UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM SISTEMAS EMBARCADOS PARA INDÚSTRIA AUTOMOTIVA

MARCELO OSS

LEITURA OBD2 ATRAVÉS DE SMARTPHONE

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2018

MARCELO OSS

LEITURA OBD2 ATRAVÉS DE SMARTPHONE

Monografia de Especialização, apresentada ao Curso de Especialização em Sistemas Embarcados para Indústria Automotiva, do Departamento Acadêmico de Eletrônica – DAELN, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Dr. André Schneider de Oliveira

CURITIBA
2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba

Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Eletrônica
Curso de Especialização em Sistemas Embarcados para Indústria Automotiva

TERMO DE APROVAÇÃO

LEITURA OBD2 ATRAVÉS DE SMARTPHONE

por

MARCELO OSS

Esta monografia foi apresentada em 05 de Dezembro de 2018 como requisito parcial para a obtenção do título de Especialista em Sistemas Embarcados para Indústria Automotiva. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. André Schneider de Oliveira
Orientador

_____
Prof. Dr. Kleber Kendy Horikawa Nabas
Membro titular

_____
Prof. M. Sc. Omero Francisco Bertol
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

RESUMO

OSS, Marcelo. **Leitura OBD2 através de smartphone**. 2018. 78 p. Monografia de Especialização em Sistemas Embarcados para Indústria Automotiva, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

O setor automobilístico utiliza inúmeras tecnologias existentes na atualidade e está em constante evolução. Com isso, o presente projeto visa desenvolver um sistema via hardware e software que realize a comunicação da rede principal do automóvel com a internet, executando leituras básicas de informações fornecidas pela norma OBD2, através do próprio navegador existente na maioria dos dispositivos. O estudo baseia-se na utilização de tecnologias com baixo custo e acessível a qualquer proprietário de automóvel. O embasamento teórico foi o alicerce para o desenvolvimento de um conjunto de aplicações que trabalham simultaneamente através de comunicações sem fio. Contempla-se neste documento as etapas para o desenvolvimento do sistema, a explicação teórica e prática de todo o processo, além de testes e analises que comprovaram o funcionamento e a eficácia de toda aplicação. Conclui-se que o cumprimento dos requisitos bem como os objetivos apresentados foram alcançados, porém, futuras atualizações visando critérios como, maior abrangência e melhorias no sistema de segurança de dados podem ser objetos de um novo estudo.

Palavras-chave: OBD2. Scanner. ELM327. Android. Conectividade.

ABSTRACT

OSS, Marcelo. **OBD2 reading through smartphone**. 2018. 78 p. Monografia de Especialização em Sistemas Embarcados para Indústria Automotiva, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

The automotive sector has used the existing technologies and is constantly evolving. With this, the project aims to develop a system through hardware and software that perform the communication of the main vehicle with the internet, perform basic information readings by the OBD2 standard, continue making use of the largest in most devices. The teaching is based on using technologies with low cost and ease of any car owner. The theoretical packaging was the foundation for the development of a set of objects that can be accessed through wireless communications. This document includes as steps for the development of the system, a theoretical and practical practice of the entire process, as well as tests and analyzes that prove the operation and application of the entire application. It concludes that compliance with the requirements has been successful, with the objective of achieving the first steps, greater availability and without any data security system.

Keywords: OBD2. Scanner. ELM327. Android. Connectivity

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ABS	*Anti-lock Braking System*
ACK	*Acknowledge*
ALDL	*Assembly Line Diagnostic Link*
API	*Application Programming Interface*
ARM	*Advanced RISC Machine*
BAAS	*Backend as a Service*
CAN	*Controller Area Network*
CLI	*Command Line*
CONAMA	Conselho Nacional do Meio Ambiente
CPU	*Central Processing Unit*
CRC	*Cyclic Redundancy Check*
CS	*Checksum*
CSMA	*Carrier Sense Multi-Access*
CSS	*Cascading Style Sheets*
CSV	*Comma-Separated Values*
DLC	*Data link Connector*
DTC	*Diagnostic Trouble Code*
ECU	*Engine Control Unit*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
GM	*General Motors*
GPS	*Global Positioning System*
HSDPA	*High Speed Downlink Packet Access*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IOS	*Iphone Operating System*
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
LED	*Light Emitting Diode*
MAC	*Media Access Control*
MAP	*Manifold Absolute Pressure*
MIL	*Malfunction Indicator Light*

MIT	*Massachusetts Institute of Technology*
MMS	*Multimedia Messaging Service*
OBD	*On-Board Diagnostic*
OHA	*Open Handset Alliance*
OSI	*Open Systems Interconnection*
OS	*Operating System*
PC	*Personal Computer*
PDA	*Personal Digital Assistant*
PID	*Parameter ID*
PMS	Ponto Morto Superior
PROCONVE	Programa de Controle da Poluição do Ar por Veículos Automotores
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computer*
SAE	*Society of Automotive Engineers*
SDK	*Software Developers Kit*
SOAP	*Simple Object Access Protocol*
SOF	*Start of Frame*
SRAM	*Static Random Access Memory*
UDDI	*Universal Description, Discovery and Integration*
USA	*United States of America*
WSDL	*Web Service Description Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Os sistemas eletrônicos automotivos vêm evoluindo devido à necessidade de controle dos gases nocivos ao efeito estufa, visto que o gerenciamento da eficiência da queima de combustíveis fósseis é de extrema importância. Para que ocorra este gerenciamento, cada vez mais sistemas eletrônicos são utilizados.

A inclusão de sistemas embarcados nos automóveis tem crescido exponencialmente nos últimos anos, chegando a ser responsável por um quarto do valor total do veículo em alguns casos. A utilização de componentes eletroeletrônicos se tornou fundamental para o setor automotivo, pois garante enormes benefícios quando confrontado a sistemas puramente mecânicos e hidráulicos (BLAKE; 2005).

Entretanto pelo cenário automotivo exigir robustez em seus sistemas, os seus componentes possuem desgaste ao decorrer do tempo, existindo a necessidade de manutenção. Esta manutenção serve tanto para o ótimo funcionamento do automóvel como também para que sejam compridos os requisitos ambientais por toda vida útil do automóvel.

No Brasil a idade média da frota circulante de automóveis de passageiros, passou de 8 anos e 6 meses em 2012 para 8 anos e 8 meses em 2014, mostrando um pequeno aumento da idade dos veículos brasileiros (SILVA, 2016). Esse crescimento exige melhorias no plano de manutenção dos veículos e na durabilidade dos sistemas embarcados.

Atualmente a maior parte dos veículos brasileiros possuem módulos que utilizam canais de comunicação para informar a situação do seu funcionamento, fornecendo um autodiagnostico que pode ser lido por ferramentas específicas conectadas aos automóveis.

Neste trabalho, é apresentado o desenvolvimento de uma aplicação que recebe as informações do automóvel e disponibiliza em uma página web, utilizando uma ferramenta de baixo custo existente no mercado e um smartphone com conexão com a internet, que permite o monitoramento das condições do veículo

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Desenvolvimento de uma aplicação para smartphones Android que realize uma leitura de dados disponibilizados por módulos existentes em automóveis e envie para *Web Service,* tornando possível a visualização remota das informações.

1.1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um aplicativo Android que se comunique com o automóvel e com o *Web Service*.
- Desenvolver um *Web Service* que armazene os dados recebidos do aplicativo e da aplicação web desenvolvida.
- Desenvolver uma aplicação web que exiba e envie informações em tempo real ao *Web Service*.

1.2 JUSTIFICATIVA

A conectividade sem fio é uma tendência para o setor automotivo nos próximos anos. A capacidade de o automóvel conseguir se comunicar com a grande rede é algo realmente interessante, pois inúmeras são as opções e interesses que surgem a partir disso. Como por exemplo, envio de dados sobre o desempenho para auxílio em economia de combustível e auxílio em eventuais defeitos. Na maioria das vezes a leitura de dados em sistemas de diagnóstico, necessita que o veículo esteja localizado em um centro de reparo juntamente com um profissional equipado com devidas ferramentas.

Soluções existentes no mercado que utilizam ferramentas de baixo custo como a escolhida no projeto, apenas possuem a conectividade disponibilizada pelo alcance Bluetooth do smartphone utilizado, tornando-se inútil na maioria dos casos, pois uma pessoa leiga que não possui o devido conhecimento técnico não irá obter êxito ao fazer uma interpretação dos dados apresentados.

Já com a solução proposta é possível fazer uma leitura de dados e até mesmo prever diagnósticos preliminares do sistema, a partir de qualquer lugar com

acesso a internet. Pelo fato de as informações estarem disponíveis na internet é possível um técnico remotamente auxiliar o motorista na tomada de decisão e até mesmo indicar qual é o reparo necessário no automóvel.

1.3 ESTRUTURA DO TRABALHO

No Capítulo 2 são apresentados os contextos essenciais para a execução do trabalho, dando um embasamento teórico para iniciar o desenvolvimento da aplicação. No Capítulo 3 é onde a aplicação é desenvolvida, demonstra-se a metodologia utilizada para executar cada passo dos objetivos. No Capítulo 4, será executado uma série de testes utilizando a aplicação comprovando a sua utilidade, praticidade e funcionalidade. E por último, o Capítulo 5 onde conclui-se o trabalho com o fechamento da ideia e o tema abordado, mencionando as características do sistema e possíveis aprimoramentos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 EVOLUÇÃO DA TECNOLOGIA EMBARCADA EM AUTOMÓVEIS

A evolução da tecnologia está em todos os setores da sociedade, sempre visando facilitar a vida das pessoas nas situações do dia-a-dia. No setor automotivo não é diferente, sistemas eletrônicos mais sofisticados vêm sendo adaptado neste meio, para utilização tanto em sistemas de conforto como na área operacional do automóvel.

A utilização do primeiro sistema de injeção eletrônico embarcado no setor automobilístico foi no ano de 1957, pela empresa *Bendix Corporation*, mas este sistema não obteve muito êxito, pois existiam muitas limitações. Já em 1960 as patentes do projeto foram compradas pela Bosch, onde as ideias foram aperfeiçoadas surgindo o primeiro sistema de injeção eletrônica automotiva desenvolvida pela Bosch o sistema "*D-Jetronic*", equipado no veículo VW 1600 TL/E. No qual utiliza poucos componentes eletrônicos, mas já obtinha melhor desempenho. Já no Brasil esta tecnologia só chegou ao ano de 1989 com o lançamento nacional do gol GTI, entretanto já utilizava um sistema mais complexo (CONTESINI, 2015).

As evoluções dos sistemas de injeção sempre buscam fornecer um melhor desempenho da queima do combustível utilizado, obedecendo a normas pré-determinadas em cada país. No Brasil, existe um órgão que regulamenta e definem as normas e leis que deverão ser seguidas o Conselho Nacional do Meio Ambiente (CONAMA), com o Programa de Controle da Poluição do Ar por Veículos Automotores (PROCONVE), criado em 1986. Entretanto as normas europeias que normalmente são das mais rigorosas e inovadoras, e que tem determinado muito os avanços feitos em relação aos motores para atenderem estas exigências. No Brasil, existe uma tendência a adaptar os benefícios já obtidos de outras normas mundiais, especialmente as europeias e, portanto, observá-las pode ser uma forma de prever quais tecnologias serão adotadas mais tarde aqui (PAPAIOANNOU, 2005).

Entretanto as montadoras de automóveis com o passar do tempo começaram a utilizar a eletrônica embarcada não apenas nos sistemas de injeção eletrônica, mas também sistemas de segurança, conforto e entretenimento. As principais

classificações de sistemas são as seguintes: *Powertrain*, Carroceria, Chassi, Telemática e Multimídia. O *Powertrain* está relacionado aos sistemas que participam da propulsão do veículo, incluindo motor, transmissão e todos os componentes. O chassi refere-se às rodas, sua posição e movimento neste domínio, os sistemas são principalmente direção e frenagem. A carroceria inclui as entidades que não pertencem à dinâmica do veículo, sendo assim aqueles que suportam o usuário do carro, tais como o *Airbag*, o limpador, a iluminação, o vidro elétrico, o ar-condicionado e etc. A multimídia inclui o equipamento que é associado com entretenimento. Já o domínio telemático está relacionado a componentes que permitem a troca de informações entre o veículo e o mundo exterior como rádios, sistemas de navegação, acesso a internet e pagamento de pedágios.

Cada sistema eletrônico automotivo possui suas necessidades e peculiaridades. Por exemplo, o *powertrain* e chassi utilizam informações em tempo real, necessitando de um alto poder de processamento. A telemática apresenta requisitos para alta taxa de transferência de dados. Sendo assim, as soluções tecnológicas necessárias utilizadas são muito diferentes.

Essas combinações da tecnologia da informação e da comunicação, em tempo real, de voz e de dados tem viabilizado o desenvolvimento de sistemas de navegação, de segurança e de serviços de emergência para os casos de acidentes e/ou de problemas mecânicos, incluindo notificação de roubo e rastreamento, diagnóstico mecânico remoto e banco de dados com informações médicas do motorista. Esses serviços resultam da combinação da telefonia móvel com os sistemas de posicionamento global para o monitoramento dos veículos (MCALIDEN, 2000).

2.2 COMPONENTES ELETRÔNICOS AUTOMOTIVOS

Dentro do universo eletrônico automotivo existem inúmeros tipos de sensores aplicados, cada um com suas funções e características. A principal função de um sensor é ter a capacidade de fazer uma leitura de um fenômeno físico em valores mensuráveis como tensão, corrente e resistência, tornando possível que essas informações sejam interpretadas por uma ECU, processadas e resultarem em ajustes executados por atuadores.

2.2.1 Sensor de Temperatura

Os sensores de temperatura têm como principal função mensurar a grandeza física de temperatura em um sinal elétrico. Utilizam-se da variação da resistência elétrica de dois termistores fornecendo uma grande tolerância e precisão. São excelentes para diversas aplicações, pois possuem uma alta sensibilidade a mudanças de temperatura. Basicamente possuem duas classificações (ARAÚJO, 2017):

- NTC: São sensores que possuem o coeficiente negativo de resistência, onde a resistência diminui com o aumento da temperatura, utiliza-se de manganês, cobre, cobalto e níquel na sua fabricação. São os mais utilizados na linha automotiva.
- PTC: São sensores que possuem o coeficiente positivo de resistência, onde a resistência aumenta com o aumento da temperatura. Utiliza-se de silício em sua fabricação.

Estes sensores estão alojados em contato com líquidos ou gases que necessitam de controle. Diversas são as estratégias baseadas nas informações desses sensores, pois é através destes valores a ECU toma as suas atitudes programadas regulando como exemplo a dosagem de combustível, acionando o eletro ventilador e fazendo o controle de circulação de gases de escapamento.

2.2.2 Sensor de Rotação

Um dos mais importantes requisitos para um ótimo desempenho do motor é o seu sincronismo. A peça chave deste sistema são os sensores de rotação, são eles que informam e dão a base para o sistema de injeção e a ignição. O sistema de injeção necessita da informação para saber qual é o momento certo para a injeção de combustível mais conhecido como tempo de injeção. Já o sistema de ignição necessita da informação para saber qual é o momento correto para acionar a centelha na câmara de combustão.

Estes sensores de rotação geralmente estão localizados próximos a eixos que giram conforme o funcionamento do motor, conhecidos como virabrequim e comando de válvulas. Juntamente com esses eixos existem polias dentadas no qual

os sensores são posicionados próximos fornecendo o sinal conforme sua especificação. Nas polias dentadas possuem chanfros que ao girar identificam o PMS, que é a posição fundamental no sincronismo do motor.

Existem diversos tipos de sensores de rotação, os mais utilizados são os sensores de efeito hall e os indutivos, assim descritos (DIAS, 2012):

- HALL: Fornece uma onda quadrada de sinal, parecida com um sistema digital, entretanto em sistemas mais novos necessita de um conversor digital mesmo assim.
- Indutivo: Nestes sensores a tensão é alternada possuindo um sinal muito parecido com uma curva senoidal, assim sendo é analógico e necessita de um conversor digital para ser possível ser interpretado pela ECU.

Em alguns sistemas modernos existem dois sensores de rotação que gerenciam o sincronismo do motor: O sensor de rotação do motor e o sensor de fase. No qual fornecem uma precisão muito melhor, no seu trabalho.

2.2.3 Sensor de Velocidade

Sensores de velocidade são semelhantes a sensores de rotação, fornece um sinal com forma de onda cuja frequência é proporcional á velocidade do automóvel. Entretanto na roda dentada não possui um chanfro que no sistema de rotação do motor identifica o PMS. Geralmente estão localizados no sistema de transmissão do automóvel dentro da caixa de câmbio.

O sinal de sensor de velocidade tem inúmeras utilidades para as ECUs, como por exemplo:

- Controle da mistura de combustível.
- Controle da marcha lenta.
- Desligar o eletro ventilador em velocidades elevadas.
- Controle para cálculos de médias e consumo.
- Hodômetro.
- Fornecer informações para sistema de transmissão automática.

Como pode-se perceber existem muitas funções relacionadas a sensor de velocidade, isso nos mostra a importância do funcionamento do mesmo, caso haja sua falha, inúmeras são os problemas que poderão ocorrer no automóvel.

2.2.4 Sensor de Pressão do Coletor de Admissão

Um meio de verificar a carga do motor é conseguir mensurar a pressão no coletor de admissão, para isso utiliza-se o sensor MAP. A pressão existente neste componente é utilizada no cálculo da massa de ar admitida e no cálculo do ponto da ignição.

Esta pressão exercida no coletor de admissão é chamada de pressão absoluta, oque significa a soma da pressão que o sensor está submetido no momento em que o motor funciona juntamente com a pressão atmosférica. Esta pressão absoluta pode variar conforme a pressão atmosférica, então o sistema utiliza a leitura deste sensor com o motor desligado para adequar o sistema conforme a pressão atmosférica exercida sobre o automóvel.

