Sensoriamento Automobilístico Usando Redes Sem Fio e Alerta de Falhas por Voz Sintetizada

Janine Kneis
Universidade do Estado de Santa Catarina - UDESC
Joinville, Brasil
janine.knies@udesc.br

Maicol Peterson Gandolphi de Almeida
Universidade do Estado de Santa Catarina - UDESC
Joinville, Brasil
gandolphi@hotmail.com

ABSTRACT

This paper proposes a sensing system for vehicles using an automotive scanner with wireless connection for sending data. The connection to the scanner provides data that is sent to a cell phone via a Bluetooth connection and to the cloud . In order to provide a response to the driver as soon as an abnormality appears, an application was created to emit a synthesized voice warning. The results showed that the driver, through the synthesized voice messages and the system interface, was previously able to become aware of the abnormalities of the vehicle and carry out preventive maintenance.

KEYWORDS

Sensoriamento, Veículos, Redes sem fio, Síntese de voz

1 INTRODUÇÃO

Internet das Coisas (do inglês *Internet of Things* - IoT) é um conceito que visa incluir conectividade sem fio em dispositivos do dia a dia, a fim de permitir muitas formas de aplicativos inteligentes [9]. Tais dispositivos possuem capacidades de processamento e comunicação, como por exemplo, etiquetas de identificação por rádio frequência, sensores e atuadores, além de poder interagir uns com os outros a fim de alcançar objetivos comuns [5].

A Internet das Coisas tem se mostrado promissora em diversas áreas, como meio ambiente, saúde, energia e na indústria automobilística. No setor automotivo, a IoT possibilita, por exemplo, captar dados em tempo real dos sensores dos veículos e emitir alertas preventivos para os motoristas.

Um passo importante para a evolução da eletrônica automobilística foi o surgimento da injeção eletrônica em 1957 pela Bendix Corporation [13]. Os avanços ocorridos nas últimas décadas permitem que os usuários acompanhem através dos painés veiculares o funcionamento dos sistemas dos seus veículos, e obtenham um melhor controle da quantidade de combustível enviada ao motor.

A evolução da eletrônica automobilística vem auxiliando os usuários a acompanharem o funcionamento dos sistemas dos seus veículos, além de controlar melhor a quantidade de combustível enviada ao motor.

Após a década de 70 os limites para emissões de gases veiculares tornaram-se mais rigorosos e a Sociedade de Engenheiros Automotivos (SAE) definiu um conector padrão para o painel elétrico, além de um conjunto de sinais de testes para diagnósticos conhecido como *On-Board Diagnostics* (OBD) [1].

A partir de 1996 foi criado o conector OBD2 com o propósito de informar dados do motor, chassi, carroceria e outros acessórios do veículo, seguindo normas da SAE [6].

Os autores [16] e [6] apresentam os vários protocolos e serviços de diagnósticos do OBD2, bem como uma lista de aparelhos vendidos no mercado. O conector OBD2 segue algumas especificações como: estar posicionado a 300 mm da unidade central de processamento *Electronic Control Unit* (ECU) e ser de fácil acesso pelo motorista. A pinagem do conector é mostrada na Figura 1 e alguns pinos, referem-se ao protocolo usado pelo fabricante do automóvel [16].

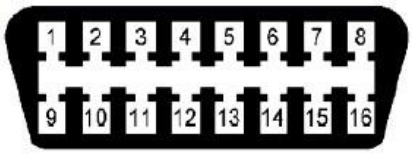

Figure 1: Conector OBD2 e Pinagem.

Através do conector OBD2 é possível realizar diferentes diagnósticos do veículo, por exemplo, na Figura 1 destacam-se, o código PID 01 que é usado para adquirir o valor da última leitura de um sensor e o PID 03, usado para identificar algum problema presente no veículo.

Neste trabalho apresenta-se um sistema de IoT que provê uma interface de comunicação sem fio com um conector OBD2 e automatiza o processo de leitura de dados dos sensores disponíveis nos veículos.

O sistema proposto também usa os diagnósticos de erros do protocolo OBD2 para síntese de voz e avisa preventivamente o motorista sobre problemas do veículo, além de informar o motorista a possível causa do problema. O objetivo é facilitar a manutenção preventiva do veículo e auxiliar na redução do número de acidentes e emissão de gases poluentes.

