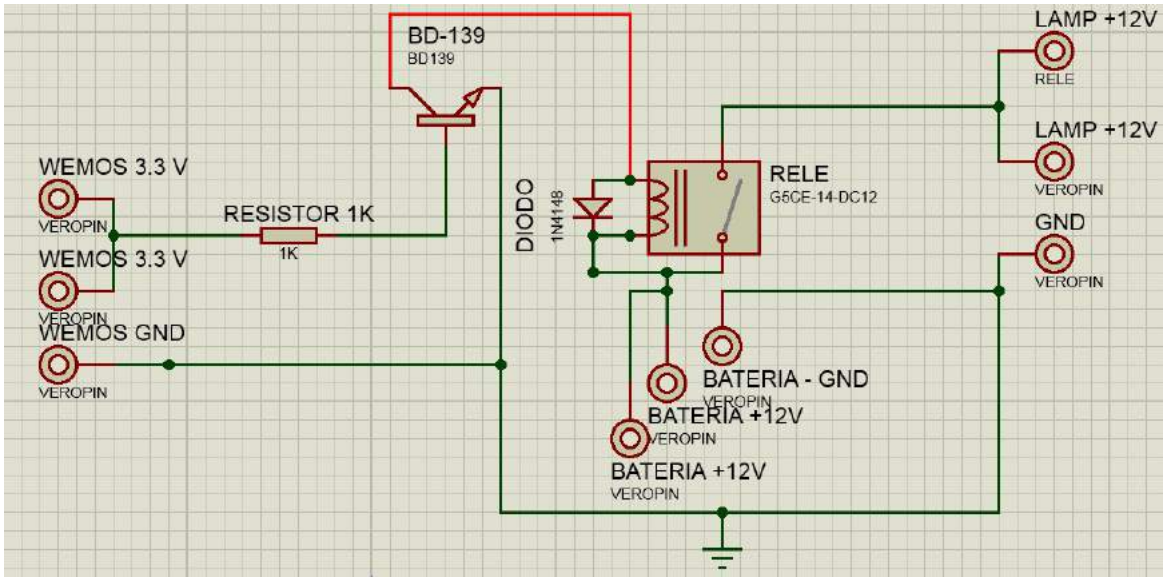
- Memória Flash: 4Mbyte
- Tensão (I/O): 3.3v

2.3 RELÉ

Para o projeto do alarme foi necessário utilizar relés para acionamento de cargas com tensão de acionamento 12V. Também é importante não conectar o relé diretamente na saída digital da WeMos D1, pois pode danificar o módulo, para solucionar o problema foi projetado um circuito acionador de relé.

Após identificar a dificuldade de acionamento dos relés pelas saídas digitais da WeMosD1; desenvolveu-se um circuito de acionamento do relé com a tensão de 3,3 V, o circuito de chaveamento desenvolvido (Figura 8), já que o veiculo conta com uma bateria de 12V.

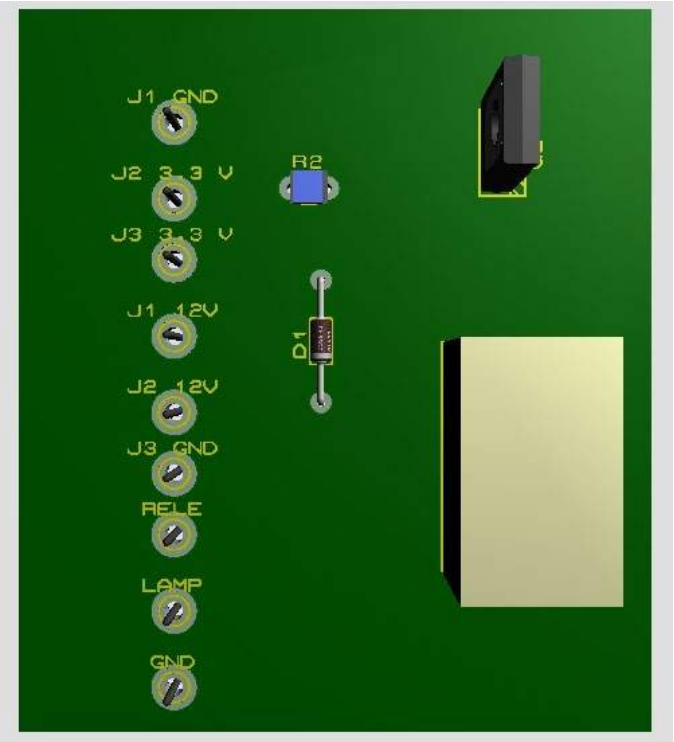
Figura 7: Circuito elétrico do acionador de relé desenvolvido no Proteus



Fonte: Elaborado pelos autores.

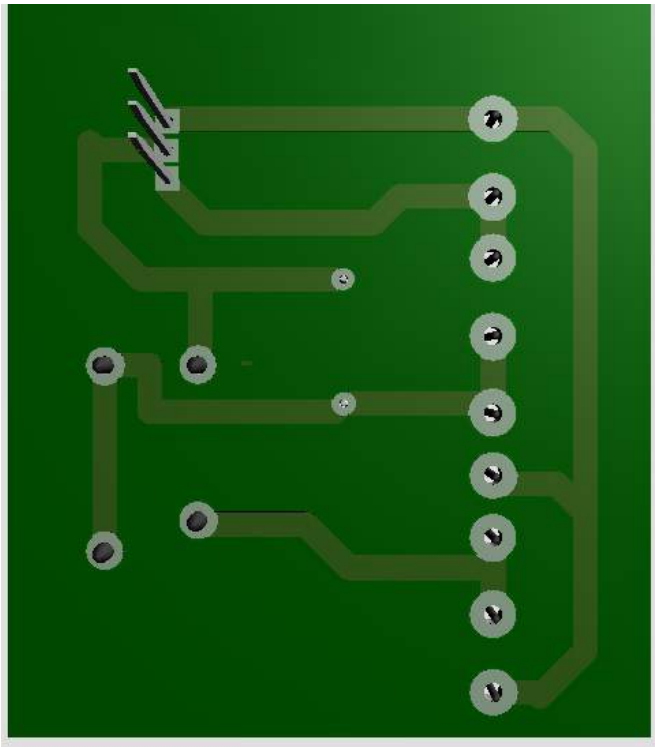
Assim, após o transistor BD139 receber na base o comando de 3.3V da saída digital da WeMosD1, logo o transistor fica em regime de saturação e aciona o relé, dando início a algum comando no alarme do veiculo.