Em veículos modernos, juntamente com este sensor MAP utiliza-se um sensor de temperatura que informa a temperatura do ar admitido auxiliando nos ajustes da injeção. Este sensor de temperatura utiliza um termistor de coeficiente negativo, no qual o aumento da temperatura diminui sua resistência.

A localização deste sensor pode variar, em alguns casos possui uma mangueira acoplada no coletor e no sensor, neste caso facilita a sua fixação. Entretanto podem aparecer localizados juntamente com o coletor de admissão.

2.2.5 Unidades de Controle Eletrônico (ECU)

As Unidades de Controle Eletrônico (ECU) são a parte fundamental para todo o sistema embarcado automotivo é o cérebro onde todas as decisões são tomadas, baseadas na leitura das informações recebidas dos sensores utilizados em cada sistema.

Após receber as informações dos sensores analogicamente a ECU converte em informações digitais, sendo possível serem processadas através de funções algorítmicas programadas em sua memória, resultando em ajustes futuros ou imediatos executados pelos atuadores que estão sobre seu controle.

Cada unidade de controle possui a sua arquitetura específica. Entretanto pode-se classificá-las em cinco partes fundamentais:

- Processamento: É onde todos os cálculos são executados e todas as informações são tratadas através do processador embarcado.
- Memória: Localizam-se as memórias RAM, ROM e Flash. É onde são armazenados os softwares e nos sistemas de injeção eletrônica ficam armazenados os mapas de injeção, curvas características e algoritmos de comando.
- Entradas: São os meios que a ECU recebe as informações dos sensores analogicamente ou de e outras ECUs através na rede, digitalmente.
- Saídas: É o controle dos atuadores que são executadas por drivers e saídas Lógicas.
- Rede de comunicação: São os canais de comunicação entre os módulos, que necessitam da troca de informações.

As ECUs não são utilizadas apenas no controle do motor, elas estão localizadas em paralelo no gerenciamento de praticamente todos os sistemas mecânicos e não mecânicos do automóvel. São utilizadas em sistemas críticos como *Airbag* e ABS. Ou em sistemas de conforto como multimídia, vidros elétricos e ar-condicionado.

Pelo fato de existirem inúmeros módulos eletrônicos na fabricação do automóvel, tornou-se necessário a criação de redes. Para assim fornecer um meio de comunicação onde todos os módulos consigam conversar entre si obtendo dados, facilitando o acesso à informação de cada um dispensando uma duplicação de sensores e cabos auxiliando na redução de custo na fabricação dos automóveis.

2.3 REDES AUTOMOTIVAS

Com o desenvolvimento de sistemas com funções mais complexas, os automóveis receberam um aumento significativo de componentes eletrônicos. Para fazer um processamento de dados mais eficaz as funções foram sendo divididas em pequenas ECUs, descentralizando os processos. E consequentemente redes necessitam de melhorias para que os dados trafegarem com segurança e velocidade interligando uma ECU na outra.

As redes seguem uma classificação, segundo a sociedade dos engenheiros automotivos (SAE), em que são divididas em três categorias de acordo com a velocidade de transmissão e funcionalidades, como mostra a Tabela 1.

Tabela 1 - Classificação das redes automotivas

Classe	Velocidade	Aplicação
A	Menor que 10 kb/s	Vidros, travas e espelhos retrovisores elétricos.
B	10 kb/s a 125kb/s	Dados de sensores e informação painel.
C	Acima de 125kb/s	Dados em tempo real, dinâmica do veículo.

Fonte: Autoria própria.

Inúmeras configurações de rede são utilizadas nos automóveis hoje em dia, algumas possuem benefícios e desvantagens comparadas às outras. Cada rede se adapta melhor à situação destinada, sempre visando o custo benefício e as necessidades de cada uma.

Pelo fato da comunicação de dispositivos eletrônicos cresceram muito ao longo do tempo, se tornou necessária uma padronização no modelo de comunicação, para ser possível que dispositivos de fabricantes diferentes se comuniquem.

Então em 1984 a ISO lançou o modelo OSI, um modelo que padroniza o desenvolvimento das redes nos mais diversos setores de comunicação. Este modelo é subdividido em sete camadas, cada um executa uma contribuição na execução do sistema, conforme mostrado na Tabela 2.

Tabela 2 - Camadas modelo OSI

7	Aplicação
6	Apresentação
5	Sessão
4	Transporte
3	Rede
2	Enlace
1	Física

Fonte: Autoria própria.

A aplicação é a camada que envolve o software que é utilizado para promover a interação entre as ECUs. Na camada apresentação é onde a informação apresenta seu formato de leitura, é onde se executa a tradução do dado recebido pela camada aplicação, neste nível é onde as criptografias dos protocolos são aplicadas.

Já na camada sessão é responsável por garantir a troca de informação entre os hosts. É onde a sincronização das mensagens é executada. Possui um mecanismo de resposta de confirmação, quando o dado é recebido o módulo receptor ele envia uma mensagem de confirmação que recebeu a mensagem, fornecendo assim uma segurança tanto no recebimento dos dados como garantindo um tráfego de informação não duplicado, e caso existam falhas neste meio, a informação é enviada a partir do último ponto de sincronização executado.

O transporte possui a função de desmontar e montar os dados. Como por exemplo, os dados recebidos pela camada sessão são divididos e enviados para camada rede. Já no lado receptor do transporte os dados que são recebidos pela camada rede são remontados e enviados para sessão.

As funções principais da rede é operar o endereçamento das ECUs, executar o roteamento das mesmas e realizar a verificação dos erros, por meio da entrega de relatórios.

Dentro do enlace encontra-se a sequência de bits já tratada para o meio físico, com a integridade dos dados já garantida, e caso necessário execute a correção dos erros. Controla também o tráfego de dados, não permitindo que o transmissor envie mais mensagens que o receptor possa receber.

E a camada física por sua vez é onde a parte eletrônica trabalha. Garantir que bits “1” e “0” sejam compatíveis com os níveis de tensão que os representam. E transmitir a sequência de bits tratadas pelo enlace garantindo que não haja conflitos nos canais de comunicação.

2.3.1 Arquitetura

Dentro da arquitetura dos sistemas eletrônicos automotivos encontra-se todos os componentes eletrônicos existentes em um automóvel: bateria, alternador, motor de arranque, sensores, atuadores, ECUs, chicotes e etc.

A forma que estes elementos estão distribuídos e interligados pelo automóvel se diz respeito a sua arquitetura que pode ser uma arquitetura centralizada ou distribuída.

Na arquitetura centralizada é a mais simples, um módulo eletrônico executa toda a tarefa de processamento dos sensores. Entretanto a sua principal desvantagem é o acúmulo de cabeamento, pois todos os sensores e atuadores

independentes da sua localização estão interligados ao módulo, utilizando uma rede apenas para questões de diagnóstico do sistema.

Já a arquitetura distribuída ela concentra uma quantidade de módulos maiores espalhados pelo automóvel interligados por uma rede, dispensando uma quantidade enorme de cabos, pois os sensores e atuadores ficam interligados no módulo mais próximo, necessitando apenas de um chicote para a rede.

As montadoras automotivas oferecem produtos veiculares com uma vasta gama de funções proprietárias ou não (disponibilizadas por fornecedores), e em função disso podem estruturar suas arquiteturas distribuídas de forma mais eficiente possível, objetivando reduzir custos de produção, diagnóstico, manutenção e melhoria constante na qualidade dos serviços oferecidos (SANTOS, 2010, p. 45).

2.3.2 Topologia

A topologia das redes automotivas são os meios em que as ECUs estão ligadas umas às outras, que são representados através de meio físico ou meio lógico. O meio físicos são basicamente o formato que a rede aparenta e como está organizada os nós, é o meio de conexão dos componentes que estão distribuídas dentro do automóvel. Já nos meios lógicos não se leva em conta onde o nó está entrelaçado, pois a informação pode ser recebida por um módulo e ele próprio encaminhar para chegar ao seu destino, não havendo a necessidade da ligação física de dois módulos distantes.

As topologias mais utilizadas nos meios automotivos são: Estrela, Barramento, Árvore e Malha. Tudo dependendo dos protocolos adotados.

2.3.3 Protocolos de Comunicação

Protocolos de comunicação são meios de transmissão e recepção de dados utilizados para intercomunicar módulos eletrônicos e/ou sensores e atuadores inteligentes equipados com microcontroladores e *transceivers*, por exemplo. Existem vários tipos de protocolos de comunicação, cada qual com suas características técnicas específicas e, portanto, com as suas aplicações mais apropriadas (GUIMARÃES, 2011, p. 209).

2.3.3.1 CAN

O protocolo CAN surgiu nos ano de 1986 pela Bosch, com o intuito de simplificar a grande quantidade de cabeamento utilizado nos automóveis da época que já possuíam sistemas de gerenciamento de motor, ABS e etc. Os seus principais benefícios eram alta taxa de transmissão de dado, imunidade à interferência eletromagnética, a capacidade de detectar erros e reduzir a imensa quantidade de cabos utilizados em um par de fios trançado.

A rede do CAN possui a característica Multi-Mestre. Onde todo o nó pode se tornar mestre ou escravo um do outro. Para isto ele utiliza um método de sincronização CSMA, que antes de cada nó enviar a mensagem ele ouve a rede e identifica oque está passando por ela, se o barramento estiver desocupado o nó envia a mensagem. Entretanto caso exista uma mensagem a sua prioridade é verificada e enviada conforme a mesma.

O seu sistema de envio de mensagens pode ser considerado um barramento série. Pois é fácil ligar subsistemas inteligentes no próprio barramento. Utiliza um tamanho de 8 Bytes de mensagem e sua taxa de transmissão pode chegar até um 1 Mbits/s.

A evolução da rede CAN, classificou-a em dois tipos: CAN 1.0, CAN 1.1, CAN 1.2, CAN 2.0A E CAN 2.0B. O *data frame* dessas versões basicamente são montadas com a estrutura conforme a Figura 1.

Figura 1 - Frame dados CAN

Início de Frame	Campo de arbitragem	Campo de controle	Campo de dados	Campo CRC	Campo ACK	Fim de frame

Fonte: Autoria própria.

O início do frame indica quando a transmissão inicia tornando-o um bit dominante, pois todos os nós necessitam se sincronizar com a transição provocada. No campo de arbitragem é onde as prioridades são distribuídas, através dela são classificadas quais mensagens são mais importantes e deverão ser encaminhadas antes.

O campo de controle indica, qual vai ser o tamanho da mensagem do campo de dados. No campo de dados é onde os dados enviados estão localizados, trazendo o corpo da mensagem.

O campo CRC serve para verificar se a mensagem recebida não foi corrompida. Já no campo ACK é informado se a mensagem foi recebida por completo. E por último é o fim do frame, uma sequência de 7 bits recessivos finalizando o data frame.

2.3.3.2 KWP

Dentro dos protocolos utilizados na linha de diagnóstico automotivo encontra-se a KWP, é um protocolo que traz o conceito de cliente/servidor, onde a ferramenta de diagnóstico utilizada é a cliente e as ECUs os servidores.

Sua padronização é baseada na norma ISO 14230, no qual possui uma estrutura de mensagens divididas em três partes. A primeira é o campo de cabeçalho possuindo 4 bytes, a segunda o campo de dados com o total de 255 bytes e a última é o byte de checagem de erro chamado CS.

2.3.3.3 ISO 9141

A norma ISO-9141 utiliza-se de um ou dois fios como meio físico, possui uma taxa de transmissão em torno de 10 Kbps um data frame de 11 bytes. É apenas utilizado durante a inicialização da ferramenta de diagnóstico para carregar o endereço. Possui uma forma de sinal de onda quadrada que pode variar de 0 Vcc a 12 Vcc.

Na sua estrutura de pinos tem-se a linha K no pino 7 e na linha L o pino 15. Seu fluxo de dados pode ser bidirecional na linha K e unidirecional na linha L.

2.3.3.4 SAE J1850

A norma SAE J1850 da atribuição de rede a cada nó usuário com base no conceito de arbitragem, esse processo que determina qual dos nós têm prioridade de acesso ao barramento quando dois ou mais deles desejam transmitir mensagens no barramento de forma simultânea. O barramento J1850 é assíncrono, sem mestre, ponto a ponto que oferece acesso igual ao barramento para cada nó. Um atributo importante dele é que um nó transmissor difunde sua mensagem sob o barramento. Isso significa que não só todos os outros recebem a mensagem transmitida, mas o nó de transmissão vê sua própria mensagem sendo transmitida sob o barramento. Todas as mensagens são assíncronas por natureza, e qualquer dispositivo enviando

mensagens determina quando uma mensagem de transmissão pode iniciar, podendo realizar a transmissão entre mensagens sem um intervalo predefinido (SANTOS, 2010, p. 143).

Este protocolo é fundamentado na transmissão de dados com dois tipos de modulação:

- PWM: Alta velocidade de transmissão de dados com 41,6 Kbps e tensão diferencial com dois fios.
- VPW: Baixa velocidade de transmissão com 10,4 Kbps e baseado em um único fio.

2.3.3.5 SAE J1939

O desenvolvimento do protocolo SAE J1939 surgiu a partir da rede CAN 2.0B. As suas principais características são: Trabalha com uma comunicação broadcast de ponto a ponto, possui gerenciamento de rede, definição do grupo de parâmetros e protocolo de transporte.

É muito utilizado em comunicações de diagnóstico projetado para suportar funções em tempo real. Foi estruturado pelo modelo OSI, sendo definido em cinco camadas: Aplicação, transporte, rede, enlace de dados e camada física.

Outra característica é que cada ECU carrega um endereço no barramento, desde que seja registrado na rede. Para ser registrado na rede a ECU faz uma requisição de determinado endereço na rede, se este endereço estiver disponível o registro é feito, caso contrário a ECU com maior prioridade utiliza o registro, necessitando de uma nova requisição para obter êxito com outro endereço.

2.4 OBD

Devido à grande produção diária de gases nocivos ao efeito estufa com o aumento da frota global de automóveis na década de 80. Surgiu a necessidade de existir um gerenciamento em cada automóvel para evitar um descontrole na poluição pré-estabelecida pelas normas ambientais durante a vida útil do automóvel.

Foi assim que houve o surgimento do sistema de auto-diagnóstico OBD, no qual o próprio sistema do automóvel verifica através dos seus sensores que controlam as emissões se estão dentro dos padrões determinados, e caso exista

alguma avaria o sistema gera um relatório indicando ao condutor que necessita de manutenção.

Esse sistema teve também o intuito de padronizar os sistemas de diagnóstico automotivo, pelo fato de existirem inúmeras montadoras com diversos sistemas o diagnóstico a manutenção dos mesmos acabou se tornando muito cara. Pois esta diversidade de sistemas aumenta o custo em ferramentas de diagnóstico e capacitação dos reparadores, encarecendo a manutenção para o consumidor final.

Anteriormente da norma OBD a forma de executar o diagnóstico nos sistema era muito rudimentar em alguns casos apenas a luz MIL de anomalia acende, e todo o procedimento de encontrar a falha ficava por conta de o reparador, verificando todos os sensores e através do seu conhecimento resolver o problema.

Entretanto a evolução desta norma ao longo do tempo passou por algumas versões até chegar à utilizada hoje em dia OBD2. Em alguns lugares do mundo foram adaptadas nomenclaturas diferentes com algumas alterações, mas baseadas na mesma arquitetura como, por exemplo, as normas: EOBD utilizada na Europa, e a norma JOBD utilizada no Japão.

2.4.1 ALDL

A versão inicial considerada o precursor do OBD chamada de ALDL desenvolvida pela GM utilizavam pelo menos quatro conectores de ferramentas de diagnóstico diferentes. Seus layouts de conectores mudavam a cada geração de ECU. Assim não havendo padronização dos protocolos nem das portas.

2.4.2 OBD 1.0

O OBD1 foi uma tentativa frustrada de conseguir padronizar as informações sobre emissões. Houve muita fragmentação da estruturação destes protocolos e muitos desencontros entre os fabricantes, não conseguindo executar com êxito o seu principal objetivo que era: Juntamente com as inspeções veiculares da época, executar uma verificação dos dados de emissão dos automóveis registrados nas ECUs, reprovando e bloqueando automóveis que não estivessem dentro das normas.

Esta norma utilizava como meio de informação para diagnóstico uma sequência de “piscadas” da lâmpada MIL, este sistema era ativado quando se

conectava um pino no outro na sua tomada de diagnóstico, não havendo a necessidade de plugar uma ferramenta específica para obter a informação. Entretanto ainda o sistema era rudimentar e necessitava da interpretação do reparador para ser possível identificar qual era o problema que ocorria no sistema.

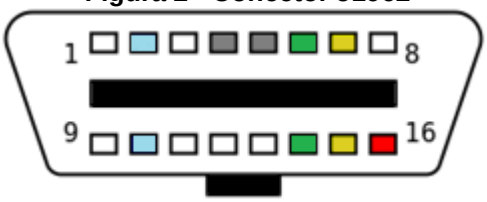
2.4.3 OBD 2.0

A norma OBD2 começou a ser desenvolvida nos Estados Unidos e na Europa no ano de 1996 e no Brasil se tornou obrigatório a partir do ano de 2010. Atualmente praticamente todas as montadoras utilizam-se desta norma na produção dos seus automóveis.

Está norma possui um padrão tanto no conector de diagnóstico, como nos sinais elétricos e no formato das mensagens. Principalmente é utilizada para monitorar itens importantes do funcionamento do motor, identificar falhas, armazená-las e notificar o motorista que existe algo de errado no seu veículo.

A padronização física da sua tomada se tornou algo essencial, tanto na localização como no layout da sua tomada, pois ferramentas de diagnóstico não necessitando de adaptadores consequentemente se tornam mais baratas, comuns e práticas. No caso da norma OBD2 o conector utilizado DLC utiliza a norma J1962 como pode-se observar na Figura 2.

Figura 2 - Conector J1962



Fonte: Autoria própria.