Este trabalho está organizado como segue. Na Seção 2 são apresentados os trabalhos relacionados. Na Seção 3, apresenta-se a abordagem proposta para sensoriamento automobilístico. Na Seção 4, discute-se os resultados encontrados com o sistema desenvolvido. Na Seção 5 são apresentadas as conclusões e os trabalhos futuros.

2 TRABALHOS RELACIONADOS

HUSNI et al. (2016) [7] desenvolveu um sistema de carro inteligente denominado *Smart Car* que monitora e relata a condição do automóvel para prever manutenções de rotina e fornecer informações para melhorias na forma de dirigir. Os parâmetros monitorados do *Smart Car* foram carregados em um servidor na nuvem utilizando aplicativos IBM BlueMix a fim de obter informações para

manutenção preditiva e monitoramento. O trabalho de HUSNI também destacou a presença de um analista de Big Data para acompanhar os dados na nuvem e informar qualquer problema ao motorista, proprietário ou aos técnicos de oficina mecânica.

RODRÍGUEZ et al. (2018) [14] implementou um escâner OBDII em um barramento CAN usando a plataforma Arduíno, monitorando os dados em um PC através do software LabVIEW. O sistema desenvolvido foi dividido em duas partes: a etapa de coleta de dados (escâner) e a interface gráfica para visualização e monitoramento de diversos parâmetros. Um diferencial do trabalho de Rodríguez [14] foi aplicar um módulo CAN Bus para Arduíno com o objetivo de simular a ECU de um veículo e seus parâmetros sem precisar de um veículo real.

GONZÁLEZ et al. (2019) [4] desenvolveu um projeto semelhante ao de Rodríguez [14], porém usou uma conexão Bluetooth num automóvel elétrico. Neste projeto foi desenvolvido um programa chamado Emolab 2.0.1 para visualização dos dados provenientes da ECU. Este programa permite a leitura de 74 variáveis com suas unidades de medida, além disso os estados das tensões das células podem ser observados graficamente. O Emolab 2.0.1 grava as coordenadas geográficas do automóvel e vários parâmetros para uma coordenada específica, desta forma é possível carregar os dados no Google Maps de maneira *offline* e visualizar a trajetória realizada. Clicando em uma coordenada que o veículo percorreu é possível abrir uma lista contendo vários parâmetros do automóvel naquele ponto [4].

SHAFI et al. (2018) [15] focaram seus estudos em aprendizagem de máquina e usaram dados dos sensores de um automóvel Corolla fabricado pela Toyota. O trabalho teve como foco a seleção de sensores que podiam causar um colapso no sistema em caso de falhas ou anormalidades. Os dados dos sensores foram usados para previsão de falhas usando técnicas de Inteligência Artificial e transmitidos na forma de códigos de erro DTC para um smartfone através de um escâner OBD2 com conexão Bluetooth com o veículo em movimento [15].

O sistema proposto neste artigo se destaca na questão da emissão de mensagens por voz sintetizada que explica ao motorista a possível origem da falha. Alguns aplicativos vendidos na Google Store por exemplo (play.google.com/store) apenas mostram o código do erro e emitem um alerta sonoro ou uma frase de aviso ao motorista indicando que determinado valor atingiu seu estado crítico, porém não informam a origem do problema, como uma mangueira rompida ou um filtro danificado.

Nos trabalhos relacionados, verificou-se que os dados dos sensores são armazenados na nuvem computacional para posterior consulta, entretando o motorista não toma conhecimento do problema durante sua viagem, considerado fator importante para o autor.

3 SISTEMA AUTOMOBILÍSTICO COM CONTROLE POR VOZ

Neste seção, apresenta-se a descrição do sistema automobilístico e as estratégias adotadas para integrá-lo ao conector OBD2. Também, discute-se o mecanismo para o diagnóstico de erros e síntese por voz.

No sistema proposto, a leitura dos dados dos sensores via o conector OBD2 é realizada através do escâner automotivo OBD2 ELM 327 [11]. Neste escâner os dados são recebidos da ECU e transmitidos para um dispositivo conectado a ele [3]. Os códigos OBD2 são conhecidos como *Parameter Indetification* (PID) e são usados para a requisição dos dados adquiridos pelo escâner.