Figura 8: Placa projetada do circuito em 3D do acionador de relé desenvolvida no Proteus



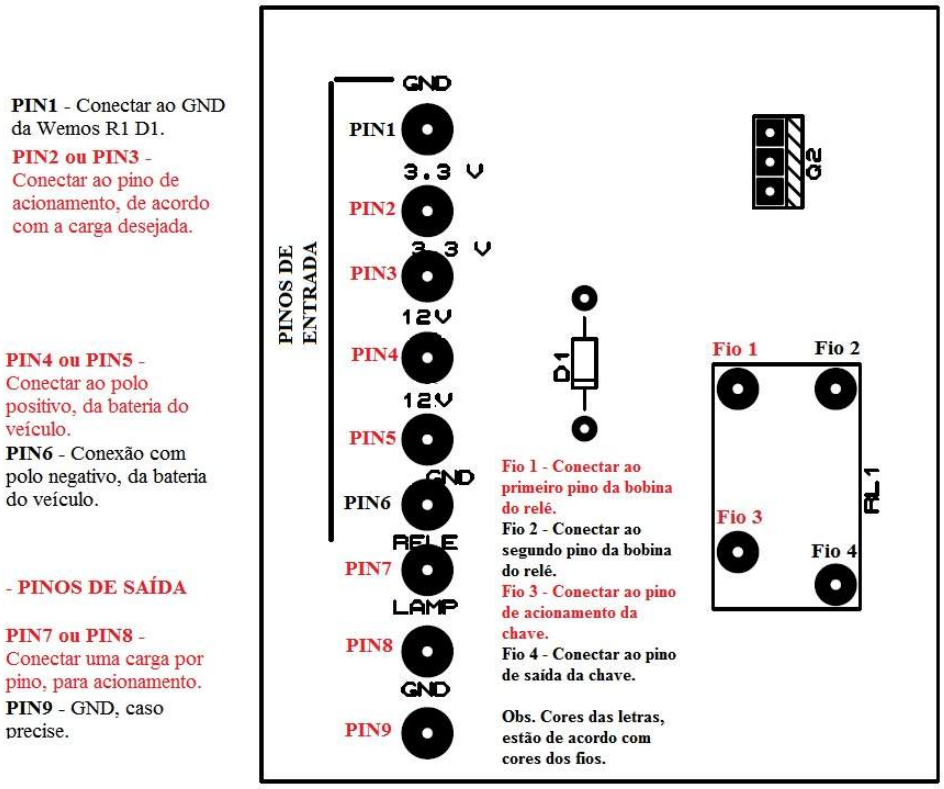
Fonte: Elaborado pelos autores.

Figura 9: Placa projetada do circuito em 3D, trilhas do acionador de relé desenvolvida no Proteus



Fonte: Elaborado pelos autores.

Figura 10: Esquema de ligação para o acionador de relé



Fonte: Elaborado pelos autores.

2.4 SENSOR ULTRASSÔNICO

No sistema foi utilizado um dispositivo como sensor de presença, assim como já é encontrado em modelos de carros. O HC-SR04 é muito utilizado em projetos com Arduino, com uma ótima precisão de 3mm. Esse dispositivo consegue executar leituras entre 2 cm e 4 metros, para isso utiliza um emissor e um leitor ultrassônico, emitindo 8 pulsos de 40 kHz e através da ressonância utilizando o cálculo:

$$Vsom = 340{,}29\ \frac{m}{s}$$

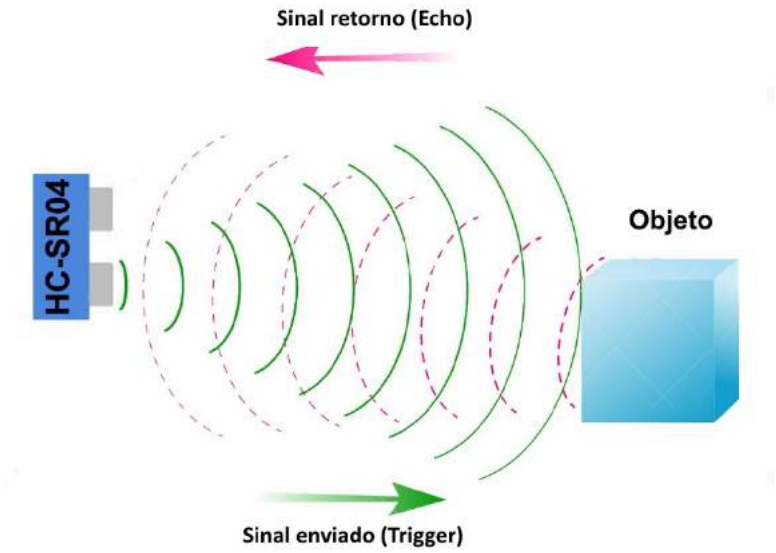
$$Vsom = \frac{\Delta S}{\Delta T} = 340{,}29\ \frac{m}{s}$$