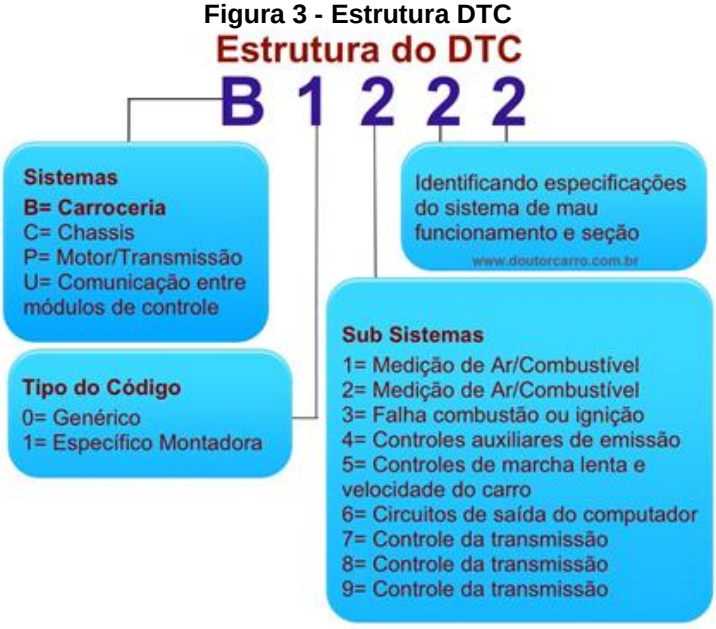
Através da organização dos seus pinos pode-se identificar quais são os protocolos de comunicação embarcados em cada automóvel. Os pinos dois e dez pintados de azul na imagem são responsáveis pelos protocolos J1850 VPW e PWM. Nos pinos quatro e cinco ilustrados em cinza é onde encontra-se o aterramento e em vermelho no pino dezesseis encontra-se a alimentação das ferramentas de diagnóstico. Em verde nos pinos seis e quatorze encontra-se a rede CAN. E finalmente nos pinos sete e quinze em amarelo encontra-se o protocolo ISO 9141-2 linha K e L.

2.4.3.1 DTC

DTC é o código de erro armazenado pelo sistema OBD2 quando ocorre algum problema no sistema e a ECU identificou. Para identificar falhas no sistema OBD2, são executados testes no circuito de todos os sensores e monitores procurando por possíveis curtos circuitos, circuitos abertos, plausibilidade do sinal é também processos da ECU (codificação, memória etc.). Se uma falha for detectada por algum dos testes, o código de defeito (DTC) é armazenado e caso a falha for relacionada com emissão de poluentes, a luz MIL será acionada. O DTC não é necessariamente a causa do defeito, mas sim, a falha isolada para uma área funcional específica do veículo (ALMEIDA; FARIA, 2013).

As falhas identificadas são informadas ao barramento de comunicação no formato de um código de falha conhecido como DTC, que possui a sua estrutura conforme a Figura 3.

Figura 3 - Estrutura DTC



Fonte: DoutorCarro (2014).

Este código utiliza uma sequência de cinco dígitos que utiliza dois bytes de memória na representação da sua estrutura, no qual o primeiro dígito informa a localização do sistema onde a falha está presente. No segundo dígito é designado para informar se é um código genérico ou específico da montadora e nos últimos três dígitos é estabelecido qual é o local e sensor onde a falha está sendo identificada.

2.4.3.2 PID

Os serviços de diagnóstico de um sistema OBD2 são organizados por modos de operação e códigos de parâmetros (PIDs) (SANTOS, 2010, p. 19). Estes PIDs estão registrados na memória em forma de um *vetor*, onde cada posição do mesmo também está localizada o código hexadecimal que o corresponde.

Todos os PIDs são regulamentados pela norma OBD2, em alguns casos não possuem obrigatoriedade ficando a critério da montadora a sua utilização. Entretanto os relacionados a emissões são obrigatórios.

Cada PID é um valor hexadecimal de dois dígitos, que devem estar entre 0x01 e 0x0A. Cada operação pode ter até 256 PIDs, que também são representados por um número hexadecimal de dois dígitos. Existe também o serviço 0x00, que é reservado à aderência do veículo ao padrão OBD, ou seja, a resposta retornada por ele é um vetor de bits em que cada valor binário indica se o PID correspondente é suportado ou não pela ECU (SANTOS, 2010, p. 19).

Existem dez serviços disponíveis, de modo que cada veículo ou ECU deverá programar seus serviços de acordo com a sua legislação (ALMEIDA; FARIA, 2013). Estes serviços são utilizados para classificar suas operações, onde cada uma aponta uma gama de trabalhos que podem operar, conforme pode-se visualizar na Tabela 3.

Tabela 3 - Lista de PIDs de serviço

Serviço	Descrição
01	Mostrar dados atualizados.
02	Mostrar dados do quadro congelado.
03	Mostrar códigos DTCs.
04	Limpar DTCs.
05	Resultados de teste, monitoramento do sensor de oxigênio.
06	Resultados de testes, outros componentes.
07	Mostrar DTCs pendentes.
08	Operação de controle do componente / sistema de bordo.
09	Solicitar informações do veículo.
0A	Mostrar DTCs apagados.

Fonte: Autoria própria.

- Serviço 0x01: encontra-se o meio do acesso à informação do sistema *Powertrain* informações de entrada e saída analógicas e digitais.

- Serviço 0x02: são apresentados os dados *Freeze Frame*, onde são armazenados os DTCs no momento em que ocorrem.
- Serviço 0x03: Lista os DTCs confirmados que estão relacionados a emissões. Através deste serviço que a ferramenta de diagnóstico obtém o DTC que fez com que a lâmpada MIL tenha acendido.
- Serviço 0x04: é utilizado pela ferramenta de diagnóstico para executar a limpeza na memória onde os DTCs estão armazenados.
- Serviço 0x05: é responsável pelos testes de sensores de oxigênio.
- Serviço 0x06: são apresentados outros tipos de testes: como falhas no sistema de combustão e *misfire*.
- Serviço 0x07: é semelhante a 03, entretanto é utilizado para verificar possíveis falhas futuras, onde não foi constatado o *Freeze Frame* ainda.
- Serviço 0x08: este serviço é bidirecional, ou seja, com ele é possível ler, alterar parâmetros de operação e é possível testar diferentes hipóteses para identificar a causa do problema. Esse serviço deve ser utilizado com cuidado, pois pode causar avarias ao veículo, através desse módulo, é possível alterar a potência do motor em alguns cavalos de força apenas por meio de reprogramação da ECU (SANTOS, 2010).
- Serviço 0x09: permite obter informações sobre o software instalado na ECU. É mais utilizado para consultar o *Vehicle Identification Number*, que é o código identificador único para carros e caminhões. Também serve para identificar a versão do software que está instalado e esse serviço permite obter relatórios sobre sistema catalisador, sensor de oxigênio, vazamento no sistema de evaporação, controle de pressão, contadores de eventos como o número de ignição e testes de autodiagnostico (SANTOS, 2010, p. 21).
- Serviço 0x0A: é utilizado para manter um histórico de anomalias do veículo não sendo possível apagá-los deste serviço com o método 0x04.

Dentro da operação do serviço 0x01 é possível obter informações a respeito dos sensores responsáveis pelo gerenciamento do motor. Na Tabela 4. estão representados alguns exemplos de PIDs de serviço 01.

Tabela 4 - Exemplos de PIDs de serviço 0x01

PID0x01	BYTES RETORNO	DESCRIÇÃO	VALOR MÍNIMO	VALOR MÁXIMO	UNIDADE	FÓRMULA
05	1	Sensor Temperatura motor	-40	215	°C	A-40
0C	2	Rotação Motor	0	16,388,75	RPM	256A+B/4
0D	1	Velocidade Automóvel	0	255	Km/h	A
0B	1	Pressão Coletor Admissão	0	255	kPa	A

Fonte: Autoria própria.

Estes PIDs exibidos são utilizados para requerimento da leitura de alguns sensores, que necessitam ser enviados juntamente com o primeiro PID de serviço o 01. Possui também o tamanho da resposta adquirida que geralmente é representada por dois números hexadecimais, necessitando da conversão para decimal para assim ser aplicado na fórmula. Nesta mesma tabela são exibidas as fórmulas, os limites e as unidades de medidas utilizadas para calcular os valores que serão ilustrados nas ferramentas de diagnóstico, que utilizam o padrão da norma OBD2.

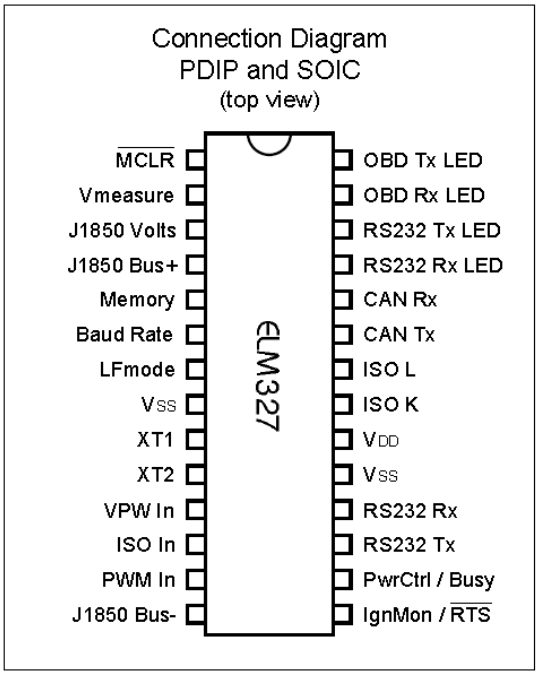
2.5 ELM 327

Após o início da utilização da norma OBD2, houve um surgimento de inúmeras ferramentas de diagnóstico, que acabou tornando-as mais populares e acessíveis. Ferramentas desse tipo estão custando no mercado em torno de 10 dólares sendo possível adquirir praticamente uma por automóvel.

Muitas destas ferramentas conhecidas utilizam o microcontrolador ELM327. Este componente desenvolvido pela *ELM electronics* desde 2005 tem a capacidade de detectar e conversar automaticamente com todos os protocolos OBD2. Com ele é possível desenvolver interfaces que se comuniquem com computadores ou smartphones, utilizando cabos ou sistemas sem fio como Bluetooth e *Wi-fi.*

Toda esta gama de opções é possível através da facilidade que encontra-se na estruturação dos seus pinos, como pode-se visualizar na Figura 4.

Figura 4 - Pinagem ELM327

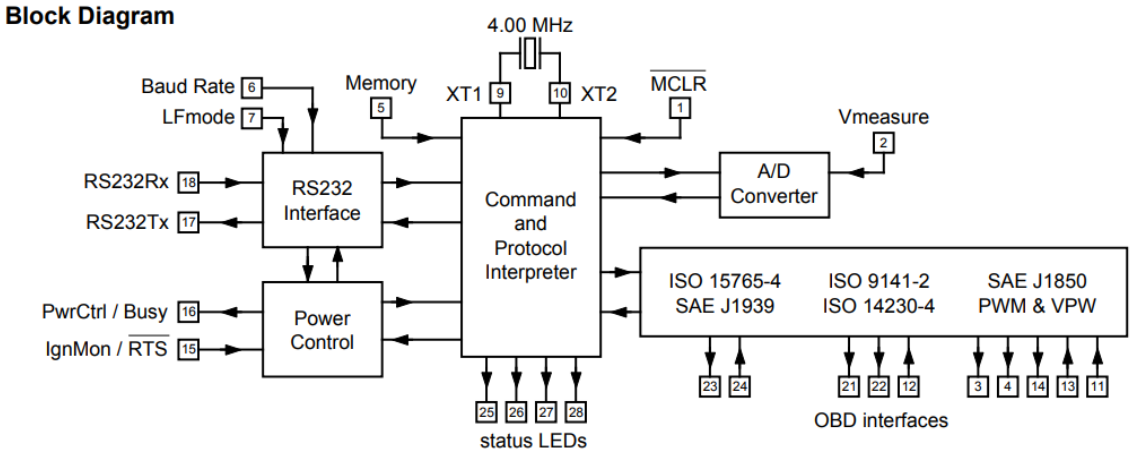


Fonte: ELM327 (2017).

Pelo fato de possuir as suas saídas digitais através de uma comunicação serial universal, ela pode tranquilamente ser adaptada a qualquer meio de envio e recebimento de informação serial, apenas utilizando o adaptador desejado. Possuindo um *Baud rate* de até 500 Kbps que é controlado pelo seu respectivo pino.

A leitura dos protocolos OBD2 como citados anteriormente possuem seus pinos independentes, facilitando assim a capacidade de identificação automática do protocolo embarcado no automóvel. A sua alimentação é de baixo consumo de energia, possui um controle de alimentação nos pinos de Vss, Vdd e *VMeasure*. Necessita de um cristal oscilador de 4Mhz ligado nos pinos XT1 e XT2. O seu diagrama de blocos (Figura 5) de uma forma mais exemplificada como são separadas suas funções através da sua pinagem.

Figura 5 - Diagrama de blocos ELM327



Fonte: ELM327 (2017).

A interface serial é executada através dos pinos: 6, 7, 18, 17. Já o controle da alimentação é executado pelos pinos 16 e 15. Já nos pinos 25, 25, 27, 28 exibe o status do funcionamento do microchip através de quatro LED's. A leitura dos protocolos OBD2 é executada pelos pinos 23, 24, 21, 22, 12, 3, 4, 14, 13, 11.

O sistema de controle do ELM327 é executado através de comandos AT. Esses comandos utilizam estrutura hexadecimal fundamentada na estrutura Hayes muito utilizada nas configurações internas de modens de acesso a internet. Esses comandos são basicamente formados por dígitos hexadecimais fornecidos em pares, utilizada tanto para envio de comandos e recebimento de informações. Totalmente compatível com os PIDS descritos no capítulo anterior.

Ao longo do tempo foram desenvolvidas inúmeras versões deste microcontrolador. Entretanto a última lançada até o momento é a versão 2.1. Utilizada na maioria dos adaptadores que utilizam a comunicação Bluetooth.

2.6 SMARTPHONE

O surgimento dos primeiros aparelhos celulares conhecidos como *Basic Phones* foi nos anos 90, onde as primeiras funções foram incorporadas nestes aparelhos como: serviço de agenda, calendário e envio de mensagens. Funções consideradas básicas hoje em dia, já eram muito avançadas para época. Entretanto no meio da transição de *Basic Phones* para Smartphones surgiram os *Features phones*.

Os *Feature phones*, que seriam aqueles intermediários, que possuem a tela menor, uma capacidade de processamento limitada, e que possuem um sistema operacional próprio que não eram os conhecidos Android, Apple IOS, Windows Phone ou Blackberry OS, não sendo capazes de processar APIs muito elaboradas, geralmente rodando JAVA ou BREW, e os *Basic phones* ou "*Dumbphones*", que seriam aqueles extremamente simples, sem qualquer tipo de conexão com redes mobile de internet ou Wi-fi e baixíssimo poder de processamento e resolução (LEE, 2010).

Conforme a evolução tecnológica, os celulares passaram a incorporar as funções de cada vez mais dispositivas, tornando-se progressivamente mais importantes. Tinham processadores ARM de 300 a 400 MHz e 64 MB ou mais de memória RAM, superior ao de muitos computadores do final da década de 1990. Passaram a assimilar as funções de outros dispositivos, assim como no caso dos computadores, agendas eletrônicas, PDAs e os *Palms*, que, ao serem incorporados, deram origem aos smartphones que são utilizados atualmente (MORIMOTO, 2009).

O termo *Smartphone* é utilizado para nominar aparelhos celulares com altíssima tecnologia em referência ao processamento dos dispositivos atuais que possuem funções avançadas como: GPS, Bluetooth e Câmera Digital.

Smartphone é um celular com capacidade avançada, que executa um sistema operacional identificável permitindo aos usuários estenderem suas funcionalidades com aplicações terceiras que estão disponíveis em uma loja de aplicativos [...] devem incluir um hardware sofisticado com: a) capacidade de processamento avançada (CPUs modernas, sensores) b) Capacidade de conexões múltiplas e rápidas (Wi-Fi, HSDPA) e c) tamanho de tela adequado e limitado. Além disso, seu Sistema Operacional deve ser claramente identificável, como Android, Blackberry, Windows Phone, Apples IOS, etc. (THEOHARIDOU; MYLONAS; GRITZALDIS, p. 3).

Os smartphones necessitam de um alto desempenho de processamento e baixo consumo de energia. Na maioria dos casos os processadores equipados nos celulares utilizam a arquitetura RISC de 32 bits para garantirem as características desejáveis. Estes processadores são desenvolvidos por uma empresa britânica chamada de ARM Holdings, os processadores não são fabricados pela ARM, apenas a sua arquitetura que é fornecida e licenciada para as empresas fabricantes. Ficando para a ARM apenas a evolução das novas arquiteturas.

2.6.1 Android

Em meados de 2005 o Google comprou uma empresa que estava desenvolvendo um sistema para celulares e passou a desenvolvê-lo por conta, já em 2007, juntamente com outras cinquentas empresas, entre elas TI, Intel e LG, fundou a *Open Headset Alliance* com o objetivo de produzir um sistema *open source* para celulares, o Android (MORIMOTO, 2009).

O maior foco do Android e sua flexibilidade e juntamente com uma grande comunidade que se formou para o desenvolvimento de aplicações, grande partes devido aos diversos campeonatos de desenvolvimento organizados pela *Open Handset Alliance* (OHA) e pela facilidade de programar para Android, tem alcançando uma grande abrangência de tecnologias e dispositivos, como *multi-touch* e aceleradores gráficos 3D (MORIMOTO, 2009).

O seu lançamento não foi encarado inicialmente como uma ameaça por as outras empresas desenvolvedoras de smartphones, pelo fato de ser um sistema *open-source*. Entretanto em poucos anos se tornou líder no mercado justamente pela facilidade de terceiros desenvolverem aplicativos para seu sistema.