Os códigos PID são padronizados e é através deles que o escâner OBD2 responde às requisições. Existem dois códigos de requisição muito usados, o PID 01 que faz a requisição dos dados atuais dos sensores e o PID 03 que solicita os dados de erros detectados, chamados normalmente de *Diagnostic Trouble Codes* (DTC) ou código de avaria [3]. Os DTCs são códigos de sistemas eletrônicos que obedecem o padrão OBD2 registrados por uma ECU acerca de alguma falha que foi detectada ou que está ocorrendo no sistema.

No sistema, uma requisição usando o PID 01 deve conter um segundo par de números responsáveis por fazer a denominação do dado que se deseja receber. Na Tabela 1 são listados os principais pares PID e suas requisições específicas.

Table 1: Pares PID.

PID (hexadecimal)	Descrição
0100	Lista os PID suportados pelo veículo entre 01 e 20
0103	Status do sistema de combustível
0104	Potência do motor
0105	Temperatura do líquido refrigerante (água)
010C	Rotação do motor
010F	Temperatura do ar de admissão no motor
010D	Velocidade do veículo
0107	Correção de combustível de longo prazo
0114	Sensor de oxigênio da entrada de combustível
0111	Abertura da injeção (posição do acelerador)
...	...

Para saber quais sensores estão presentes no veículo, o sistema envia o código 0100 para o escâner automotivo através do aplicativo para celular desenvolvido (ver Figura 4). Os dados são transmitidos via Bluetooth. Após o envio do código 0100 é retornado para o sistema as informações dos sensores presentes no automóvel. A lista de códigos pode ser obtida em [12].

Supondo que uma requisição usando o PID 03 seja enviada pelo sistema, uma mensagem de resposta contendo o valor 43 seguido por 6 pares, por exemplo: "43 01 33 00 00 00 00" é recebida. Para determinar a origem do DTC, o sistema unifica cada par "0133 0000 0000" e lê o valor do primeiro dígito, que neste exemplo é 0. Neste caso, isto significa que o problema está relacionado com o sistema *Powertrain* (motor e transmissão), ou seja, P0133 é o código que deve ser pesquisado na lista de erros DTC do fabricante [3]. A Figura 2 mostra o primeiro dígito e seu correspondente significado.

If the first hex digit received is this, Replace it with these two characters

0	P0	Powertrain Codes - SAE defined
1	P1	" " - manufacturer defined
2	P2	" " - SAE defined
3	P3	" " - jointly defined
4	C0	Chassis Codes - SAE defined
5	C1	" " - manufacturer defined
6	C2	" " - manufacturer defined
7	C3	" " - reserved for future
8	B0	Body Codes - SAE defined
9	B1	" " - manufacturer defined
A	B2	" " - manufacturer defined
B	B3	" " - reserved for future
C	U0	Network Codes - SAE defined
D	U1	" " - manufacturer defined
E	U2	" " - manufacturer defined
F	U3	" " - reserved for future

Figure 2: Dígito verificador e sua origem DTC [3].

Para o envio e recebimento da requisição pelo escâner OBD2 é necessário limpar os valores previamente armazenados no escâner. O acionamento da limpeza dos dados ocorre automaticamente através do aplicativo de celular.

Conforme descrito na Figura 3, a conexão entre o celular e o escâner OBD2 é realizada via uma rede sem fio (Bluetooth) com um telefone celular. A escolha pelo Bluetooth, deve-se ao fato do escâner disponibilizar esta tecnologia de comunicação. Os dados do celular são enviados de forma online para um servidor Mosquito (MQTT) [10] através de uma conexão também sem fio *Global System for Mobile Communication* (GSM).

O sistema possibilita ao usuário obter os dados dos sensores instalados no veículo via um escâner automotivo OBD2 ELM 327 [11].

Os dados podem ser visualizados em tempo real diretamente na tela do celular do motorista através de gráficos ou remotamente na tela de um computador através de uma interface gráfica que acessa o servidor MQTT.

A Figura 4 é uma montagem feita com um print dos gráficos retirados da tela do celular durante a execução do aplicativo e a imagem do celular ao redor é apenas ilustrativa e idêntica ao modelo utilizado no experimento.