$$\Delta T = T/2$$

$$\frac{\Delta S}{T/2} = 340{,}29$$

$$\boldsymbol{\Delta S = \frac{340{,}29 * T}{2}}$$

Figura 11: Método de Ressonância



Fonte: https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/

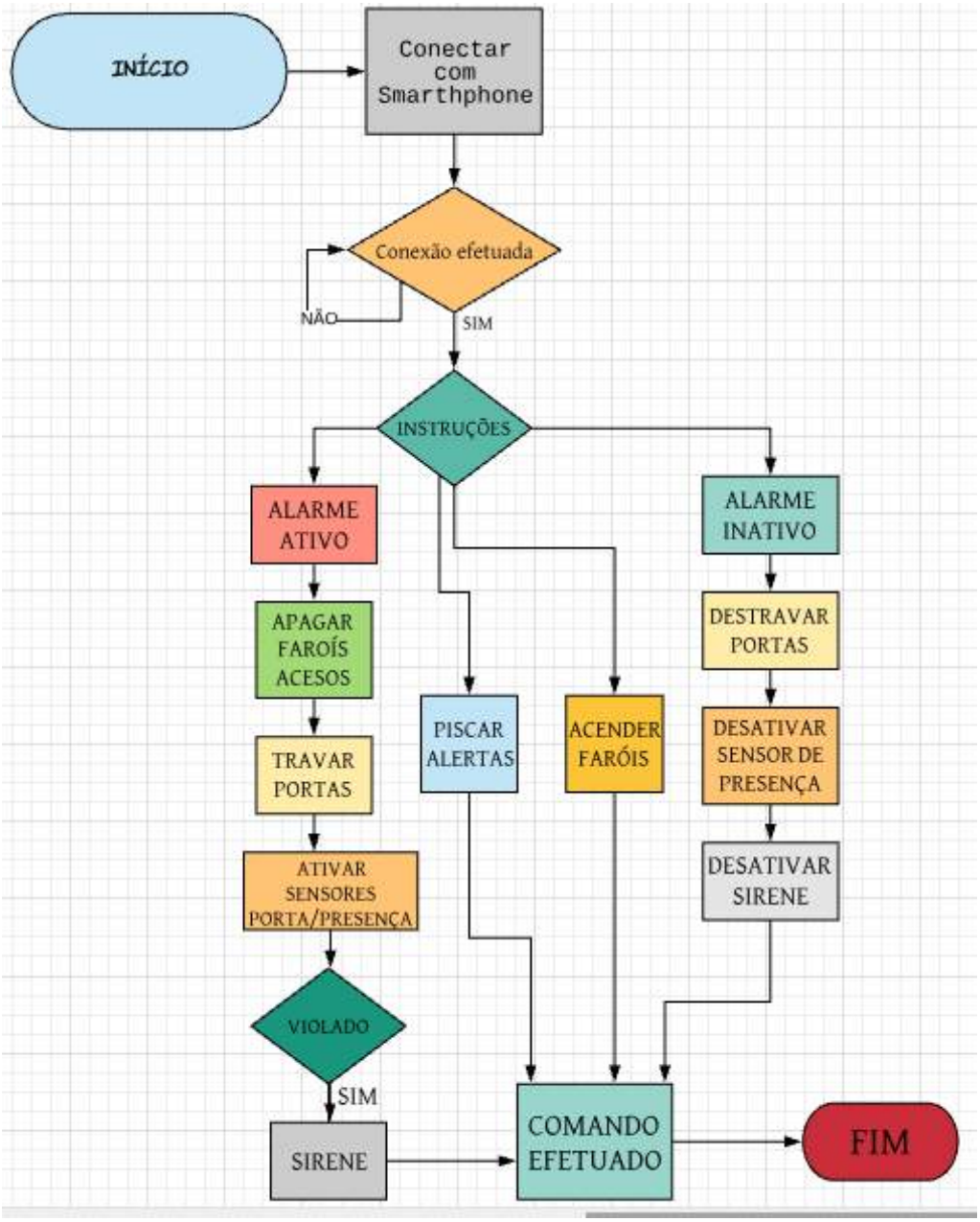
O sensor possui apenas 4 pinos, sendo eles: VCC, GND, Trigger e Echo.

- VCC: alimentação positiva do arduino, 5V.
- GND: conexão com o terra da placa.
- Trigger: emissor dos pulsos.
- Echo: leitor da sinal de eco (retorno).

3. SOFTWARE DESENVOLVIDOS

Para a parte lógica do alarme, foram utilizados duas ferramentas para a construção do projeto, a IDE do Arduino para a programação da placa de desenvolvimento WeMosD1 e pagina Web, e a ferramenta App Inventor para o desenvolvimento do aplicativo para o smartphone. Na figura 11, é apresentado o fluxograma de todos os *sofrwares* em conjunto.

Figura 12: Diagrama em blocos dos *softwares*



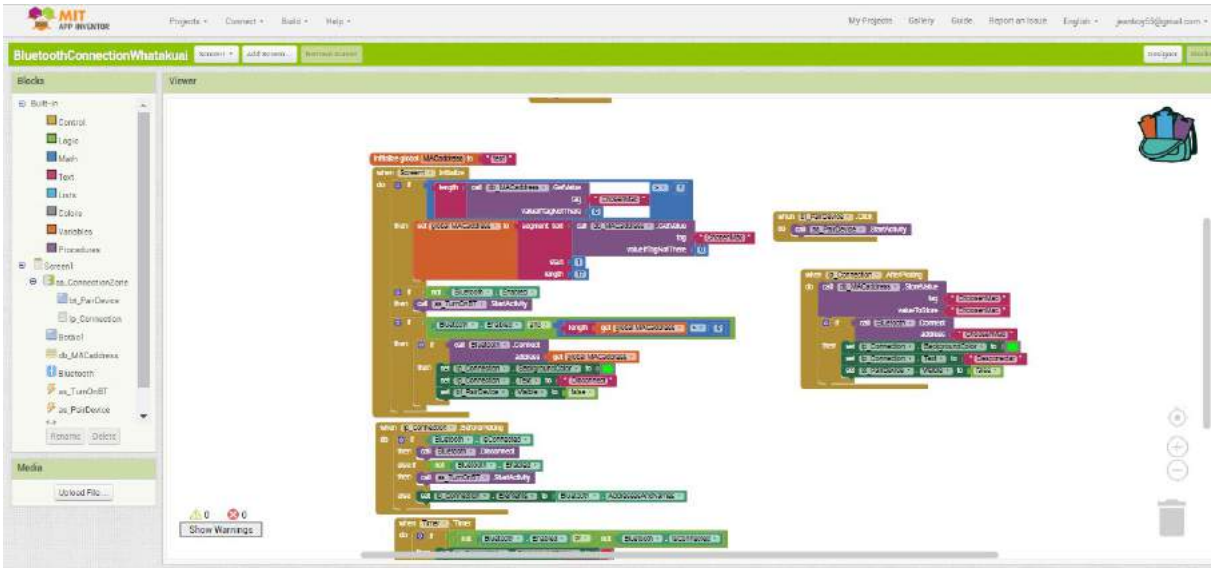
Fonte: Elaborado pelos autores.

3.1 APP INVENTOR

A ferramenta App Inventor (http://appinventor.mit.edu/explore/ – site do fabricante) foi desenvolvida pelo Google e nos dias de atuais, mantida pelo Instituto de Tecnologia de Massachusetts (MIT). Possui uma interface simples que permite a criação de aplicativos para *smartphones* que possuem o sistema operacional Android, sem que seja necessário conhecimento nas linguagens de programação Kotlin, Java ou C++.

Essa ferramenta pode ser utilizada através de um navegador de internet, como mostrado na Figura 12. O projeto pode ser acessado através do e-mail, ou através da geração de um arquivo (.aia), que permite total a acesso ao aplicativo. Após a programação, podendo gerado um arquivo tipo .apk para ser baixado e instalado no *smartphone*.

Figura 13: Programação em blocos



Fonte: Elaborado pelos autores.

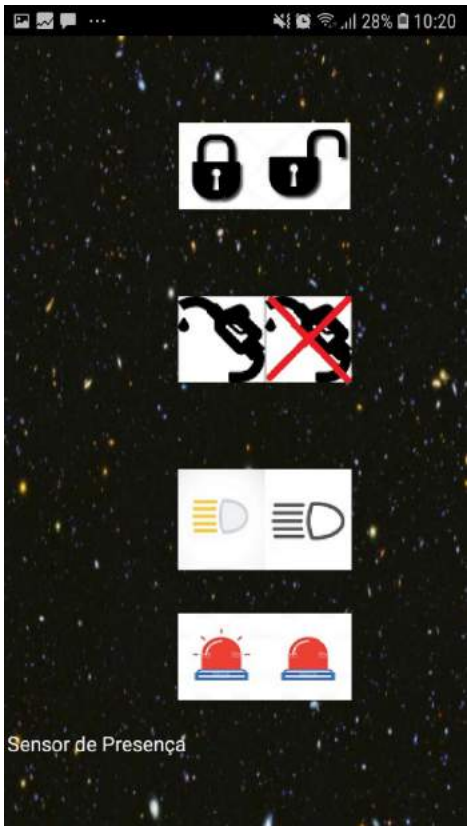
Também é oferecido um emulador, que pode diminuir o tempo de criação do aplicativo, pois através desse emulador não precisará baixar o aplicativo diversas vezes para testes e modificações.