Estatísticas mostram que em 2009 o Android representava apenas 2,8% dos aparelhos vendidos no mundo; já no final do ano seguinte detinha 33%, ou seja, 1 em cada 3 aparelhos do mundo, o suficiente para transformá-lo já na plataforma móvel mais vendida do planeta. Em 2011 já tinha passado da metade, mais precisamente 52,5%, em 2012 passou para 75%, em 2013 para 78.7% e, em 2014, para 81,5% (MEYER, 2017).

Esses números aumentam a cada ano, pois diversos equipamentos utilizam o sistema Android como sistema operacional, sua arquitetura multiplataforma fornece essa flexibilidade. Equipamentos como: *netbooks, smart tvs*, relógios, tocadores mp3 e câmeras digitais, já utilizam esse sistema operacional.

2.6.2 Java

Java é a linguagem de programação orientada a objetos, desenvolvida pela Sun Microsystems, capaz de criar tanto aplicativos para desktop, aplicações comerciais, softwares robustos, completos e independentes, aplicativos para a Web. Além disso, caracteriza-se por ser muito parecida com C++, eliminando as

características consideradas complexas, dentre as quais ponteiros e herança múltipla (CLARO; SOBRAL, 2008).

Está linguagem é utilizada para desenvolvimento de *Applet's*, que são mini aplicativos que rodam dentro da página web, se tornando muito valiosa para desenvolvedores de sites. Entretanto é uma linguagem muito poderosa e vai além das simples aplicações, pois é através desta que a maioria dos aplicativos Android são desenvolvidos.

Existem vários tipos de ferramentas que são utilizadas para desenvolver aplicativos, dentre elas encontram-se ferramentas mais complexas com necessidade de um nível maior de entendimento de programação e encontram-se ferramentas mais simples que tratam de camadas mais altas no desenvolvimento de aplicativos, tornando possível que pessoas leigas desenvolvam seus próprios aplicativos com mais facilidade.

2.6.3 App Inventor

O App inventor é uma plataforma de desenvolvimento de aplicativos Android online, ela foi criada originalmente pelo Google e atualmente pertence ao MIT do USA. Utiliza de metodologias voltadas a natureza educacional com um ambiente de programação visual e intuitivo permitindo em poucos minutos uma pessoa que nunca programou consiga montar seu aplicativo.

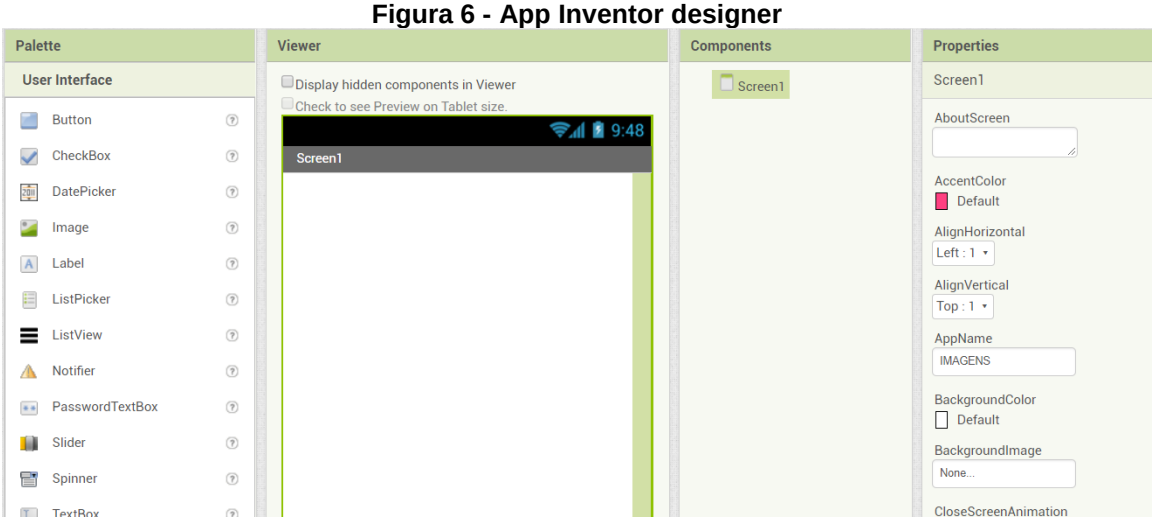
A sua interface dinâmica utiliza de um conjunto de blocos classificados e organizados que facilitam a representação de funções tradicionais de programação de uma forma mais interativa e simples de se entender, estimulando o interesse especialmente dos jovens para a área de desenvolvimento de tecnologia.

Estão disponíveis duas versões atualmente do App inventor. Há uma série de diferenças fundamentais entre o App Inventor 2 e sua versão original. Primeiro - e mais importante - o App Inventor 2 é totalmente executado no navegador. Com o App Inventor original, os usuários precisavam instalar e executar um arquivo Java. Houve também inúmeras melhorias para a experiência do usuário, bem como alterações estéticas (CLARK, 2013).

O sistema de desenvolvimento de aplicativos no App Inventor 2 e dividido em duas seções: *App Inventor Designer* e *Blocks Edition*. Na sessão *App Inventor*

Designer encontra-se a parte inicial onde a interface é configurada, como mostra a Figura 6.

Figura 6 - App Inventor designer



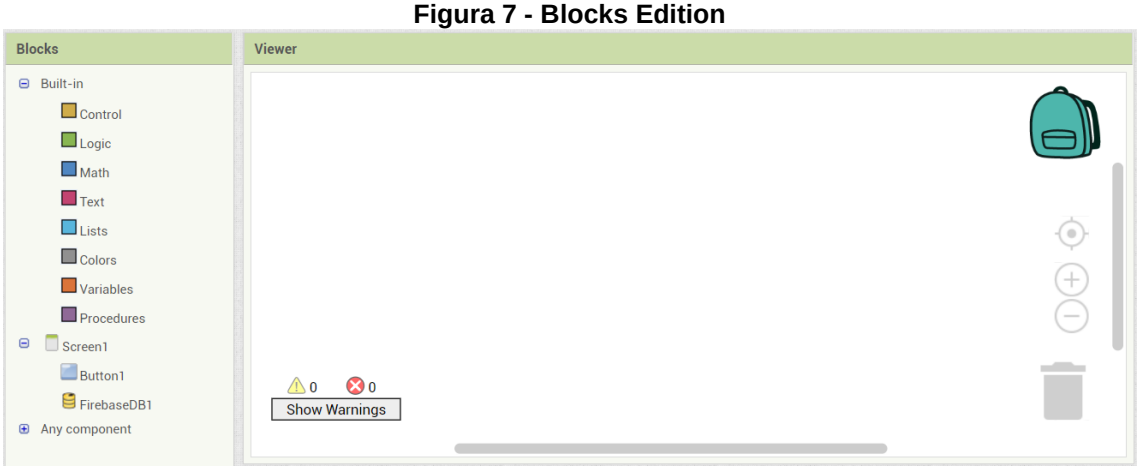
Fonte: Autoria própria.

A página inicial do *App Inventor 2* possui quatro divisões principais: Inicialmente encontra-se a divisão *Palette*, que é onde está todos os componentes utilizados para construção do aplicativo desde botões, caixa de texto, animações, funções de GPS e conexão a bancos de dados. Para utilizar os componentes apenas é necessário clicar sobre o mesmo e arrastar para a seção *Viewer*.

Já no outro campo, encontra-se o *Viewer*. É onde se organiza a interface responsável pelo aspecto visual do aplicativo. Ao lado se localiza o *Components* onde possui a lista de todos os componentes arrastados para o *Viewer*, sendo possível apagá-los e renomeá-los. E por último, encontra-se o *Properties*, onde são configurados todos os atributos responsáveis pela interface dos componentes e configura-se como, por exemplo, o acesso ao banco de dados externo do aplicativo.

Mudando a seção para a área *Blocks Edition*, encontra-se a área de desenvolvimento dos blocos como mostra a Figura 7.

Figura 7 - Blocks Edition



Fonte: Autoria própria.

O *Blocks Edition* é o ambiente no qual os usuários podem programar a funcionalidade dos componentes que eles adicionaram à sua aplicação. O *Blocks Edition* usa uma linguagem de programação de blocos para que usuários programem seus aplicativos. A linguagem de programação utilizada no primeiro *App Inventor* é baseada no *framework MIT Open Blocks*, uma biblioteca Java criada como parte do programa MIT STEP. A segunda versão do App Inventor usa uma programação de blocos linguagem projetada em JavaScript usando *Blockly*, (CHADHA, 2014).

Na aba *Blocks* encontram-se as funções que são executadas pela ferramenta. Dentro da aba *Blocks* no setor *Built-in* estão as operações essenciais que estão disponíveis para inseri-las, como por exemplo, funções Matemáticas, funções de texto, funções lógicas, funções de controle e etc.

Já na aba *Screen1* abaixo, localiza-se as funções que estão disponíveis conforme os componentes adicionados na sessão *App Inventor Designer*, que são funções específicas de cada componente inserido no projeto.

Ao lado, tem-se outro campo *Viewer*, mas este é responsável por receber os blocos, é onde as funções são encaixadas podendo compará-lo com a área do código do aplicativo, é onde se aplica a parte lógica utilizando todas as funções disponibilizadas.

Os blocos são de cores diferentes para que sejam mais fáceis de identificar e distinguir visualmente uns dos outros. Os usuários arrastam blocos que desejam usar dessas gavetas para o espaço de trabalho para compor seus programas (CHADHA, 2014).

As funções são disponibilizadas conforme os componentes são inseridos dentro da plataforma, como a plataforma é *open source* as funções inseridas podem ser desenvolvidas por terceiros. Em alguns casos existem componentes que estão em estado experimental, principalmente as funções com servidores externos que algumas vezes podem apresentar algum tipo de problema.

2.7 WEB SERVICE

Web Service é um meio de comunicação via internet, utilizada para a integração de diferentes softwares e aplicações, rodando em uma variedade indeterminada de plataformas. Tornando possível a comunicação de softwares novos com antigos, utilizando a tradução dos seus dados para formatos como XML, CSV, JSON e etc.

Basicamente ela disponibiliza informações através da rede de forma normalizada, dispensando a adaptação de linguagens diferentes. E sendo possível compartilhar os dados existentes de uma forma mais prática e eficaz, fazendo com que os recursos estejam disponíveis para qualquer aplicação consiga extrair ou alterar os dados existentes.

A implementação de um Web Service é baseada em um conjunto de protocolos e linguagens padrões de Web, das quais podem-se destacar: o HTTP, o SOAP, WSDL e o UDDI. O formato XML é a base dos três últimos elementos citados: SOAP, WSDL e UDDI.

2.7.1 Firebase

O Firebase é uma plataforma BAAS de desenvolvimento de aplicativos para Android, IOS e Web. Comprada pela Google em 2014, que fornece todo o gerenciamento para o desenvolvimento de aplicações.

BAAS é um serviço de computação em nuvem que serve como *middleware*. Ele fornece aos desenvolvedores uma forma para conectar suas aplicações mobile e web a serviços de nuvem a partir de APIs e SDKs. Esse tipo de serviço auxilia os desenvolvedores a acelerar a criação de aplicações web e mobile. Em vez de codificar o *backend* inteiro, o desenvolvedor usa o BAAS para criar as APIs e conectá-las às aplicações, desta forma a infraestrutura do lado do servidor é

abstraída completamente permitindo ao desenvolvedor se concentrar à experiência de usuário, o *frontend* (BATSCHINSKI, 2016).

2.7.7.1 Realtime Database

O *Realtime Database* é um banco de dados do Firebase, hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando você cria apps em plataformas cruzadas com SDKs para iOS, Android e JavaScript, todos os clientes que compartilham uma instância do *Realtime Database* e recebem automaticamente atualizações com os dados mais recentes (FIREBASE, 2018).

JSON é um modelo para armazenamento e transmissão de informações no formato texto. Apesar de muito simples, tem sido bastante utilizado por aplicações web devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido o *parsing* dessas informações. Isto explica o fato de o JSON ter sido adotado por empresas como Google e Yahoo, cujas aplicações precisam transmitir grandes volumes de dados (GONÇALVES, 2012).

2.7.7.2 Hosting

O Firebase *Hosting* foi criado pensando no desenvolvedor Web moderno. Quer você esteja implantando uma página inicial de app simples ou um app da Web progressivo e complexo, o *Hosting* conta com infraestrutura, recursos e ferramentas adaptadas a implantação e o gerenciamento de websites estáticos. Com a CLI do *Firebase*, você pode implantar arquivos de diretórios locais no seu computador para o seu server do *Hosting*. Além da hospedagem de conteúdo estático, o Firebase *Hosting* conta com opções de configuração leves para que você possa criar *Progressive Web Apps* sofisticados (FIREBASE, 2018).

Para que seja possível o envio dos diretórios do seu computador para o servidor *Hosting* é necessário a instalação do Node.js, no qual permite a execução do CLI do Firebase no computador desejado.

2.7.2 Node.js

O Node.js é um interpretador JavaScript do lado do servidor que altera a noção de como um servidor deveria funcionar. Seu objetivo é possibilitar que um programador crie aplicativos altamente escaláveis e escreva código que manipule dezenas de milhares de conexões simultâneas em uma, e somente uma máquina física. Node é um programa de servidor (SILVA, 2012).

2.8 VISUAL STUDIO CODE

O Visual Studio Code é uma ferramenta de edição de código fonte desenvolvida pela Microsoft para Linux, MacOS e Windows. É utilizada para edição de linguagens como: CSS, HTML, Javascript entre outras. Muito utilizada por desenvolvedores WEB, unificando assim em uma única ferramenta todos os meios necessários para o desenvolvimento dos seus projetos.

O VSCode como é conhecido, é capaz de abrir tanto um arquivo quanto uma pasta completa, podendo assim fazer a interação entre arquivos de uma forma mais eficaz. Possui um layout intuitivo e simples, objetivando ao máximo a área do desenvolvimento do código e o espaço de navegação das pastas do projeto.

2.8.1 HTML

No universo da internet, encontra-se um meio comum de comunicação entre as páginas web, todas elas se comunicam e interpretam informações através da linguagem HTML, esta linguagem surgiu nos anos de 1990 com a difusão da internet. Inicialmente possuía marcações simples que facilitavam a utilização na baixa velocidade de comunicação que existia na época.

Baseia-se em um conjunto de *tags* que decifram a organização de uma página Browser. Como por exemplos títulos, subtítulos, imagens, rodapés e corpo de mensagens, são posicionados pelas *tags* abertas e fechadas devidamente entre os textos que cada uma representa.

Estas marcações disponíveis na linguagem HTML é o que permite a interpretação de páginas web por diferentes navegadores, fornecendo a mesma

interface e organização. Os navegadores possuem a capacidade de ler as *tags* e assim montarem oque os usuários veem.

2.8.2 CSS

CSS é uma linguagem de folha de estilos, que tem o papel de tornar uma página apresentável na web, relacionada diretamente com o design e aparência. Ou seja, o CSS é uma camada que se usa para controlar o estilo da sua página da web (GONÇALVEZ, 2018).

No começo da internet, a produção de páginas web era totalmente baseada em HTML simples. A baixa velocidade de comunicação da internet e a pouca capacidade de processamento dos computadores da época tornavam os recursos de aparência limitados. Entretanto com a evolução da tecnologia de comunicação houve o aperfeiçoamento e embelezamento das páginas, e assim a linguagem HTML não supria mais as necessidades gráficas dos usuários. Através do HTML tornava-se uma tarefa difícil para os *Web Designers* embelezarem as suas páginas desenvolvidas, pois cada alteração era necessária alterar todas as páginas do projeto manualmente e assim dava muito trabalho para padronizar os *layouts* dos sites.

A partir desta dificuldade surgiu o CSS, no qual através de uma folha complementar de estilo ligada na página HTML, carrega toda a parte da apresentação das páginas. Executam as alterações no ambiente gráfico através de chamadas de classes da página HTML. Ou seja, você cria uma classe na folha CSS com a formatação escolhida e quando for iniciar um campo na página HTML que deseja usar esta formatação, apenas chama-se a classe desejada através de uma *tag*.

2.8.3 JavaScript

JavaScript é uma linguagem de programação script criada em 1995 pela Netscape. Esta linguagem é utilizada no desenvolvimento de web sites, utilizada muitas vezes no próprio documento HTML, possibilitando inúmeras funcionalidades e incrementos no documento tornando uma página mais dinâmica e sofisticada.

JavaScript permite criar pequenos programas embutidos no próprio código de uma página HTML e capazes de gerar números, processar alguns dados, verificar

formulários, alterar valor de elementos HTML e criar elementos HTML. Tudo isso diretamente no computador cliente, evitando a troca de informações com o servidor e o tempo passa a depender somente do processamento local do cliente, não mais da latência da rede (GRILLO; FORTES, 2008).

A partir do momento que uma página web faz mais do que apresentar conteúdo estático, começa a apresentar conteúdos atualizados, imagens 3D, mapas interativos, vídeos e imagens em movimento, são códigos em JavaScript que estão sendo executados e permitindo que isto ocorra.

2.9 REDES SEM FIO

Conhecida como Wireless às redes sem fio são um meio de transmissão e comunicação de dados sem a necessidade de cabos. A comunicação é dada pela transmissão de ondas eletromagnéticas pelo espaço, onde existe o transmissor e receptor destas ondas. Não necessariamente é uma conexão com a internet, equipamentos como mouses, fones de ouvido, controles entre outros dispositivos que não utilizam cabos possuem uma rede Wireless.

As redes sem fio surgiram como redes complementares às redes cabeadas, com o intuito de promover a mobilidade e a visualização rápida dos dados independentemente da localização do usuário, tendo os dados transmitidos pelo ar ou espaço livre, que se constituem como meio físico para propagação de sinais eletromagnéticos, provendo uma interconexão completa, e permitindo uma grande flexibilidade na localização das estações, sendo essa a principal diferença entre as redes sem fio e as redes convencionais (CONCEIÇÃO JÚNIOR, 2012).