Para a leitura remota dos dados no servidor Mosquito e apresentação na tela do computador foram utilizadas as linguagens HTML5, CSS e Javascript em conjunto com a API Google Charts (gratuita) para o desenvolvimento dos gráficos.

O código abaixo é um trecho da programação para desktop feito em Javascript para obter os tópicos publicados em tempo real pelo aplicativo do celular.

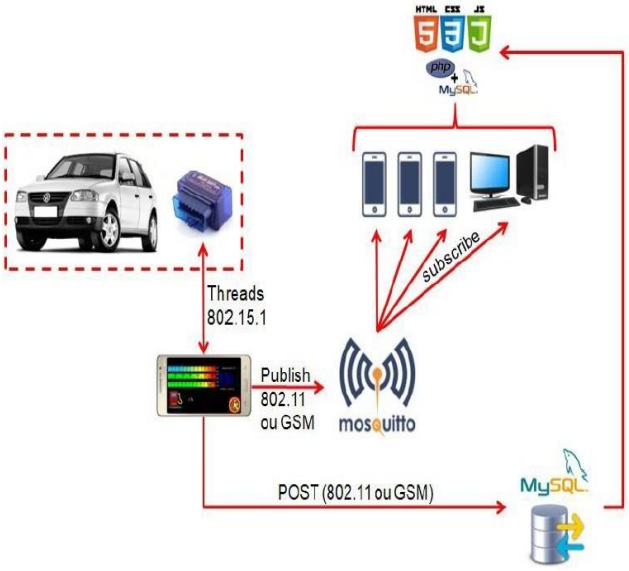

Figure 3: Esquema de Funcionamento do Sistema.

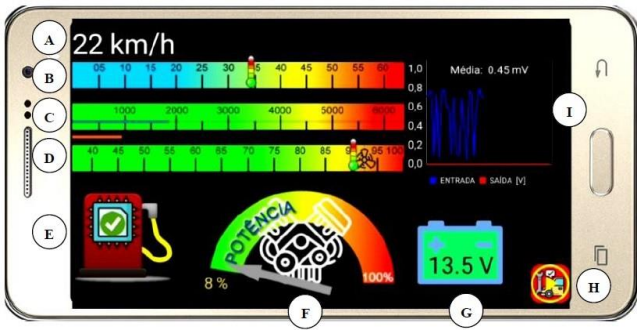

Figure 4: Interface do Aplicativo para Celular durante sua execução (print da tela).

```
function onConnect() {
        // Conexao ao Broker
        mqtt.subscribe("golmaicol/temperatura_ar");
        mqtt.subscribe("golmaicol/temperatura_agua");
        mqtt.subscribe("golmaicol/rpm");
        mqtt.subscribe("golmaicol/potencia");
        mqtt.subscribe("golmaicol/acelerador");
        mqtt.subscribe("golmaicol/velocidade");
        mqtt.subscribe("golmaicol/estado");
        mqtt.subscribe("golmaicol/lambda1");
        mqtt.subscribe("golmaicol/bateria");
    }
```

Um trecho do código para desktop também é mostrado abaixo para criar os gráficos usando o Goolge Charts.

```
function drawPotencia() {
    dataPotencia = google.visualization.
```

```
    arrayToDataTable([
        ['Label', 'Value'],
        ['Pot. [%]', 0]
    ]);
    optionsPotencia = {
        min: 0, max: 100,
        majorTicks: ["0", "20", "40", "60",
        "80", "100"],
        minorTicks: 2,
        greenFrom: 0, greenTo: 50,
        greenColor: "#00c0ff",
        yellowFrom: 50, yellowTo: 80,
        redFrom: 80, redTo: 100
     };
     chartPotencia = new google.visualization.
     Gauge(document.getElementById('potencia'));
     chartPotencia.draw(dataPotencia,
     optionsPotencia);
}
```

O aplicativo faz as requisições usando o PID 01 do serviço de diagnóstico. O modo PID 01 apresenta diversas opções conforme exemplificado na Tabela 2.