A programação dessa ferramenta ocorre através de um sistema de blocagem, utilizando a união dos blocos para desenvolvimento do aplicativo desejado, ele oferece um grande leque de escolhas, possibilitando a criação de diversos tipos de aplicativos, que vão de jogos, até projetos mais simples, como uma galeria de fotos.

Esta forma de programação ajuda no entendimento do código, sendo mais simples de trabalhar, servindo para pessoas mais qualificadas com uma grande bagagem no ramo de programação, mas também para pessoas que não possuem tanto conhecimento e expertise em programação.

O primeiro passo é a inclusão e organização do que se vai precisar no APP, em seguida, é criada a interface, ou melhor, o *layout* do programa com suas funcionalidades como botão para travar e destravar, para interromper a alimentação da injeção de combustível, para ligar e desligar faróis, e sinal sonoro, como mostra a Figura 11.

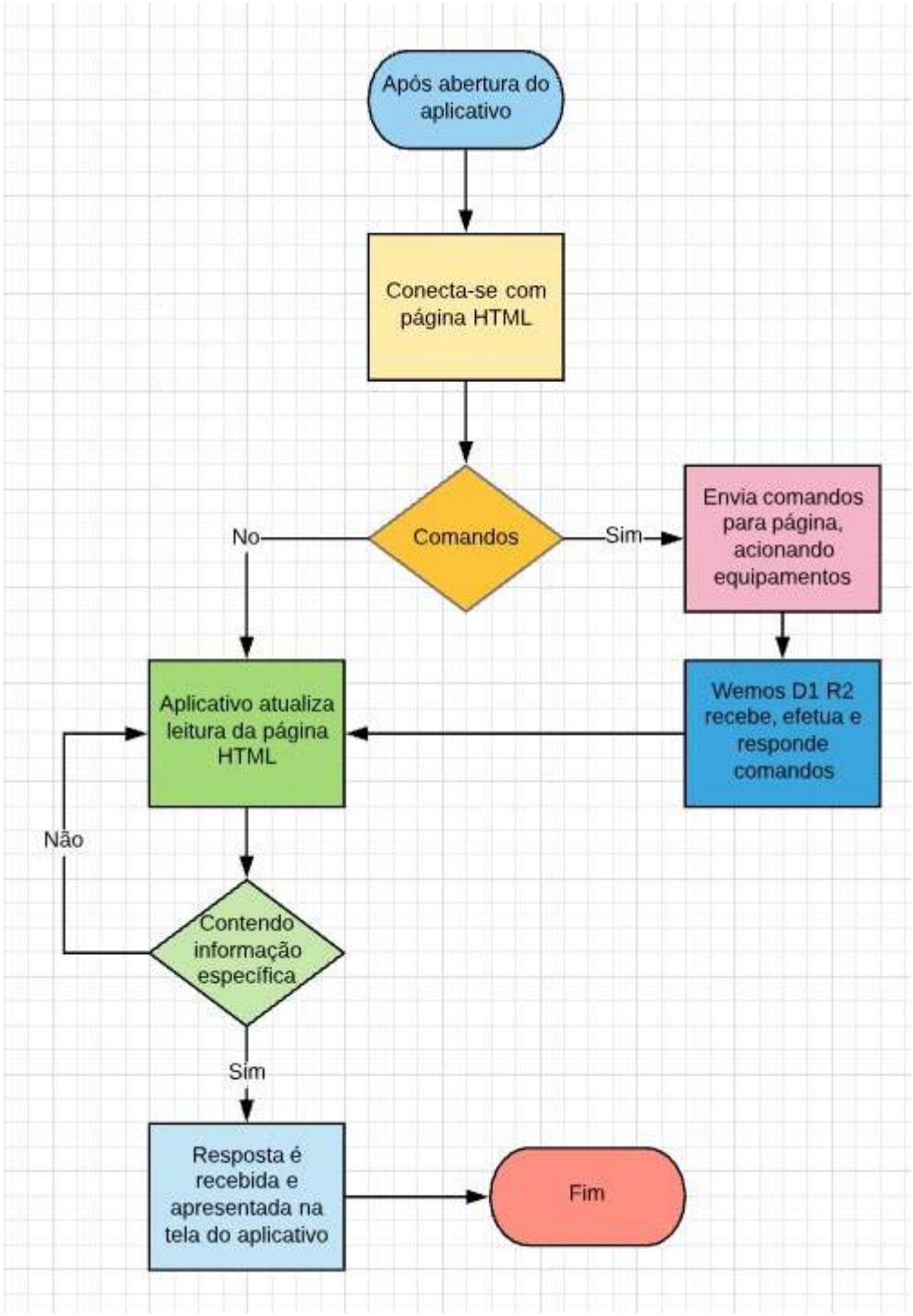
Figura 14 – Designer do aplicativo



Fontes: Elaborado pelos autores.

Por último, será efetuado a programação dos blocos, que trata-se do conjunto de comando e instruções de elementos do APP, sua logica é está apresentada na Figura 12. Após programação, só é preciso transferir o arquivo, extensão tipo .apk para seu smartphone com sistema Android, essa transferência pode ser através da leitura de um QR-Code, transferência pela internet, ou via cabo USB.