As redes sem fio estão se tornando cada vez mais populares pela sua facilidade de instalação/configuração. Com o avanço da rede sem fio foi permitido disponibilizar rede e acesso à internet rapidamente a ambientes onde há demanda de mobilidade, quando não é possível instalar os cabos tradicionais, quando não existe viabilidade na instalação dos cabos (REIS, 2012).

O desenvolvimento das novas tecnologias aumentou a velocidade de transmissão de dados que contribuiu com a diversificação das possibilidades até ao desenvolvimento de tecnologias para aplicações mais simples assim como o

Bluetooth, com infraestrutura mais simples e baixo consumo energético o que lhe vocaciona a tal desinência (BOMFIM; GOMES, 2003).

2.9.1 Bluetooth

Bluetooth é um padrão de comunicação sem fio de curto alcance, baixo custo e baixo consumo de energia que utiliza tecnologia de rádio. Embora tenha sido imaginada como uma tecnologia para substituir cabos pela Ericsson em 1994, Bluetooth tem se tornado largamente utilizado em inúmeros dispositivos e já representa uma parcela significativa do mercado wireless. Dentre os dispositivos que utilizam Bluetooth pode-se incluir os dispositivos inteligentes, como PDAs, telefones celulares, PCs, periféricos, como mouses, teclados, joysticks, câmeras digitais, impressoras e dispositivos embarcados, como os utilizados em automóveis (SIQUEIRA, 2006).

Seu nome é dado em referência ao rei da Dinamarca Harald Bluetooth, um grande diplomata que unificou os povos nórdicos na Europa no século X. E assim é o Bluetooth, uma ferramenta que permite a comunicação unificada de diferentes dispositivos das mais variadas marcas.

Os dispositivos Bluetooth se comunicam entre si e formam uma rede denominada *piconet*, na qual podem existir até oito dispositivos interligados, sendo um deles o mestre (*master*) e os outros dispositivos escravos (*slave*); uma rede formada por diversos "*masters*" (com um número máximo de 10) pode ser obtida para maximizar o número de conexões. A banda é dividida em 79 portadoras espaçadas de 1 Megahertz, portanto cada dispositivo pode transmitir em 79 diferentes frequências; para minimizar as interferências, o dispositivo "*master*", depois de sincronizado, pode mudar as frequências de transmissão dos seus "*slaves*" por até 1600 vezes por segundo. Em relação à sua velocidade pode chegar a 3 Mbps em modo de transferência de dados melhorada e possui três canais de voz (BONATTO; CANTO, 2014).

2.9.2 Rede Móvel

Junto com o surgimento da tecnologia de telefones móveis surgiu à primeira rede móvel de comunicação, totalmente analógica, susceptível a ruídos e

interferências, não possuindo nenhuma criptografia facilitando que as ligações fossem facilmente interceptadas. E o único serviço existente era o de voz.

Já as segundas gerações visam à normalização da rede, tornando a comunicação criptografada, totalmente digital, permitindo ligações telefônicas em roaming e transferência de dados. Dentro do pacote de normalização que emergiu o GSM, que visa à normalização global do sistema de comunicação de telefones móveis.

Entretanto com a difusão da internet a transferência de arquivos multimídias, revelou assim a limitação do sistema GSM, que utilizava de uma transmissão de dados de 9,6 Kbps sendo impossível o compartilhamento de arquivos.

A partir disso houve o surgimento de novas normatizações que permitem uma taxa de transferência de dados elevada, que é o caso do 3G e 4G. Oferecem um conforto de rede banda larga e um meio móvel com serviços de *broadcasting* como YouTube, mensagens multimídia, vídeo chat e os serviços básicos como voz e dados.

Estes serviços são prestados pelas operadoras de telefonia móvel, possuem como fundamentos sempre a qualidade do alcance do sinal, a garantia da troca de pacotes no ambiente IP, e a capacidade de suportar toda a demanda dos usuários existente.

2.9.3 Wi-fi

O seu nome é oriundo de uma abreviação do termo em inglês *"Wireless Fidelity"* que é muito confundido como um termo Wireless. A entidade responsável pelo desenvolvimento da tecnologia Wi-Fi Alliance não reconhece tal alusão descabida (GETO, 2016).

A conexão Wi-Fi e representada por todo tipo de conexão que obedece ao padrão IEEE 802.11 e todas as suas variantes. Basicamente esse e o padrão que foi definido para que as conexões de internet fossem possíveis pelos dispositivos. A conexão através da Wi-Fi acontece a partir de um ponto onde existe uma conexão com a internet tradicional, cabeada, e esse ponto e conectado a um transmissor que envia um sinal de internet pelo ar em determinado raio de efetividade. A difusão desse sinal pode ser feita de forma aberta ou fechada com o uso de senhas ou endereços físicos também conhecidos por MAC para o acesso (GETO, 2016).

Está tecnologia é implementada em larga escala em dispositivos móveis ou não (computadores de mesa, notebooks, PDAs, aparelhos celulares e outros) que tenha certa proximidade, ou melhor, que estejam dentro do seu raio de alcance que é algo em torno de 100 a 300 metros classificando-a como uma WLAN (ALECRIM, 2008).

Junto ao Bluetooth o Wi-Fi é amplamente utilizado e por conta disso muito popular, além disso, possui inúmeras vantagens para implementação como: Permite criar redes locais sem fios para dispositivos clientes, com redução sensível dos custos de implantação e expansão, uma grande flexibilidade para instalação e uso podem ser utilizados no mais diversos ambientes em substituição ou complementação às redes cabeadas (CONCEIÇÃO JÚNIOR, 2012).

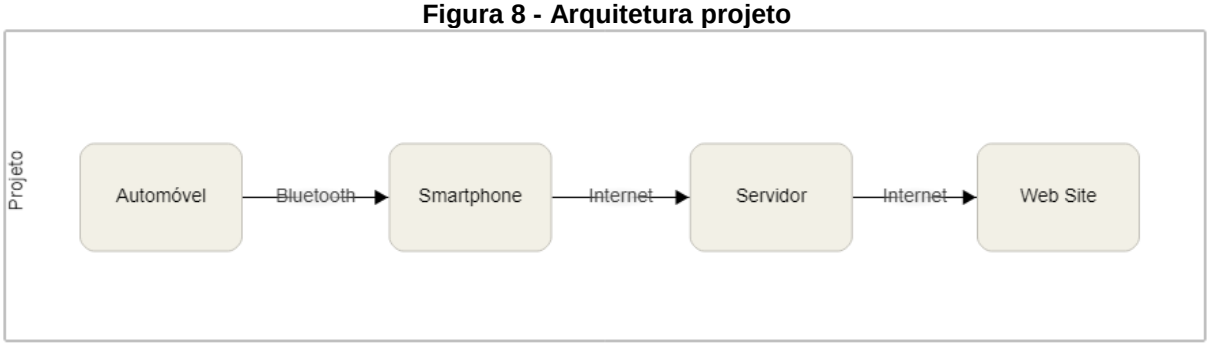
O baixo custo na aquisição do chip set que só tem a diminuir devido a sua crescente utilização, muito confiável com dispositivos de segurança que desde 2007, usa o WPA como sistemas de segurança que não consegue ser facilmente quebrado se forem usadas senhas fortes elaboradas, a constante preocupação desenvolvimento com segurança e possui uma nova encriptação WPA2 que não possui vulnerabilidades conhecidas (ALECRIM, 2008).

Um dos grandes problemas da tecnologia é a não uniformidade das frequências de utilização o que estabelece certa discrepância na utilização de canais adicionais e a autorização para uso em outros, consumo de energia alto em relação ao Zigbee e ao Bluetooth, uma poluição excessiva por ter muitos pontos de acessos com outros pontos em uma mesa área, uma alta taxa de ruído o que pode lhe proporcionar uma grande interferência entre dispositivos que utilizam a mesma faixa de 2,4GHz (SILVEIRA FILHO, 2007).

3 DESENVOLVIMENTO DO TEMA

O projeto aborda o desenvolvimento de um sistema que fornece uma leitura de sensores e falhas automotivas, a longa distância utilizando tecnologias conhecidas e de baixo custo. A arquitetura do projeto baseia-se na estrutura apresentada na Figura 8.

Figura 8 - Arquitetura projeto



Fonte: Autoria própria.

Pode-se visualizar qual será a sequência que o sistema projetado trabalhará e quais são os meios de comunicação de um componente para outro. Primeiramente necessita-se que a norma OBDll esteja presente no sistema de injeção do automóvel, pois, através deste requisito que será possível fornecer a alimentação dos dados para outros sistemas.

O smartphone será o centro do sistema, através do Bluetooth existente na maioria dos aparelhos, se comunicará com o adaptador OBDll escolhido. A partir da conexão Bluetooth, será desenvolvido um aplicativo que irá executar o processamento dos dados recebidos, enviando através da interface do próprio aplicativo a informação e enviará para um servidor online.

O servidor será responsável por armazenar todas as informações na nuvem, sendo possível ser acessadas de qualquer lugar com conexão com a internet. Será necessário também um domínio de hospedagem para o website que será desenvolvido.

E por último, o website através da interface desta ferramenta que qualquer usuário irá conseguir visualizar os dados do automóvel em qualquer lugar com acesso à internet e requisitar os dados que desejar em tempo real.

3.1 COMPONENTES

Inicialmente o sistema projetado necessitará de dois hardwares. O primeiro será o Adaptador ELM327 apresentado na Figura 9.

Figura 9 - Adaptador ELM327

Fonte: Amazon (2018).

Este adaptador ELM327 irá fazer a comunicação do automóvel com o Smartphone via Bluetooth. O segundo hardware que precisa-se é um Smartphone com sistema operacional Android, sendo qualquer tipo de aparelho que cumpra o requisito de possuir Android e Bluetooth. A partir destes dois componentes já pode-se iniciar o desenvolvimento dos softwares.

Os softwares necessários para o desenvolvimento resumem-se a três tipos: O primeiro é o aplicativo, pois dentro dele será executado todo o processamento das informações. A segunda parte é servidor, que executa o armazenamento das informações na nuvem e fornece a comunicação entre o aplicativo e o website. E por último é o website, ele que permitirá que o usuário visualize as informações e requisite os dados que desejar.

3.2 APLICATIVO ANDROID

A parte central do projeto é o aplicativo, ele que executará a maioria das funcionalidades de todo o sistema: Fornecerá a comunicação entre o Smartphone e o dispositivo ELM327 e executará a comunicação com o servidor através da internet.

O aplicativo é desenvolvido para trabalhar com dois modos de operação, online e off-line. No modo online o aplicativo possui uma comunicação com o servidor do Firebase, onde toda a informação processada é enviada para a nuvem e automaticamente.

Já no modo off-line pode-se utilizar o aplicativo de uma maneira diferente. Onde toda a comunicação do aplicativo com o automóvel é possível através dos botões desenvolvidos na tela do smartphone, entretanto só pode ser utilizada dentro do alcance do Bluetooth.

Dentre todas as informações que podem ser obtidas do automóvel através da norma OBD2, delimitou-se algumas utilizadas de exemplo. São elas: a) Temperatura do motor; b) Pressão no coletor de admissão; c) Velocidade do veículo; d) Rotação por minuto do motor; e) Ler código de falhas; e f) Apagar falhas.

Para o desenvolvimento do aplicativo utilizaou-se o App Inventor2, uma plataforma totalmente online e gratuita. Esta ferramenta possui inúmeros componentes, para a fabricação do aplicativo que alguns serão explorados ao decorrer do projeto.

3.2.1 Designer

Quando o *App Inventor2* é iniciado o primeiro passo é configurar a interface gráfica do aplicativo através da página Designer. Inicialmente insere-se os componentes necessários e monta-se a interface conforme a Figura 10.

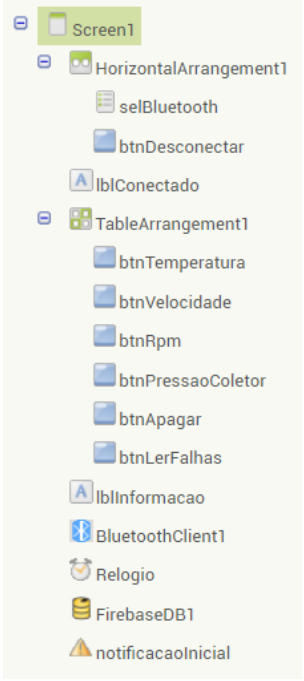
Figura 10 - Interface gráfica aplicativo

Fonte: Autoria própria.

Utilizou-se na interface seis botões, cada botão possui uma função bem declarada. São dois botões responsáveis pela conexão/desconexão do sistema no Bluetooth, e mais quatro botões que serão utilizados para ativar as funcionalidades desejadas pelo usuário.

Esta será a única tela que o usuário visualizará. Entretanto ainda na página Designer, serão necessários inserir alguns componentes não visíveis, para organizar a parte gráfica e para fazer as funções importantes do aplicativo, conforme a Figura 11.

Figura 11 - Componentes necessários



Fonte: Autoria própria.

Todos os componentes inseridos possuem funções especificas, o “HorizontalArrangement1” serve para organizar os dois botões superiores um ao lado do outro. Já o “selBluetooth” é utilizado para exibir a lista de dispositivos Bluetooth que podem ser pareados com o Smartphone e fazer a conexão com o escolhido. O “btnDesconectar” é utilizado para desconectar o dispositivo Bluetooth. No “lblConectado” exibe o status do Bluetooth se está conectado ou não. Já o “TableArrangement1” é utilizado para organizar os botões que solicitarão as informações desejadas. O “lblInformacao” exibe os dados processados, que são a respostas requisitadas através dos botões.

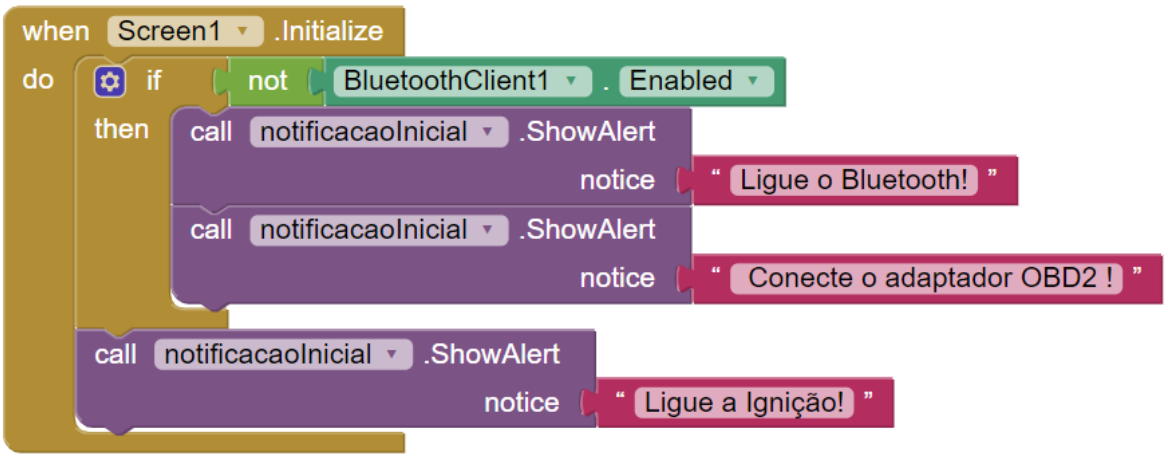
Estes próximos componentes não trabalham na parte gráfica do aplicativo, entretanto são adicionados juntamente com os outros, para assim ser possível utilizar as suas funções futuramente. O “bluetoothCliente1” disponibiliza um pacote da funções que são utilizados na comunicação Bluetooth. No “Relógio” são disponibilizadas as funções relacionadas com tempo. O “FirebaseDB1” libera as funções do *database* do Firebase. E por último a “notificacaoInicial” que libera a disponibilidade de utilizar avisos na tela principal.

3.2.2 Blocks

Nesta página do *App Inventor2* é onde toda a programação é executada, a partir dos componentes inseridos na página Designer as funções são liberadas. Assim pode-see montar a lógica intercalando as funções padrões existentes com as liberadas, através dos blocos de comando.

Quando a aplicação é iniciada é necessário verificar se o Bluetooth está ativado, e caso não esteja avisa-se o usuário para ligar o mesmo. Através do bloco apresentado na Figura 12.

Figura 12 - Screen1.Initialize

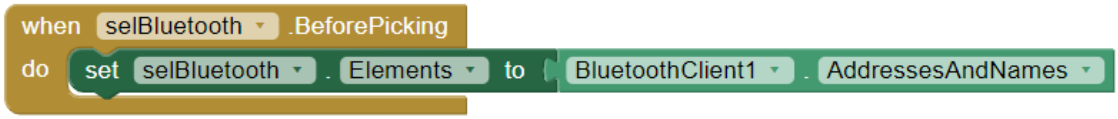


Fonte: Autoria própria.

Neste conjunto de blocos utiliza-se a função “Initialize” do componente “Screen1”, com uma condicional que verifica através de um bloco lógico se o Bluetooth não está ativado e caso não esteja emite um alerta indicando para o usuário ligar o Bluetooth, conectar o adaptador OBD2 e ligar a ignição do automóvel, utilizando a função “ShowAlert” do componente “notificacaoInicial”. Para assim ser possível existir a comunicação com a unidade de comando do automóvel.

Após ser ligado o Bluetooth do smartphone precisa-se conectar com a aplicação, para isso antes do usuário clicar no botão deve-se listar os dispositivos existentes no alcance para após ser feito a seleção. Então os blocos são apresentados como mostra a Figura 13.