Verificou-se que o módulo OBD2 leva alguns milissegundos para responder a uma requisição. As informações sobre esta limitação estão disponíveis no sítio do fabricante [3]. Para solucionar este problema, usou-se no sistema threads para a leitura em intervalos de tempo diferentes.

Assim que o carro é ligado e a conexão com o módulo OBD2 ELM 327 é estabelecida com o celular, o sistema é acionado. Dessa forma o módulo é reiniciado e o envio de dados pode ser realizado. O código desejado é enviado pela *Integrated Development Environmen* (IDE) do aplicativo de celular.

Após a conexão estabelecida e o módulo configurado, enviam-se os códigos para a leitura dos sensores. Os dados dos sensores são enviados em intervalos definidos na aplicação, por exemplo, dados do sensor de temperatura da água, são enviados a cada 2157 milissegundos porque a temperatura não sofre muita variação. Já para o sensor de oxigênio no combustível os dados são enviados a cada 179 milissegundos.

O aplicativo desenvolvido no sistema operacional Android envia a requisição dos dados desejados (rotação do motor, temperatura do ar, sonda lambda, entre outros) e recebe do conector OBD2 o valor correspondente. O código abaixo é um trecho da requisição para o conector ELM 327 retornar a rotação do motor (PID = 010C).

```
Timer temporizador1 = new Timer();
temporizador1.scheduleAtFixedRate(new
TimerTask() {
    public void run() {
        runOnUi Thread(new Runnable() {
            @Override
            public void run() {
                if(conectado==1) {
                    if(semaforo==0) {
                        semaforo=1;
                        MyConexionBT.write
                        ("010C\r");
                    }
                }
            }
        });
    }
}, 317, 317);
```

O código abaixo é usado para receber a informação do ELM 327 através de uma expressão regular que verifica se os dados estão completos. A função "Publicar" é chamada para enviar o tópico e a informação da rotação para o servidor MQTT.

```
...
else if(PID.trim().matches(
  "010C\\s41\\s0C\\s[0-9A-F]{2}\\s[0-9A-F]{2}"
  )){
    RPM = trataPID.Rotacao(PID.trim());
    Publicar("golmaicol/rpm",String.
    valueOf(RPM));
    if(RPM==0){
        Publicar("golmaicol/estado","0");
    }else{
        Publicar("golmaicol/estado","1");
    }
...
```

Para enviar a informação ao servidor MQTT desenvolveu-se o código abaixo na plataforma Android Studio.

```
public void Publicar(String topico, String
payload){
    if(!client.isConnected()){
        ConectarMQTT();
    }else{
        byte[] encodedPayload = new byte[0];
    try {
        encodedPayload = payload.getBytes
        ("UTF-8");
        MqttMessage message = new
                    MqttMessage(encodedPayload);
        message.setRetained(true);
        client.publish(topico, message);
        } catch (UnsupportedEncodingException |
        MqttException e){
            e.printStackTrace();
    }}}
```

Para a sintetização da voz durante o recebimento dos códigos de erro DTC foi utilizado a biblioteca *textToSpeech.speak* do JAVA, nativa do Android Studio. A frase a ser sintetizada é armazenada e atribuída através dos erros DTC fornecidos pelo fabricante do automóvel (são mais de 1000 códigos DTC). Cada modelo de automóvel possui um número de erros DTC variável.

Table 2: Serviços para a Leitura dos Sensores: Automóvel Gol G4.

Código enviado	Sensor lido	Exemplo de resposta
0105	Temperatura da água	0105 4105 2F
010C	Rotação do motor	010C 410C 1A 3D
010F	Temperatura do ar de admissão	010F 410F 6B
0111	Posição do acelerador / Borboleta	0111 4111 9E
0104	Potência do motor	0104 41 3C
0103	Controle de dosagem de combustível	0103 4103 1F 5B
ATRV	Tensão da bateria	ATRV 12.4V
03	Danos ocorridos	03 43 04 41 00 00 00 00

O código abaixo é um trecho do método usado para enviar a mensagem de erro ao aplicativo.

```
public String VerificarErro(String PID){
    A = PID.substring(6,8);
    B = PID.substring(9,11);
    textofalar = A + B;
    switch (textofalar){
        case"0000":
        textofalar="Nenhuma falha detectada.";
        break;
        case "0001":
        textofalar = "Detectado circuito aberto
        no controle do regulador de volume de
        combustivel. Verifique os cabos e o
        solenoide de controle do regulador.";
        break;
        case "0002":
        textofalar = "Detectado erro no controle
        do regulador de volume de combustivel.
        Verifique os cabos e o solenoide de
        controle do regulador.";
        break;
    }
    return textofalar;
...
```

4 RESULTADOS

A conexão OBD2 com o celular foi realizada através de um módulo Bluetooth. O conector OBD2 tem um valor médio de R$ 70,00 (setenta reais).