Figura 15: Diagrama em blocos do *softwares* para *smatphone*



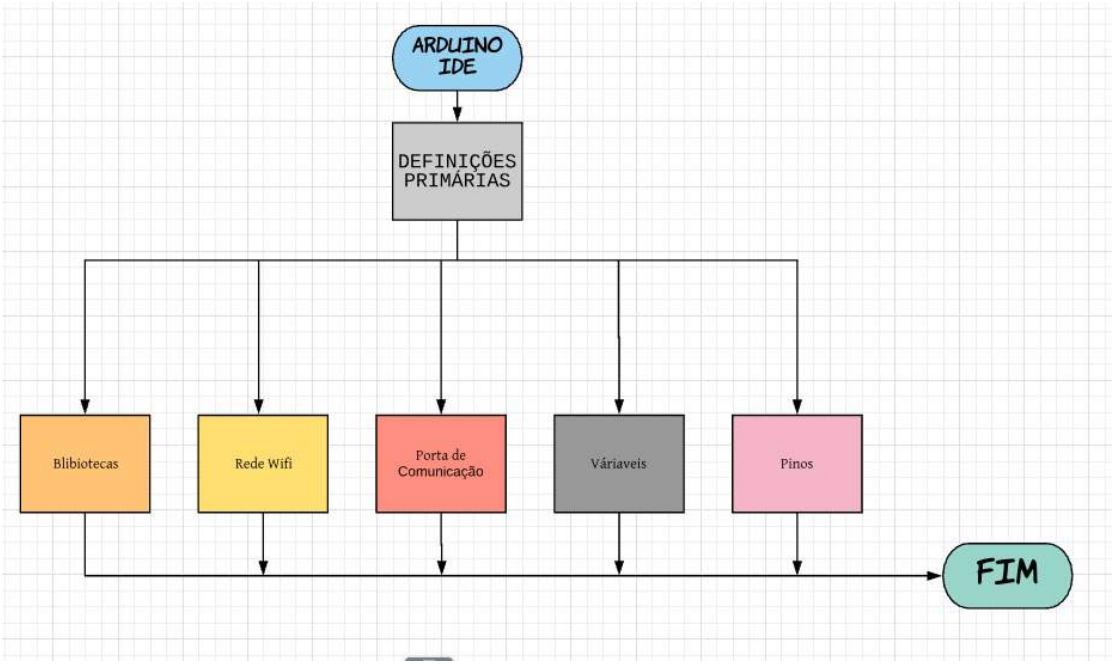
Fonte: Elaborado pelos autores.

3.2 PROGRAMAÇÕES NA IDE DO ARDUÍNO

A IDE do Arduino é um ambiente de desenvolvimento integrado, conta com um aplicativo programado em Java que dispõe de destaque de sintaxe, biblioteca e correções de erros, além de servir como canal para se enviar o código programado para a placa.

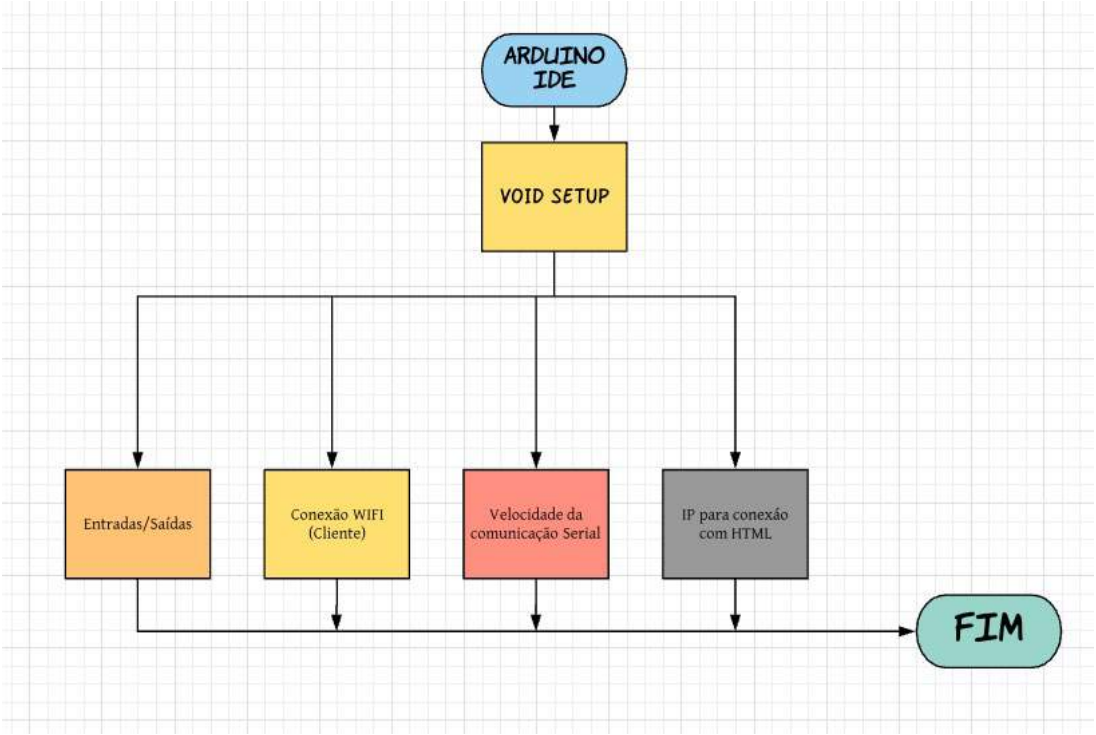
A implementação do software foi feita através da IDE do Arduino e desenvolvida na linguagem C++, sua logica está apresentada na figura 13.

Figura 16: Fluxograma do *software* para WeMosD1 (Parte1)



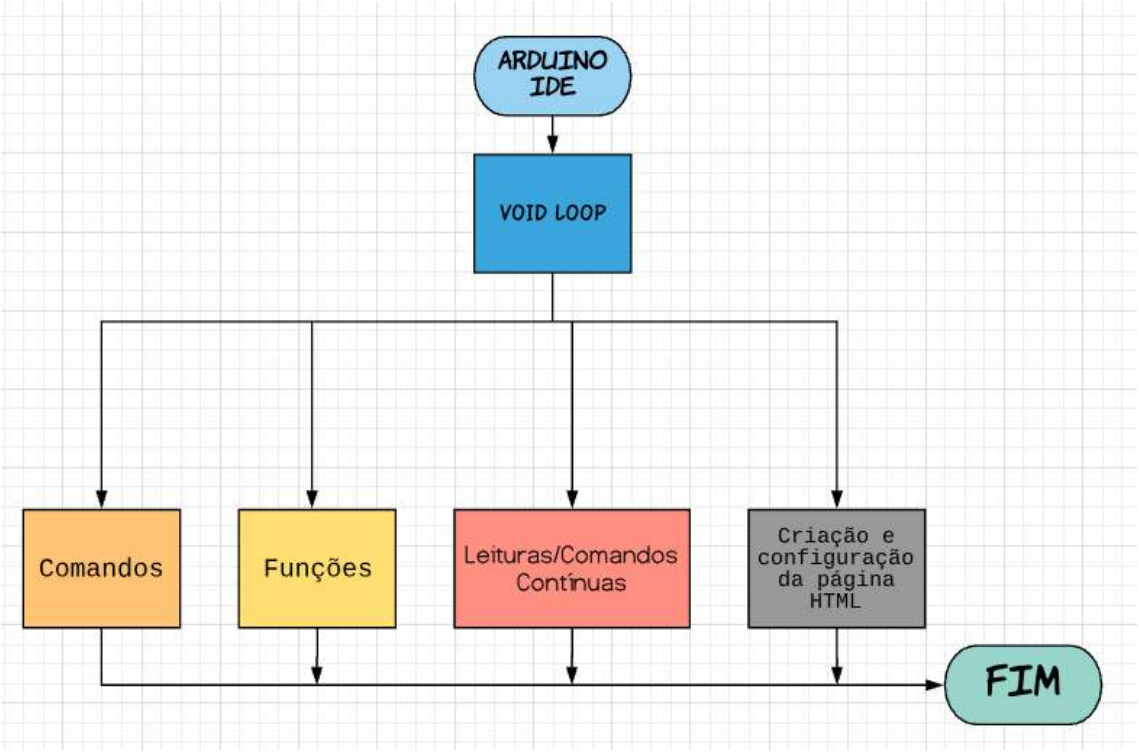
Fonte: Elaborado pelos autores.