Figura 13 - selBluetooth.BeforePicking

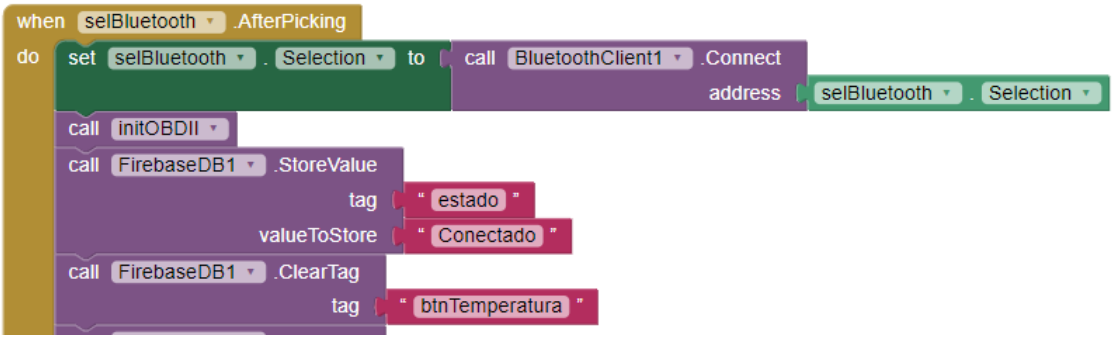


Fonte: Autoria própria.

Este conjunto de blocos utiliza a função “BeforePicking” do componente “selBluetooth” que executa os seus comandos antes do botão ser pressionado. O comando set que foi inserido, altera os elementos da lista “selBluetooth” para os endereços MAC e nomes dos dispositivos existentes no Bluetooth do aparelho. Facilitando assim a visualização do dispositivo que necessita-se conectar.

Após clicar no dispositivo escolhido da lista, executa-se automaticamente a função “AfterPicking”. Que será utilizada para indicar para o aplicativo qual dispositivo da lista foi escolhido (Figura 14).

Figura 14 - selBluetooth.AfterPicking



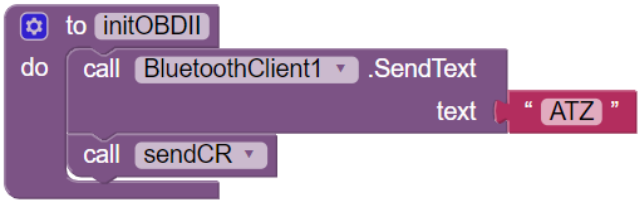
Fonte: Autoria própria.

No primeiro bloco em verde informa-se a aplicação qual dispositivo da lista foi selecionado no componente “BluetoothClient1” anteriormente.

Entretanto pode-se utilizar este evento para acrescentar algumas funções necessárias de outros componentes, como por exemplo, funções do Firebase. Utiliza-se a função “StoreValue” para armazenar no servidor que a aplicação foi conectada com sucesso. Pode-se também zerar o estado de todos os botões que utilizando o website através do método “ClearTag”, onde as informações

relacionadas as *tags* desejadas são removidas. Utiliza-se também neste evento uma chamada de procedimento “initOBDII” que é criada como mostra a Figura 15.

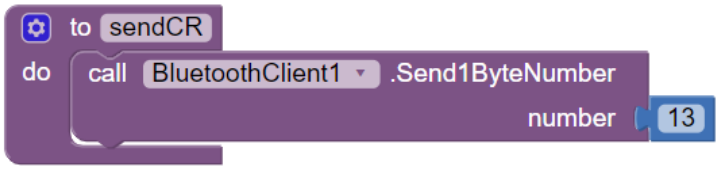
Figura 15 - initOBDII



Fonte: Autoria própria.

O procedimento acima chamado de “initOBDII”, é um procedimento que é utilizado para fazer a primeira comunicação com o ELM327 enviando o comando “ATZ” através da função “SendText“, do componente Bluetooth. Este comando requisita qual é a versão do dispositivo que está sendo utilizada. Entretanto dentro deste procedimento, chama-se outro procedimento conforme apresentado na Figura 16.

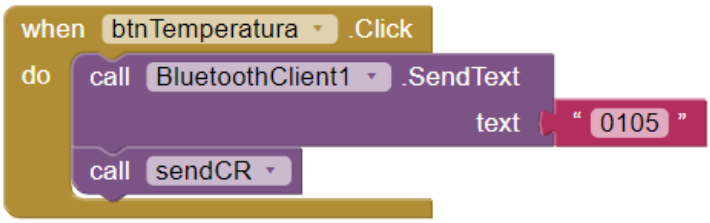
Figura 16 - SendCR



Fonte: Autoria própria.

Este procedimento chamado “sendCR” é utilizado para informar o tamanho dos bytes que estão sendo enviados para o dispositivo ELM327, sendo utilizado o de padrão 13. A sua utilização será necessária em todo o momento em que for enviados comandos ao dispositivo OBDII, principalmente nos botões configurados na interface, como pode-se observar na Figura 17.

Figura 17 - btnTemperatura.Click

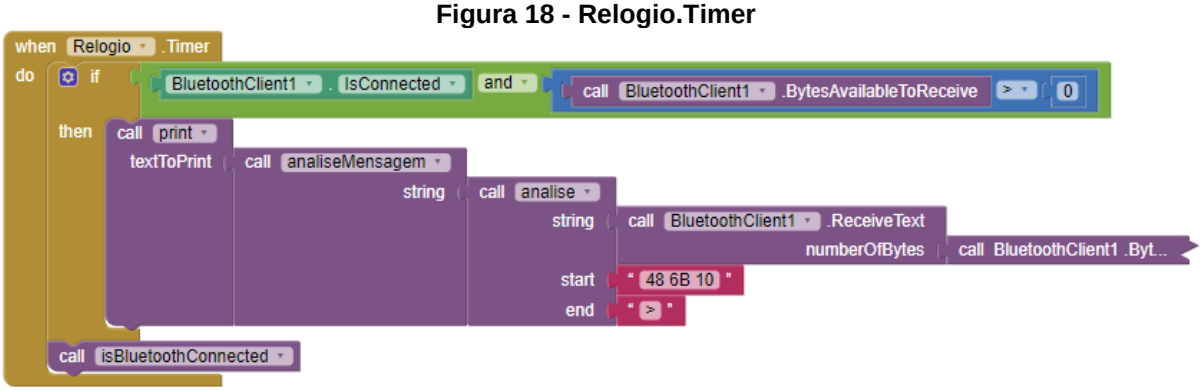


Fonte: Autoria própria.

Utilizou-se como exemplo de botão o da temperatura. Este botão ao ser clicado ele envia o conjunto de texto “0105”, que representa a família de serviço 01 juntamente com o PID 05 responsável pela temperatura, este exemplo é replicado aos demais botões, alterando apenas o PID de cada informação. Em seguida chama-se o “sendCR” novamente conforme citado anteriormente.

Para o recebimento das informações no canal de comunicação do Bluetooth, utiliza-se uma função “Timer” do componente “Relógio”. Que permite repetir todas as funções que estão dentro do seu bloco em uma quantidade determinada de vezes por segundo. Através desta função e funções Bluetooth, será verificado se a aplicação está recebendo dados, como pode-se verificar na sua estrutura apresentada na Figura 18.

Figura 18 - Relogio.Timer

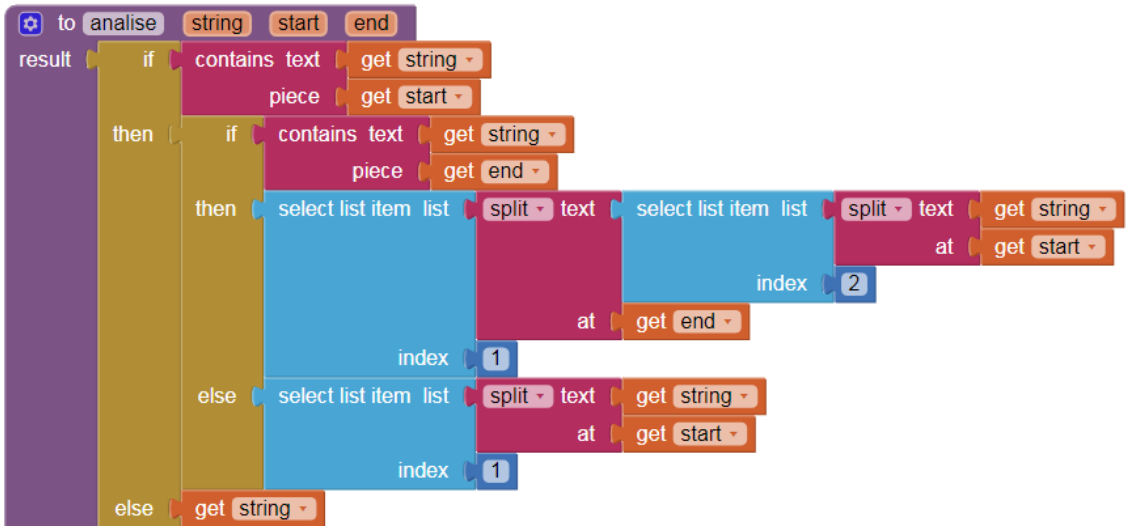


Fonte: Autoria própria.

Dentro desta função, inicia-se comparando com uma função logica “*and”* se o Bluetoooth está conectado e se a quantidade de bytes disponíveis para serem recebidos são maiores que zero. Então um grupo de procedimentos intercalados será chamado para filtrar o pacote de dados recebido.

Através do procedimento “analise” é verificado se a informação não está corrompida, pois este procedimento que determinará onde inicia e aonde termina o pacote de dados. Aceitando apenas mensagens que possuam os caracteres declarados na estrutura inicial “start” e final “end”. Conforme pode-se observar na sua estrutura apresentada na Figura 19.

Figura 19 - Procedimento “analise”



Fonte: Autoria própria.

Inicialmente através de uma função de texto, é verificada se a *string* recebida possui os caracteres iniciais. Em seguida inicia-se uma nova condição para verificar se a *string* recebida também possui caracteres finais, deste modo sabe-se que o pacote de dado não está corrompido. Então remover-se as informações não utilizadas para limpar a *string* dos caracteres determinantes do ”start” e “end”. Através da função de divisão de texto ”slipt text” e listagem indexada ”select list item” deixa-se o corpo da mensagem pronto para ser tratado no próximo procedimento. Conforme a versão do dispositivo ELM327 poderá haver alterações neste procedimento, pois os caracteres determinantes iniciais e finais da resposta podem variar, mas o corpo da mensagem será sempre o mesmo.

Entretanto quando é recebida a resposta após o procedimento “analise” as informações estão preparadas para o próximo procedimento. Estas informações já preparadas possuem a sua estrutura conforme os exemplos da Tabela 5.

Tabela 5 - Exemplos de resposta PIDs

PID	RESPOSTA	INFORMAÇÕES
0105	41 05 46	Sensor Temperatura motor
010C	41 0C 00	Rotação Motor
010D	41 0D 00	Velocidade Automóvel
010B	41 0B 5A	Pressão Coletor Admissão
03	43 01 02 04	Ler Falhas
04	NENHUMA	Apagar Falhas

Fonte: Autoria própria.

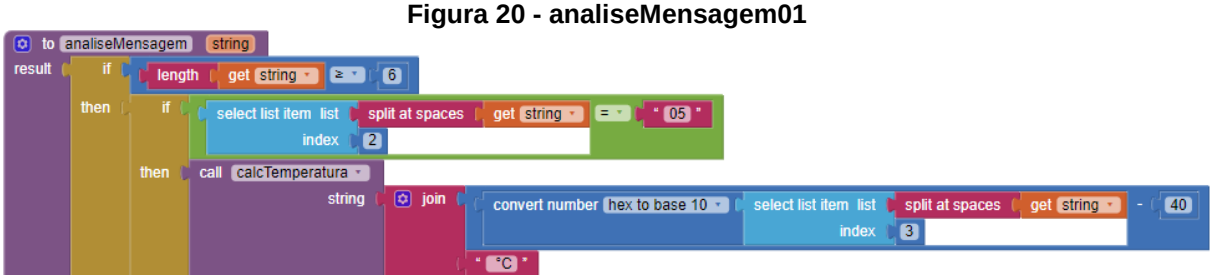
Estes exemplos de resposta nos mostram que informações que não possuírem a quantidade de caracteres maiores ou iguais a seis, por padrão não são válidas.

Ao analisar as respostas, identificou-se um padrão no primeiro caractere, o número “40” está presente em todas as respostas. Está dezena representa um padrão que o modulo do automóvel emite que é somado juntamente com a família de serviço que o PID pertence, formando assim, por exemplo, o número “41” do sensor de temperatura e o número “43” do código de falhas.

O segundo par de dados é o próprio PID que foi enviado, é através dele que será identificado do que a informação trata especificamente.

A partir destas informações monta-ses um conjunto de condicionais que verificam cada pacote de dados, tentando encaixá-las em algum PID. Está tarefa é responsabilidade do procedimento “analiseMensagem”, é ele que classificará e montará o tipo da mensagem recebida, como por exemplo: Temperatura do motor, rotação do motor, DTC e etc., como mostra a Figura 20.

Figura 20 - analiseMensagem01



Fonte: Autoria própria.

Este procedimento inicia verificando o tamanho da *string* produzida no procedimento ”analise”, após identificarmos o seu tamanho iniciamos outra condicional para sabermos qual tipo de informação se trata.

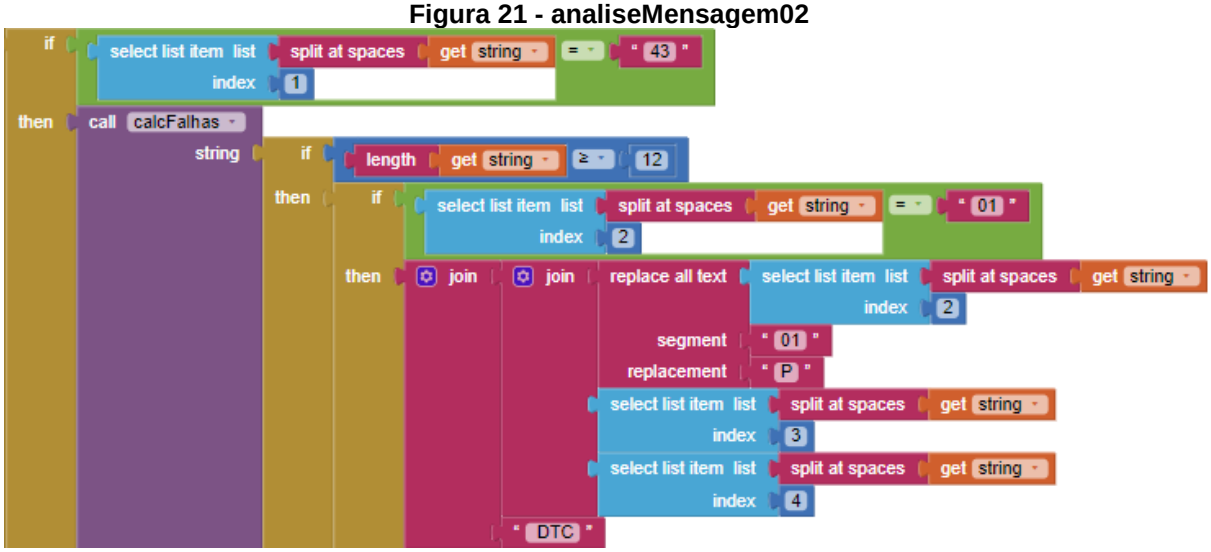
Através da função “Split at spaces” e “select list item” dividimos a *string* em uma lista, e selecionamos o segundo indexador, que é onde o PID da resposta está. Em seguida verificamos se o PID da resposta é igual ao que iremos calcular. Caso haja igualdade entre os PIDs partimos para o próximo passo, onde calculamos a informação. Como exemplo utilizaremos o “calcTemperatura”. Entretanto este método condicional de identificação da resposta é utilizado para todos os PIDs de serviço da família 01.

A partir do momento que o endereço corresponde a o PID que estamos comparando, separamos através da função “Split at spaces” novamente a string da resposta e agora utilizaremos o terceiro par de dados, pois é onde está a informação do sensor que requisitamos. Entretanto em alguns PIDs são retornados dois pares de dados para a resposta, necessitando repetir o procedimento mais uma vez, variando apenas o indexador para o próximo da lista.

Iniciamos o cálculo para processamento da informação aplicando uma função matemática que converte o número hexadecimal para decimal. Em seguida aplicamos uma formula matemática que é tabelada pela norma OBDll, para calcularmos a informação. Após a informação ser calculada adicionamos a unidade de medida através da função de texto “Join” que permite adicionar um texto em uma *string*, formando assim a informação pronta para ser exibida ou armazenada.

Entretanto para montarmos os DTCs utilizamos uma abordagem diferente. Pois utilizou-se a família de PIDs “03” para requisitarmos esta informação. Necessitando de algumas alterações na estrutura da montagem dos DTCs, conforme pode-se observar na Figura 21.

Figura 21 - analiseMensagem02



Fonte: Autoria própria.

Após as condicionais dos PIDs da família 01, montamos uma estrutura de verificação para DTCs. Inicialmente necessitamos identificar na resposta um valor estático para não haver perda de dados. O PID de serviço 03 é único, não possui PIDs complementares então o valor de retorno “43” será unicamente para as respostas de verificação de DTCs.

Para montarmos a estrutura para identificarmos o valor estático, utilizamos o mesmo procedimento que utilizamos para identificar os PIDs da família “01”. Após identificarmos o valor estático, dentro do procedimento “calcFalhas” mensuramos o tamanho da *string* com o valor 12, pois através dos exemplos de resposta que visualizamos anteriormente, sabemos que um DTC necessita no mínimo de 12 caracteres, caso não atinja este tamanho não correspondente a um. Entretanto caso o tamanho seja maior que “12” fazemos uma nova condição para identificarmos *strings* maiores para respectivamente montarmos mais de uma DTC ao mesmo tempo.

Com a classificação do tamanho da *string* executada, precisamos identificar neste procedimento se o DTC existente é relacionado à *powertrain* por exemplo. Com as funções “select list item” e “slipt at spaces” dividimos e comparamos se o item indexado “2” da lista de dados é o número “01”, caso for, através da função “replace all text” substitui-se o par de dados “01” pela letra “P“, que representa o sistema de *powertrain* do automóvel e montamos o restante do DTC selecionado os itens “3” e “4” da *string*, através de uma função de texto chamada “join”. Na Figura 22 pode se observar o antes e depois da montagem da DTC.

Figura 22 - Antes e depois DTC

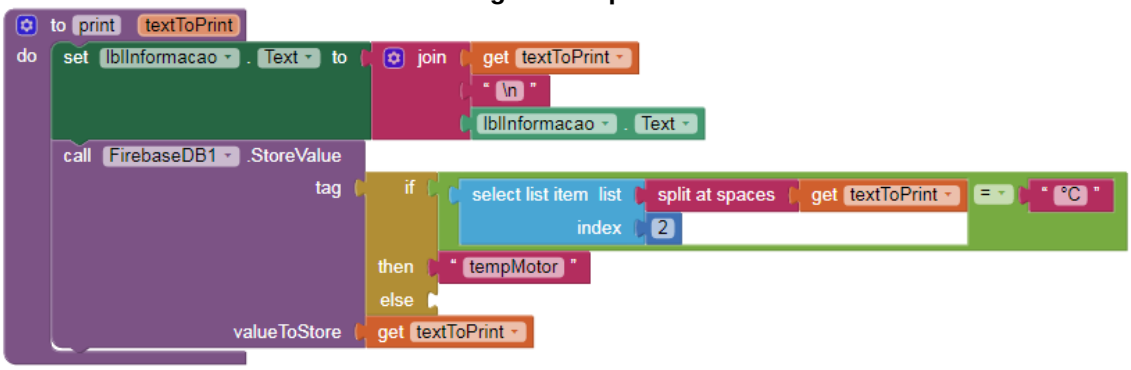
Antes ➔ 43 01 02 04

Depois ➔ *P0204*

Fonte: Autoria própria.

Após todos os procedimentos de montagem dos dados, chegamos a procedimento que classificamos como iremos exibir as informações. Dentro do procedimento “print”, destinamos as informações já tratadas para o seu meio de exibição. Este procedimento é responsável por encaminhar as informações para o servidor ou para o componente “lblInformacao” que existe na tela inicial do aplicativo. O seu bloco possuía a estrutura conforme a Figura 23.

Figura 23 - print



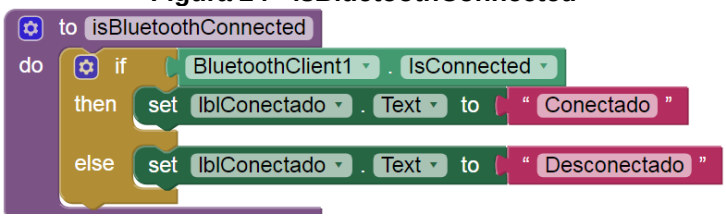
Fonte: Autoria própria.

Este procedimento inicia pegando a variável recebida do procedimento anterior, e envia para um componente “lblInformacao”, alterando a sua propriedade texto. Entretanto para enviarmos para o servidor necessitamos armazenar de uma forma mais organizada, primeiramente achamos uma referência estática dentro da *string*, neste caso utilizaremos a unidade de medida que foi adicionada no bloco anterior, e através desta informação criamos uma *tag* no banco de dados adicionando a *string* dentro.

Este procedimento da identificação da unidade de medida se replica a todas as outras informações, se tornando necessário, montarmos mais um conjunto de condicionais que verificarão cada informação, é adicionará separadamente em respectivas *tags*. Sendo assim o projeto já está enviando e recebendo dados.

Entretanto ainda necessitamos tratar do último procedimento da função “Timer” o “isBluetoothConnected”, este procedimento é utilizado para informar na “lblConectado” da interface o status do funcionamento da aplicação, se está conectada ou desconectada (Figura 24).

Figura 24 - isBluetoothConnected

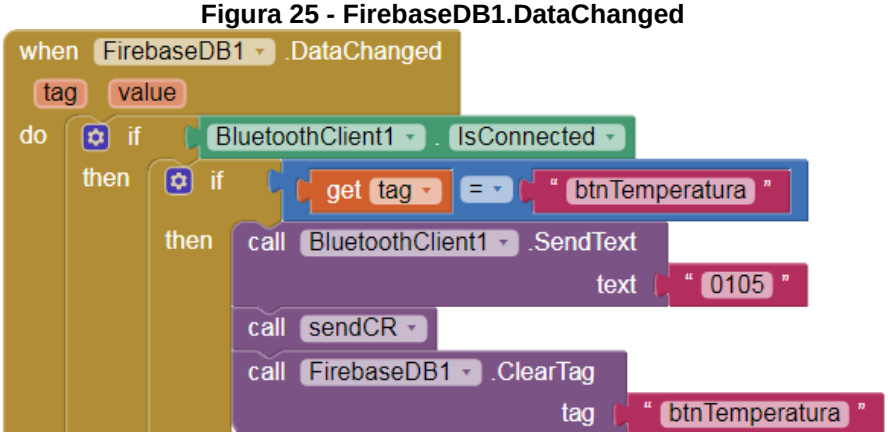


Fonte: Autoria própria.

Verificou-se basicamente neste procedimento se o Bluetooth está conectado. E então se estiver, altera-se o texto da “lblConectado” para Conectado, senão altera-se para Desconectado.

Deste modo concluímos o desenvolvimento da função “Timer” do componente “Relogio”. A próxima função que utilizaremos, é a “DataChanged” do componente “FirebaseDB1”, através desta função que iremos executar a leitura dos botões que estão requisitando dados no website. Esta função é delimitada conforme a Figura 25.

Figura 25 - FirebaseDB1.DataChanged

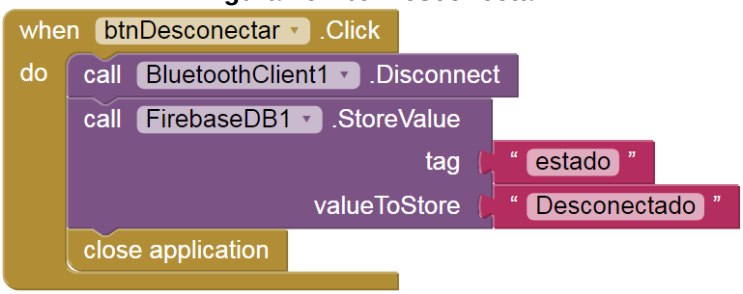


Fonte: Autoria própria.

Quando a aplicação é conectada ao Bluetooth, executa-se a limpeza do estado dos botões, então não existirá nenhum registro de botão no banco de dados. A partir do momento que o Bluetooth está conectado e solicita-se algum dado no website, inclui-se um registro do botão no banco de dados. Então está função irá perceber que ouve alguma mudança e iniciará uma checagem para identificar qual *tag* sofreu o registro, e após a identificação executará o envio do PID referente ao botão solicitado. E por último, executará novamente a limpeza da *tag*, pois caso necessite de um novo requerimento da mesma informação.

A última função que utilizou-se no aplicativo, foi a do componente “btnDesconectar”, está função é executada quando sofre um click em seu botão como apresentado na Figura 26.

Figura 26 - btnDesconectar



Fonte: Autoria própria.

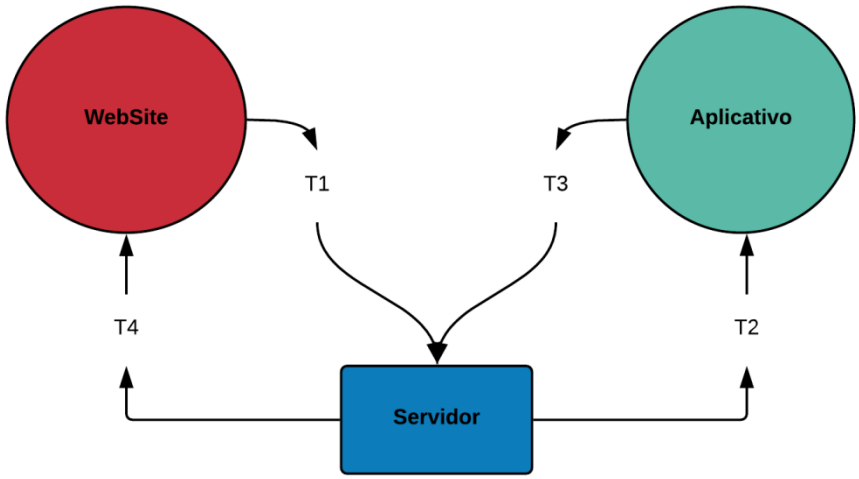
Ao ser clicada, irá desconectar o “BluetoothClien1”. Armazenará no banco de dados o estado “Desconectado” e encerrará o aplicativo através do “close application”.

3.3 SERVIDOR FIREBASE

Para o desenvolvimento completo do projeto necessita-se de um servidor na nuvem. O servidor é necessário para realizar comunicação das duas ferramentas para interligar e hospedar o website que desenvolvido.

Inicialmente necessitamos compreender que as aplicações não se comunicarão diretamente uma com a outra, tudo necessitará passar pelo banco de dados. Então para requisitarmos alguma informação, precisamos enviar a requisição pelo banco de dados. A Figura 27 apresenta um diagrama que exibe os componentes do sistema e as principais tarefas de comunicação do projeto.

Figura 27 - Diagrama comunicação



Fonte: Autoria própria.

- T1: A primeira tarefa é executada através dos botões do website. Basicamente ao ser clicado, o botão envia um valor para a *tag* do respectivo botão no banco de dados.
- T2: A segunda tarefa é executada pelo aplicativo, onde ele faz uma varredura para identificar qual a *tag* no banco de dados sofreu alteração e após a identificação executa o procedimento relacionado à *tag*.

- T3: A terceira tarefa é executada pelo aplicativo, ele processa e encaminha a informação ao banco de dados do servidor. Executando sempre todos os procedimentos programados.
- T4: A quarta tarefa é a atualização do website através das informações que foram armazenadas no servidor pelo aplicativo.

Assim o sistema identifica o que estão sendo pedido, executa as suas funções relacionadas e responde com a informação ao servidor que é respectivamente exibido no website.

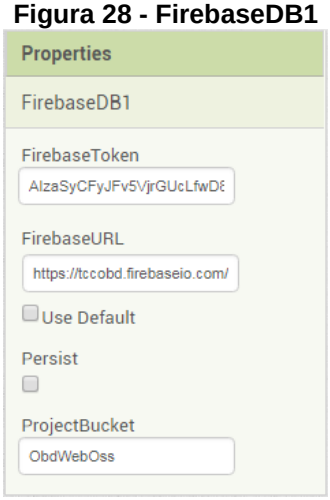
A ferramenta que iremos utilizar para o desenvolvimento do servidor é o Firebase. Existem tarefas no Firebase que vão desde autenticação de usuário até ferramentas de controle de trafego de websites. Entretanto apenas duas tarefas desta ferramenta serão necessárias para o projeto: a) o *Realtime Database*, e b) o *Hosting*.

Inicialmente para a comunicação entre as aplicações utilizaremos o recurso *Realtime Database*. Esta ferramenta é um banco de dados que se comunica com diversas plataformas como, por exemplo, as duas que iremos utilizar: Android e Web. As informações que trafegarem por este banco de dados são acessíveis a todos os usuários que compartilharem da mesma base de dados.

O funcionamento do *Realtime Database* utiliza de sincronização de dados, muito diferente das solicitações HTTP, pois sempre que os dados são alterados na base de dados, as informações são atualizadas automaticamente em todas as aplicações. O armazenamento utiliza o formato JSON, sendo assim a informação se torna mais leve e fácil de trabalhar.

Após a criação do banco de dados, dentro do domínio do Firebase. Necessita-se fazer a ligação das aplicações no banco de dados, para isto foi utilizado o modelo fornecido pela própria ferramenta. No caso do aplicativo é muito simples, insere-se o Token, URL do projeto e o nome da base de dados que se deseja utilizar. Tudo isso dentro das propriedades do componente “FirebaseDB1” no App Inventor 2, como pode ser observado na Figura 28.

Figura 28 - FirebaseDB1



Fonte: Autoria própria.

Todas as informações inseridas neste componente são disponibilizadas nas configurações da página do servidor. A partir desta ligação o aplicativo já estará enviando os seus dados para o servidor.

Entretanto ainda é necessário ligar o website ao banco de dados, como padrão do Firebase será necessário inserir dentro do arquivo Javascript do site o código exibido na Figura 29.

Figura 29 - Database web

```
var config = {
  apiKey: "apiKey",
  authDomain: "projectId.firebaseapp.com",
  databaseURL: "https://databaseName.firebaseio.com",
  storageBucket: "bucket.appspot.com"
};
firebase.initializeApp(config);
var database = firebase.database();
```

Fonte: Firebase (2018).

Este código armazena as informações de endereço do projeto do *Real Database* na SDK do Javascript. O “apiKey” é o Token do projeto, no “authDomain” coloca-se o endereço do domínio da hospedagem, já no “databaseURL” insere-se a URL do banco de dados e por último no “storageBucket” o nome da base de dados URL criada. Após a inserção destes dados, a aplicação web já está endereçada no banco de dados, sendo possível através das funções executarem a leitura e alteração dos dados por meio do código Javascript.

Outra tarefa que utilizada do Firebase é o *Hosting*. Está ferramenta é responsável por armazenar o website e fornecer um domínio totalmente gratuito. Algumas limitações são encontradas dentro do pacote gratuito, dentre elas o acesso

possui um limite de conexões simultâneas e o domínio possui um padrão “firebaseapp.com”.

Com esta ferramenta será conquistada uma conexão segura, de acesso rápido e de simples implementação. Através do painel do Firebase e do Node.js, com poucos segundos consegue-se executar atualizações, implementações e *rollbacks*.

3.4 APLICAÇÃO WEB

O website funcionará como um painel de controle do automóvel. O usuário ao abrir no navegador, possuirá o acesso as informações do veículo que estiver conectado com o aplicativo. Sendo possível executar algumas leituras de sensores, visualizar os códigos de falhas e apagar os códigos de falhas. Na interface também exibirá o estado do aplicativo, se está conectado ou desconectado. Serão cinco campos para visualização de informações e seis botões (Figura 30).

Figura 30 - Layout website

Fonte: Autoria própria.

Através da linguagem HTML desenvolvemos o website, utilizando alguns componentes gráficos para embelezamento e organização dos componentes por meio da folha CSS.

Entretanto as funções mais importantes estão relacionadas com a parte Javascript do código. É através desta linguagem de programação que inserimos as principais funções, como a conexão ao banco de dados, funcionamento dos botões e a leitura das informações armazenadas no banco de dados.

A leitura dos dados do servidor é executada com o código apresentado na Figura 31.

Figura 31 - Busca de dados web

```
  var postElement1 = document.getElementById('postElement1');
  var updatevelocidade = function(element, value) {
    element.textContent = value;
};
  var velocidade = firebase.database().ref('ObdWebOss/velocidade');
velocidade.on('value', function(snapshot) {
  updatevelocidade(postElement1, snapshot.val())
  });
```

Fonte: Autoria própria.

Utilizou-se de exemplo o processo de leitura da informação sobre velocidade. Primeiramente criou-se a variável “postElement” que irá receber a informação, após um identificador HTML é inserido com o mesmo nome da variável. Em seguida criou-se outra variável “updatevelocidade” que é responsável por executar uma função que serve para alterar o valor de texto do elemento, pela a informação recebida.

Posteriormente declarou-se a variável “velocidade” referenciada com o endereço da informação no banco de dados, juntamente com uma função “snapshot” padrão do Firebase, que identifica sempre quando houver uma alteração na informação referenciada.

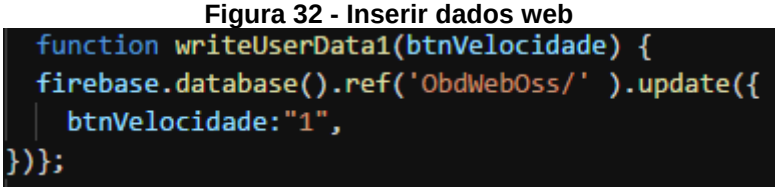
Após chamou-se o “updatevelocidade” que foi criado anteriormente e atualizou-se o “postElement” com o valor da alteração adquirido no “snapshot”. Assim a variável “postElement” carrega a informação, que pode ser chamada pelo seu ID para exibir a informação aonde deseja-se no código HTML.

Para buscar as outras informações executa-se o mesmo procedimento alterando apenas as referências e o nome do identificador.

Nos botões utilizou-se uma função um pouco diferente. Apenas é necessário inserir um dado qualquer na respectiva *tag*, pois o aplicativo irá identificar somente a

tag que sofreu alteração, não tendo importância a variável, como pode ser observado na Figura 32.

Figura 32 - Inserir dados web



Fonte: Autoria própria.

Para inserir o valor na *tag* desejada, cria-se uma função “writeUserData” relacionada com o botão velocidade. Quando acionada, executa a ferramenta referenciada ”update” do Firebase, que atualiza o dado dentro da *tag* declarada interiormente.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Este capítulo apresenta alguns experimentos que serão utilizados para validação do funcionamento do sistema proposto. Inicialmente trataremos da leitura de alguns sensores e após simularemos um código de falha em um automóvel real. Verificaremos também a conexão e comunicação das aplicações desenvolvidas, monitorando o funcionamento de todos os componentes existente no sistema. Entretanto para maior clareza destes experimentos delimitamos alguns requisitos básicos descritos na Tabela 6 que o sistema deverá cumprir.