Figura 17: Fluxograma do *software* para WeMosD1 (Parte2)



Fonte: Elaborado pelos autores.

Figura 18: Fluxograma do *software* para WeMosD1 (Parte3)



Fonte: Elaborado pelos autores.

4 TESTE E VALIDAÇÃO

Para a validação do *hardware* e *software*, primeiramente o equipamento foi testado na parte externa, em bancada, com fontes e com total atenção para seu comportamento, assim como para o circuito acionador de relé.

Em seguida, o alarme foi alimentado por um carregador veicular *Samsung* (https://www.samsung.com/br/ – Site do fabricante), ver figura 16, através de sua entrada micro USB e inserido em um veiculo tradicional, na parte abaixo do volante, onde se concentra grande parte da eletrônica do veiculo, assim tornando mais fácil sua instalação e sendo protegido de qualquer eventual problema, pelo sistema de vedação do veiculo.

Figura 19 – Foto Carregador Veicular *Samsung*

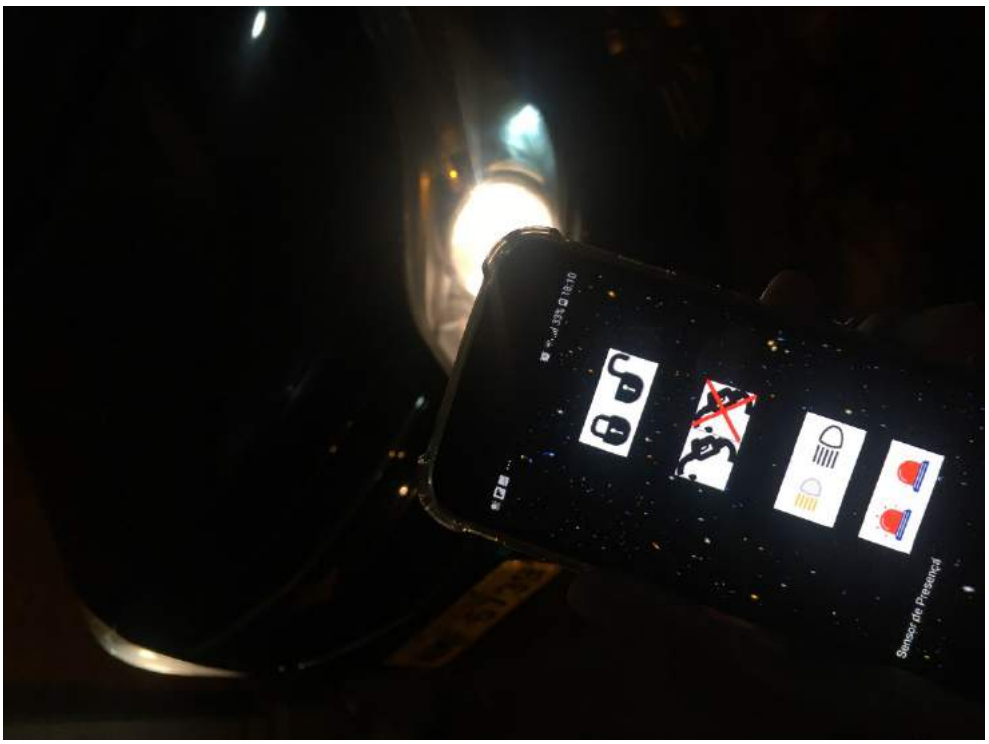
Fonte: https://www.americanas.com.br/produto/131546644/carregador-veicular-ultra-rapido-2-portas-usb-samsung

Observou-se que o equipamento recebia comandos de longas distâncias na prática, chegando a 65 metros de distância do veiculo, assim tornando-o seguro para acioná-lo após um eventual furto.

Pode-se constatar também, que os comandos foram bem recebidos e o sistema para acionar as funcionalidades no carro teve um bom desempenho, tendo um tempo de resposta rápido.

Para realizar os comandos no smartphone foi desenvolvido um aplicativo Android, na ferramenta App Inventor, com os seguintes botões apresentados na Fig.13, sendo isso o grande diferencial do nosso dispositivo, pois através de um aplicativo no *smarthphone* se envia comandos para o alarme no veiculo, ou seja, ele se torna um controle remoto com diversas opções para o usuário. Também é possível enviar comandos para localização do veículo, como o acionamento dos faróis e piscas do veículo, como observado na figura 17.

Figura 20: Imagem do aplicativo acionando os faróis do veiculo



Fonte: Elaborado pelos autores.

5 CONCLUSÕES E PERSPECTIVAS

Podemos perceber como foi desenvolvido e implementado um alarme de baixo custo e bastante versátil, projetado com o intuito de oferecer mais segurança para veículos particulares e comerciais. Utilizando plataformas intuitivas e comuns no mercado, foi possível desenvolver um dispositivo simples porém robusto e com um ótimo custo de aproximadamente R$90,00 representando uma economia de mais de 70% do valor de um alarme automotivo convencional.

Além disso, foi realizada a validação do funcionamento do protótipo. Assim como a validação da maneira de alimentação escolhida e melhor localização interna no veiculo, evitando calor excessivo ou trepidações.

Observou-se que é possível, utilizando esse equipamento, realizar comandos de longa distância para o veiculo realizar o alerta sonoro e visual, além de bloquear a alimentação da injeção eletrônica do veículo e travar as portas, sendo assim, impedindo a partida do veículo e atendendo o principal objetivo de um alarme convencional, impedir o furto do mesmo.

O próximo passo desse projeto é a complementação com um módulo GPS Duinopeak e GPRS SIM900. De forma que, qualquer pessoa através de seu smartphone, munido com o aplicativo do *Google Maps* localize seu veículo em qualquer situação. Além de que o aplicativo desenvolvido retorne informações de interesse, como distância total percorrida, tempo total em movimento e consumo de combustível, além de que em casos de acidentes possa enviar alertas para contatos pré-estabelecidos. Espera-se que em trabalhos futuros, haja o aproveitamento de todas as funcionalidades e opções do equipamento desenvolvido, tornando-o não só um alarme, mas um item indispensável para veículos tradicionais.