Tabela 6 - Requisitos funcionais

REQUISITO	DESCRIÇÃO	EXPERIMENTO
R1	O sistema deverá fornecer informações dos sensores de temperatura, pressão admissão, velocidade e rotação do motor.	1
R2	O sistema deverá fornecer informações sobre código de falhas (DTC).	2
R3	O sistema deve enviar dados via Bluetooth e Internet.	1, 2
R4	O servidor deverá se comunicar com as duas aplicações	1, 2

Fonte: Autoria própria.

4.1 EXPERIMENTO 1: LEITURA DOS SENSORES

Introdução: A leitura dos sensores é utilizada para identificarmos a situação do automóvel, auxiliando a verificação de falhas que poderão ocorrer. O proposito deste experimento é comprovar o funcionamento de toda aplicação. Inicialmente verificaremos o funcionamento da comunicação Bluetooth, após verificaremos a integridade das informações processadas pelo aplicativo. Por último verificaremos se a informação foi recebida pelo website corretamente.

Métodos: O experimento é executado em um veículo Montana LS 1.4 2013, que possui a norma OBD2 embarcada no seu sistema de injeção. Executamos a instalação do adaptador OBD2 Bluetooth no automóvel conforme a Figura 33. Após a execução da instalação do aplicativo em um Smartphone Galaxy S6 modelo SM-G9020l. Submete-se o veículo em estado de rodagem em condições urbanas, para verificação das informações, utilizando internet Móvel 4G como meio de comunicação com o servidor Firebase.

Figura 33 - Instalação adaptador

Fonte: Autoria própria.

Resultados: Após uma sequência de testes e requisições, obtemos um resultado muito satisfatório com o sistema, mesmo com o veículo em movimento conseguimos obter os dados requisitados conforme a Figura 34.

Figura 34 - Leitura do aplicativo

Fonte: Autoria própria.

Estes dados visualizados no aplicativo possuem o formato correto e não possuem pacotes perdidos. A partir do momento que possuímos uma informação com integridade, a mesma está sendo enviada para o servidor, através da estrutura de envio de informações que desenvolvemos no aplicativo. Neste momento verificamos o recebimento destes dados no website para comprovar o armazenamento do servidor conforme a Figura 35.

Figura 35 - Leitura do website

Fonte: Autoria própria.

Com o sucesso na exibição do website, concluímos que o armazenamento das informações no servidor está sendo executada corretamente. Após este procedimento executamos requisições através dos botões, obtendo novos dados que comprovam o funcionamento total do website.

4.2 EXPERIMENTO 2: SIMULAÇÃO DTC

Introdução: Os DTCs são códigos que exibem os erros presentes nos sistemas embarcados do automóvel, através destes códigos que os técnicos identificam problemas e solucionam. O propósito deste experimento é testar o funcionamento do sistema, no quesito de ler e apagar DTCs.

Métodos: Utilizaremos os mesmos componentes do ensaio anterior. Entretanto simulamos algum erro no sistema de injeção do automóvel, para conseguirmos executar a leitura do DTC. Neste caso iremos remover o conector do primeiro bico injetor localizado no coletor de admissão conforme a Figura 36.

Figura 36 - Simulação DTC

Fonte: Autoria própria.

A partir da remoção damos partida no motor do veículo, para o sistema armazenar o DTC P0204. Após a luz MIL acender conectamos novamente o bico injetor e submetemos veiculo a situação em estado de rodagem.

Resultados: Logo após iniciarmos o procedimento de teste executamos a leitura dos sensores novamente através dos botões do website, e após verificamos a memória de erros remotamente e visualizamos o erro armazenado conforme a Figura 37.

Figura 37 - Website DTC

Fonte: Autoria própria.

Após a constatação do DTC presente no sistema, executamos a limpeza da memória, através do botão “Apagar Falhas”.

4.3 DISCUSSÃO GERAL

Os exemplos utilizados permitiram ter uma ideia da funcionalidade de todo o sistema, que apesar de serem simples comprovam a eficácia da aplicação. Os experimentos foram todos executados em campos urbanos com a cobertura de internet móvel 3G e 4G.

Entretanto em ambientes que não possuírem cobertura móvel é possível a utilização de redes Wifi. Em casos de falta de acesso à internet o sistema online será atualizado quando retornar a conexão com a última informação cadastrada exibida no smartphone.

As funcionalidades do sistema não estão limitadas apenas na comunicação com o servidor. Não impedindo assim o funcionamento off-line do sistema, que seria a execução das leituras somente através do aplicativo dentro do alcance Bluetooth.

Os requisitos delimitados no início do capitulo foram todos compridos, dentro de suas limitações. Entretanto dão um parecer básico das funcionalidades do sistema como um todo.

5 CONSIDERAÇÕES FINAIS

Através de questões levantadas a respeito do futuro da comunicação dos sistemas automotivos, surgiu a motivação de desenvolvermos um sistema que proporcionasse uma leitura das informações dos sistemas embarcados do automóvel por meio da internet. Com o direcionamento adquirido através do embasamento teórico delimitamos as principais especificações e requisitos, que foi dividido em duas funções básicas: leitura de sensores e leitura de código de falhas, ambas descritas e estruturadas. Em seguida foram submetidos testes que a fim de demonstrar a validação e comprimentos dos requisitos delimitados ao longo do projeto.

O sistema desenvolvido cumpre os objetivos declarados inicialmente, a saber: desenvolver um aplicativo que execute uma comunicação Bluetooth com o automóvel, projetar um *web service* que receba e armazene as informações recebidas, implementar um website que recebas as informações do servidor. Entretanto por depender principalmente de rede móvel para a comunicação da da aplicação, existem problemas que são acarretados devido a qualidade do sinal e o alcance da cobertura do sinal. Que muitas vezes até em centros urbanos existem perdas de pacotes de dados.

Pelo fato do sistema ser um protótipo encontramos algumas limitações. Primeiramente, o sistema só pode ser executado apenas em um veículo por vez, dificultando assim testes e comparações simultâneas. Outra limitação é a respeito de segurança, onde não foram implementadas nenhuma criptografia ou senha para acesso às informações, tornando-as totalmente vulneráveis.

A partir das limitações citadas, podemos futuramente adotar algumas melhorias em todo o sistema: Aprimorando políticas de segurança através de alguma criptografia, estruturarmos um sistema de cadastro com devido *login* e senha utilizando ferramentas fornecidas pelo Firebase e podemos também executar atualizações a respeito de incorporações de novos veículos ao sistema. Executando a leitura de dois ou mais veículos ao mesmo tempo, juntamente com sistema de posicionamento global (GPS).

REFERÊNCIAS

ALECRIM, Emerson. **O que é wi-fi (IEEE 802.11).** INFO WESTER, publicado em: 19 mar. 2008. Disponível em: <http://www.infowester.com/wifi.php>. Acesso em: 04 out. 2018.

ALMEIDA, Eduardo Luciano de; FARIA, Felipe Freitas de. **Scanner OBD-II em plataforma LabView.** Trabalho de Conclusão de Curso (Graduação), Faculdade de Tecnologia (FATEC), Tecnologia em Eletrônica Automotiva, Santo André, SP, 2013. Disponível em: <http://fatecsantoandre.edu.br/arquivos/TCC232.pdf>. Acesso em: 17 set. 2018.

AMAZON. **ELM327**. Copyright© 1996-2018, Amazon.com, Inc, 2018. Disponível em: <https://www.amazon.co.uk/Version-Bluetooth-Diagnostic-Multi-Language-12Kinds/dp/B0136BJ0XC>. Acesso em: 11 out. 2018.

ARAÚJO, Bruno Faria de. **Características dos sensores de temperatura NTC e PTC**. Copyright© 2016 ADD-THERM, post publicado em: 12 jan. 2017.Disponível em: <http://www.addtherm.com.br/2017/01/12/sensores-de-temperatura-ntc-e-ptc/>. Acesso em: 27 nov. 2018.

BATSCHINSKI, George. **Backend as a service: Prós e contras.** Copyright© 2006-2016 C4Media Inc, post publicado em: 05 jul. 2016. Disponível em: <https://www.infoq.com/br/news/2016/07/backend-pros-e-contras>. Acesso em: 02 out. 2018.

BLAKE, Daniel. **Embedded systems and vehicle innovation**. Copyright© 2018 SAE International, publicado em: jan. 2005. Disponível em: <http://www.aerospacestandards.com/automag/features/futurelook/01-2005/1-113-1-86.pdf>. Acesso em: 10 out. 2018.

BOMFIM, Alex Nério de Andrade; GOMES, Pedro Rodrigues. **Redes sem fio (wireless)**. Universidade Católica do Salvador (UCSal), Salvador, BA, trabalho de disciplina, 2003. Disponível em: <http://www.logicengenharia.com.br/mcamara/ALUNOS/Wireless.pdf>. Acesso em: 04 out. 2018.

BONATTO, Aurélio; CANTO, Diego Oliveira do. **Bluetooth Technology (IEEE 802.15)**. Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Faculdade de Administração, Contábeis e Economia, Trabalho da disciplina de Redes e Sistemas Distribuídos, 2014. Disponível em: <https://www.inf.pucrs.br/~cnunes/redes/Trabalho%20Bluetooth.pdf>. Acesso em: 04 out. 2018.

CHADHA, Karishma. **Improving the usability of App Inventor through conversion between blocks and text.** Honors Thesis Collection, 25 abr. 2014. Disponível em: <https://repository.wellesley.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1290&context=thesiscollection>. Acesso em: 30 set. 2018.

CLARK, Andrew. **App Inventor launches second iteration**. Massachusetts Institute of Technology, Cambridge, 30 dez. 2013. Disponível em: <http://news.mit.edu/2013/app-inventor-launches-second-iteration>. Acesso em: 01 out. 2018.

CLARO, Daniela Barreiro; SOBRAl, João Bosco Mangueira**. Programação em Java**. Copyleft Pearson Education. Florianópolis, SC, 2008. Disponível em: <http://www.faeterj-rio.edu.br/downloads/bbv/0031.pdf>. Acesso em: 26 set. 2018.

CONCEIÇÃO JÚNIOR, André Lisboa. **Redes sem Fio: Protocolo bluetooth aplicado em interconexão entre dispositivos.** Copyright© 2018 Teleco, tutorial publicado em: 29 out. 2012. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialredespbaid/default.asp>. Acesso em: 04 out. 2018.

CONTESINI, Leonardo. **Qual foi o primeiro carro "injetado" da história?** Copyright© FlatOut Brasil, publicado em: 30 mai. 2015. Disponível em: <https://www.flatout.com.br/qual-foi-o-primeiro-carro-injetado-da-historia/>. Acesso em: 27 nov. 2018.

DIAS, Anderson. **Injeção eletrônica: Sensor de rotação, vital para o motor.** Carros Infoco, Post publicado em: jul. 2012. Disponível em: <http://www.carrosinfoco.com.br/carros/2012/07/injecao-eletronica-sensor-de-rotacao-vital-para-o-motor/>. Acesso em: 27 nov. 2018.

DOUTORCARRO. **Códigos de Erro OBD-II de carroceria: B0000 a B2606.** Copyright© 2018 Doutor Carro, post publicado em: 15 jul. 2014. Disponível em: <https://www.doutorcarro.com.br/codigos-de-erro-obd-ii-de-bordo-b0000-a-b2606/>. Acesso em: 17 set. 2018.

ELM327. **ELM327: OBD to RS232 Interpreter.** Copyright© 2005-2017 Elm Electronics Inc, 2017. Disponível em: <https://www.elmelectronics.com/wp-content/uploads/2017/01/ELM327DS.pdf>. Acesso em: 20 set. 2018.

FIREBASE. **Documentação: Firebase por plataforma.** 2018. Disponível em: <https://firebase.google.com/docs/>. Acesso em: 02 out. 2018.

GETO, Daniel. **Saiba a diferença que existe entre wi-fi e wireless**. Copyright© 2017 Menos Fios, post publicado em: 18 nov. 2016. Disponível em: <https://www.menosfios.com/saiba-as-diferencas-entre-wifi-e-wireless/>. Acesso em: 04 out. 2018.

GONÇALVES, Eduardo Corrêa. **JSON tutorial**. DEVMEDIA, artigo publicado em: 2012. Disponível em: <https://www.devmedia.com.br/json-tutorial/25275>. Acesso em: 02 out. 2018.

GONÇALVEZ, Ariane. **O que é CSS? Aprenda sobre CSS com este guia básico.** Copyright© 2004-2018 Hostinger, post publicado em: 2018. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/#gref>. Acesso em: 03 out. 2018.

GRILLO, Filipe Del Nero; FORTES, Renata Pontin de Mattos. **Aprendendo JavaScript**. Universidade de São Paulo (USP), São Carlos, 21 fev. 2008. Disponível em: <http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_ND_72.pdf>. Acesso em: 03 out. 2018.

GUIMARÃES, Alexandre de Almeida. **Eletrônica embarcada automotiva**. São Paulo: Érica, 2011. 326p.

LEE, Nicole. **The 411: Feature phones vs. smartphones.** Copyright© CBS INTERACTIVE INC, post publicado em: 01 mar. 2010. Disponível em: <https://www.cnet.com/news/the-411-feature-phones-vs-smartphones/>. Acesso em: 25 set. 2018.

MEYER, Maximiliano. **A história do Android.** Copyright© 2018 M3 Mídia, post publicado em: 06 mai. 2015. Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-android>. Acesso em: 26 set. 2018.

MORIMOTO, Carlos. E. **Smartphones: Guia prático.** GDH Press e Sul Editores, 2009. 431 p.

PAPAIOANNOU, Iannis, Nicolaos. **Estudo da eletrônica embarcada automotiva e sua situação atual no Brasil**. Trabalho de Conclusão de Curso (Mestrado Profissionalizante em Engenharia Automotiva). Escola Politécnica da Universidade de São Paulo, São Paulo, SP, 2005. 89 p. Disponível em: <http://automotiva-poliusp.org.br/wp-content/uploads/2013/02/papaiounnou_iannis.pdf>. Acesso em: 23 ago. 2018.

REIS, Gustavo Henrique da Rocha. **Redes sem fio**. Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Campus Rio Pomba, Disciplina Redes de Computadores, 2012. Disponível em: <http://www.riopomba.ifsudestemg.edu.br/dcc/dcc/materiais/1042283583_redes-sem-fio.pdf >. Acesso em: 03 out. 2018.

SANTOS, Max Mauro Dias. **Redes de comunicação automotiva: Características, tecnologias e aplicações.** São Paulo: Érica, 2010. 220 p.

SILVA, Alexandre Roberto da. **O que é Node.js?** Copyright© 2018 BABOO, post publicado em: 06 mai. 2012. Disponível em: <https://www.baboo.com.br/arquivo/software/o-que-e-node-js/>. Acesso em: 02 out. 2018.

SILVA, José Roberto Batista da. **Tratamento de veículos em fim de vida: Modelos de gestão internacionais e brasileiro.** Dissertação (mestrado profissional) – Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia Ambiental. Florianópolis, SC, 2016. 116p. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/172181/342649.pdf?sequence=1>. Acesso em: 23 ago. 2018.

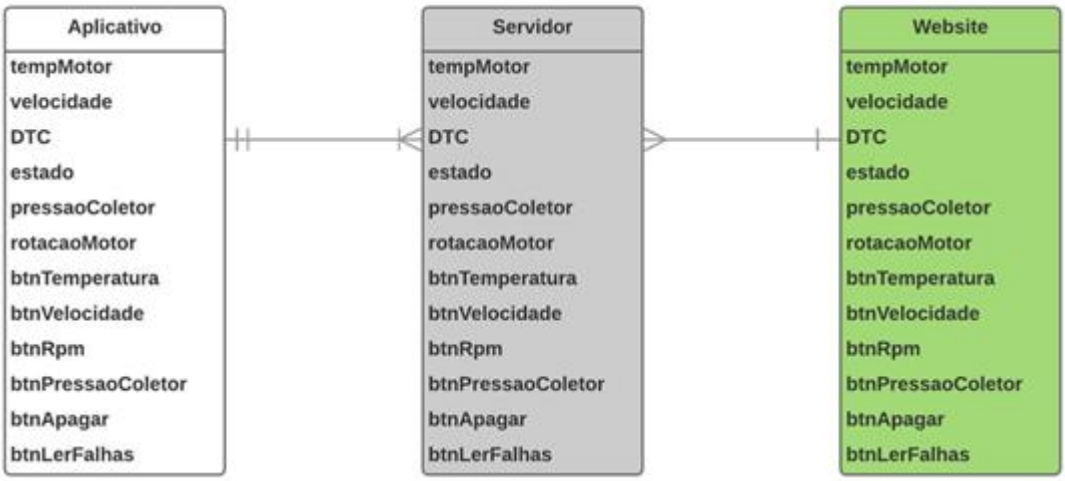
SILVEIRA FILHO, Otton Teixeira da. **Redes: Uma introdução**. Material didádito de Informática I, publicado em: 20 jul. 2007. Disponível em: <http://www.ic.uff.br/~otton/graduacao/informatical/redes.pdf>. Acesso em: 04 out. 2018.

SIQUEIRA,Thiago Senador de. **Bluetooth: Características, protocolos e funcionamento.** Universidade Estadual de Campinas, Instituto de Computação, Trabalho da Disciplina de Arquitetura de Computadores I, 2006. Disponível em: <http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf>. Acesso em: 03 out. 18.

THEOHARIDOU, Marianthi; MYLONAS, Alexios; GRITZALIS, Dimitris**. A risk of assessment method for smartphones**. Athens: Athens University of Economics and Business (AUEB). Copyright© IFIP International Federation for Information Processing, 2012. P. 443-456. Disponível em: <http://www2.aueb.gr/users/amylonas/docs/SEC-12SmartphoneRiATechRe.pdf>. Acesso em: 25 set. 2018.

APÊNDICES

APÊNDICE A: DIAGRAMA ENTIDADE E RELACIONAMENTO



APÊNDICE B: DIAGRAMA FUNCIONAMENTO DO SISTEMA