REFERÊNCIAS

BANGGOD. Wemos D1 R2 WiFi ESP8266 Placa de desenvolvimento compatível Ardino Uno programa por arduino IDE. Disponivel em :<https://www.banggood.com/pt/WeMos-D1-R2-WiFi-ESP8266-Development-Board-Compatible-Arduino-UNO-Program-By-Arduino-IDE-p-1011870.html?version=3&akmClientCountry=BR&cur_warehouse=CN> Acesso em: 21 de fevereiro de 2019

ELETROGATE. Módulo WiFi ESP8266 ESP-12F (NOVA VERSÃO). Disponível em: <https://www.eletrogate.com/modulo-wifi-esp8266-esp-12f-nova-versao> Acesso em: 21 de fevereiro de 2019

EVANS, Dave. The Internet of Things How the Next Evolution of the Internet Is Changing Everything. Cisco Internet Business Solutions Group (IBSG), 2011.

GROKHOTKOV, Ivan. ESP 8266 Arduino Core Documentation**.** Disponível em: <https://media.readthedocs.org/pdf/arduino-esp8266/docs_to_readthedocs/arduino-esp8266.pdf > Acesso em: 17 de fevereiro de 2019

LEITÃO, Gustavo Bezerra, Algoritmos para Análise de Alarmes em Processos Petroquímicos. 2008. Monografia de Graduação. UFRN, 2008.

MONK. S. Programming Arduino getting stard with sketehes [S.l]: McGraw Hill Professional. 2016.

PINTO, F. D. M. Desenvolvimento de um protótipo de um sistema domótico. Dissertação de Mestrado, Instituto Superior Técnico - Universidade Técnica de Lisboa, 2010. 21

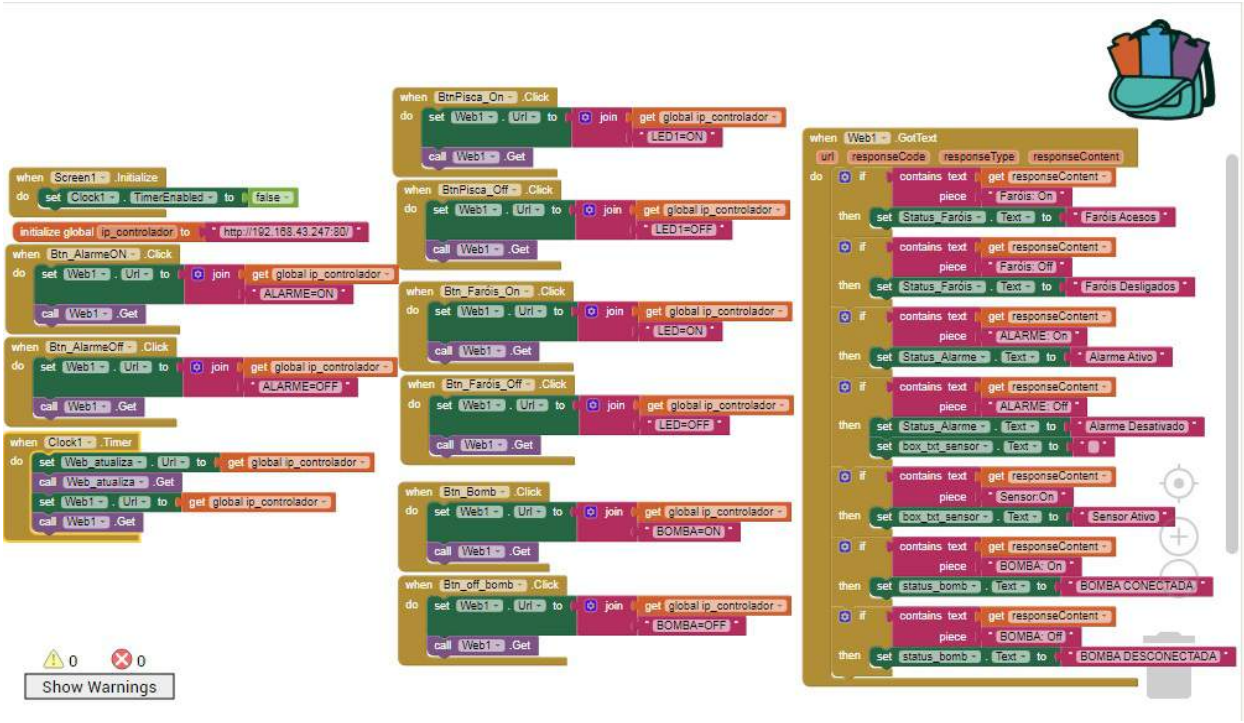
THOMSEN, Adilson. Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino**.** Disponível em: < https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/ > Acesso em: 20 de fevereiro de 2019

WAHER, PETER. Learning Internet of Things Paperback. Packet Publishing Ltd. Birmingham Mumbai, 2015.

WETMORE, J. M. Driving the Dream – The History and Motivations Behind 60 Years of Automated Highway Systems in America. Automotive History Review, p. 04-19, 2003.

APÊNDICE A – BLOCOS DO APLICATIVO DESENVOLVIDO NO APP INVENTOR

Figura 21: Programação em blocos, através do MIT 2



Fonte: Elaborado pelos autores.

APÊNDICE B – PROGRAMAÇÃO DESENVOLVIDA NO ARDUINO IDE

```
#include <ESP8266WiFi.h>
 #include <String.h>
 #include <Ultrasonic.h>

 //A cima bibliotecas específicas, para utilização do HC-SR04, ESP8266 e Strings.


 Ultrasonic ultrasonic(4, 5); //Definição de pinos do HC-SR04

 const char* ssid = "AndroidAP"; // Nome da Rede
 const char* password = "12345678"; //Senha da rede

 String readString=String(30); //Quantidades de Strings, trinta, para evitar travamento.

 // Abaixo, designação da pinagem especifica.
 const int pino_saida = 14;  //D5
 int ledPin = 13;  //D7
 int ledPin1 = 2;  //D4
 int trava_Pin1 = 16; //D0
 int trava_Pin2 = 15; //D8
 int PORTAS = 12; //D6

 WiFiServer server(80); // Porta de comunicação

 void setup() //
 {
  pinMode(trava_Pin1, OUTPUT);// Define como saída o pino D4
  pinMode(trava_Pin2, OUTPUT);// Define como saída o pino D8
  pinMode(PORTAS, OUTPUT);// Define como saída o pino D6
  pinMode(ECHO_PIN, INPUT); // Define o pino como entrada (RECEBE)
```

```
  pinMode(TRIGGER_PIN, OUTPUT); //Define o pino como saída (ENVIA)
  pinMode(pino_saida, OUTPUT);// Define como saída o pino D5
  pinMode(ledPin, OUTPUT); // Define como saída o pino D7
  pinMode(ledPin1, OUTPUT); // Define como saída o pino D4

 digitalWrite(ledPin1, LOW); // Os faróis começam desligados.

  Serial.begin(115200); // Velocidade do monitor serial
  delay(10);

 // A baixo comunicação com a rede WiFi

 Serial.println();
 Serial.println();
 Serial.print("Conectando à ");
 Serial.println(ssid);
 WiFi.begin(ssid, password); // Inicia a ligação a rede


 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
}
 Serial.println("");
 Serial.println("WiFi connected");

// Servidor
 server.begin(); // Comunicação com o servidor
 Serial.println("Servidor iniciado"); //É apresentado no monitor serial que o  servidor foi iniciado.

 Serial.print("Use o seguinte URL para a comunicação: ");// Mensagem apresentada no monitor. Impressão do endereço IP
 Serial.print("http://");
```

```
Serial.print(WiFi.localIP());
Serial.println("/");

 void Alarme_Ativo(){ // Armazenamento das definições.

  digitalWrite(PORTAS, HIGH); //Saída positiva
  digitalWrite(ledPin1, LOW);// Desliga faróis
  digitalWrite(trava_Pin1, LOW);//Primeiro pino da trava "NEGATIVO"
  digitalWrite(trava_Pin2, HIGH);//Segundo pino da trava "POSITIVO"

}
void Alarme_Desativado(){

  digitalWrite(pino_saida, LOW); //Sirene desligada.
  digitalWrite(trava_Pin1, HIGH);//Primeiro pino da trava "POSITIVO"
  digitalWrite(trava_Pin2, LOW);//Segundo pino da trava "NEGATIVO"

}

void loop() {

if (alarme_pir0 == 1){  // Se a variável receber 1, são lidos o comandos a baixo.
  int cmMsec;  //Definindo cmMsec como float
  long microsec = ultrasonic.timing();
  cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);   //Converte a variável
cmMsec em centímetros.

    //Apresentar no monitor serial.
    Serial.println("Lendo dados do sensor...");
    Serial.print("distancia: ");
    Serial.print(cmMsec);
    Serial.print("...");
    delay(2000);
```

```
            if ((cmMsec > 30) &&(cmMsec < 50 )) { //Se a distância em for entre 30 e 50 cm,
executa as definições a baixo.
              digitalWrite (pino_saida,HIGH);
             if (alarme_pir0 == 0){ // Se variável receber 0, prossegue
             {return;}          //Responsável por parar verificação
            }
}

            WiFiClient client = server.available(); // Conexão com servidor.
            if (!client) {    // Verifica se o cliente está conectado ao servidor, executa este ciclo até
estar conectado.
              return;   //Encerra o ciclo, espera até o cliente enviar dados
            }

           Serial.println("novo cliente");

             while(!client.available()){       //Quando cliente conectado.
             delay(10);
            if(client.available())    {           //Se conectado.
             char c=client.read();              // ler caractere por caractere vindo do HTTP
              if(readString.length()<30) { // Se conter menos que 30 caracteres
                readString +=(c);              // armazena os caracteres para String
           }
}
           String pedindo = client.readStringUntil('\r'); // Ler a primeira linha do pedido
           Serial.println(pedindo); //Apresenta o pedido no monitor serial
           client.flush(); //Aguarda até todos caracteres de saída sejam enviado.




           int valor2 = LOW; //Iniciar variável com valor baixo.

           // Se comandos HTML recebidos
```

```
if (pedindo.indexOf("/ALARME=ON") != -1) {
 Alarme_Ativo();alarme_pir0 = 1; // Variável recebe 1, ativando alarme.
 valor2 = HIGH;  }

if (pedindo.indexOf("/ALARME=OFF") != -1)  {
 Alarme_Desativado(); //Void
 alarme_pir0 = 0; // Variável recebe 0, desativando alarme.
 valor2 = LOW; }

int valor = LOW;
if (pedindo.indexOf("/LED=ON") != -1)  {
digitalWrite(ledPin, HIGH); //Acende os Piscas
valor = HIGH; }


if (pedindo.indexOf("/LED=OFF") != -1)  {
digitalWrite(ledPin, LOW); // Apaga os Piscas
valor = LOW; //Define nível lógico baixo à variável. }

if (pedindo.indexOf("/LED1=ON") != -1)  {
digitalWrite(ledPin1, HIGH); // Acende os Faróis
valor = HIGH;.}

if (pedindo.indexOf("/LED1=OFF") != -1)  {
digitalWrite(ledPin1, LOW); //Apaga os Faróis
valor = LOW;}




// Inicialização da página HTML (Configurações)
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
```

```
        client.println(""); //
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");
          client.println("<head>");
                    client.println("<title>Aula01</title>");
                    client.println("<meta name=\"viewport\" content=\"width=320\">");
                    client.println("<meta name=\"viewport\" content=\"width=device-
width\">");
                    client.println("<meta charset=\"utf-8\">");
                    client.println("<meta name=\"viewport\" content=\"initial-scale=1.0, user-
scalable=no\">");
                    client.println("</head>");
                    client.println("<body>");
                    client.println("<center>");

        // Cabeçalho da página
                    client.println("<font size=\"5\" face=\"verdana\"
color=\"black\">Meu_TCC</font>");
                    client.println("<font size=\"3\" face=\"verdana\" color=\"black\"> &
</font>");
                    client.println("<font size=\"5\" face=\"verdana\"
color=\"black\">Arduino_e_AppInventor</font><br />");


        //A baixo apresentação do estado dos faróis, alarme e portas, na página.

        client.print("Faróis:");
        if(valor == HIGH) {
        client.print("On");
        } else {
        client.print("Off");
        }

        client.print("ALARME:");
```

```
if(valor2 == HIGH) {
client.print("On");
} else {
client.print("Off");
}

client.print("Portas:");
if(valor2 == HIGH) {
client.print("On");
} else {
client.print("Off");
}

 //Abaixo, criação e definições de botões.

client.println("<br><br>"); //Espacamento
client.println("<a href=\"/LED=ON\"\"><button>Turn On </button></a>");
// Ligar Piscas

client.println("<a href=\"/LED=OFF\"\"><button>Turn Off </button></a><br />");
// Desligar os Piscas

client.println("<a href=\"/LED1=ON\"\"><button>Turn1 On </button></a>");
 // Ligar Faróis

client.println("<a href=\"/LED1=OFF\"\"><button>Turn1 Off </button></a><br />");
//Desligar Faróis

client.println("<a href=\"/ALARME=ON\"\"><button>Turn2 On </button></a>");
//Ligar Alarme

client.println("<a href=\"/ALARME=OFF\"\"><button>Turn2 Off </button></a><br
/>");    //Desligar Alarme
```

```
        client.println("</html>");
        delay(1);
        Serial.println("Cliente desconectado"); // Depois do cliente efetuar o pedido apresenta esta mensagem no monitor série
        Serial.println("");
        }
```