Para o desenvolvimento do aplicativo, utilizou-se o ambiente de implementação Android Studio em um celular Samsung modelo Gran Prime Duos com sistema operacional versão Lollipop. O automóvel utilizado para o experimento é um Gol G4 ano 2006.

Os dados do celular foram enviados para uma interface WEB através de uma conexão GSM, usando-se o protocolo *Message Queuing Telemetry Transport* (MQTT) [10] com Qos de nível 1 (a mensagem é sempre entregue pelo menos uma vez). A interface WEB desenvolvida é apresentada na Figura 5.

O aplicativo Android sintetiza o texto em voz para alertar o motorista caso algum problema ocorra. Para a geração de gráficos de visualização no celular, utilizou-se a biblioteca *MPAndroidChart* para Android Studio [8]. Para gerar os gráficos no navegador foi utilizado a API Google Charts [2].

Para a leitura dos códigos DTC, criou-se uma thread que requisita o PID em intervalos de milissegundos. Além disso, criou-se uma variável global que serviu como um semáforo de acesso ao escâner ELM 327 pelas threads.

A Figura 4 mostra a tela do aplicativo em execução contendo os gráficos de velocidade (A), temperatura do ar de admissão (B), rotação do motor e posição do acelerador (C), temperatura da água (D), potência do motor (F), bateria (G) e o gráfico do sensor de oxigênio (H).

O estado do controle da dosagem de combustível (E) e o alerta de mensagem de voz para código de erro DTC (H) são mostrados como imagens. Sempre que o usuário desejar ouvir sobre algum problema no veículo, basta tocar na imagem. A imagem de erro DTC (H) aparece quando algum problema for detectado pelo módulo ELM 327 através do envio do código PID “03”.

Nos testes com o veículo foi obtido o código "03 43 04 41 00 00 00 00". No manual do fabricante do automóvel este erro, refere-se ao filtro de combustível chamado cânister. O carro foi posteriormente analisado por um mecânico, que verificou que este problema realmente existia. Ressalta-se, que não há luz de aviso no painel do motorista para este problema e o sistema desenvolvido possibilitou que o motorista tomasse conhecimento prévio e realizasse a manutenção.

O módulo OBD2 ELM 327 desenvolvido para o sistema está apto para detectar os mais de 1000 códigos de erros possíveis do manual do fabricante, sendo aplicado em qualquer automóvel utilizado.

Todos os gráficos apresentados na Figura 5 foram customizados através do Google Charts. O gráfico de velocidade (J) não foi incluído no aplicativo para celular porque o motorista já possui seu próprio velocímetro, por isso este gráfico só aparece no navegador. A imagem do controle de combustível (E) e da mensagem de erro (H) não são apresentadas na interface WEB, porém podem ser customizados caso o usuário deseje.

No momento do teste o automóvel já estava ligado por algum tempo e, como nota-se, a temperatura da água e do ar de admissão estavam altas. Na Figura 5, o gráfico com o título "Injeção", refere-se a posição do acelerador (ou a quantidade de combustível injetada).

Nos testes, observou-se um atraso na entrega da resposta do PID pelo escâner devido ao mesmo estar ocupado por outra requisição. O manual do ELM 327 contém informações sobre a temporização das respostas às requisições, porém durante os testes verificou-se

que cada PID possui um tempo de resposta diferente, variando entre 250 ms até aproximadamente 2000 ms.

Existem escâneres do mesmo modelo que podem requisitar mais de um PID ao mesmo tempo, mas há a necessidade de fazer um estudo para verificar se existe vantagem em aguardar a resposta destas requisições e quanto tempo leva para o módulo responder em relação ao custo-benefício.

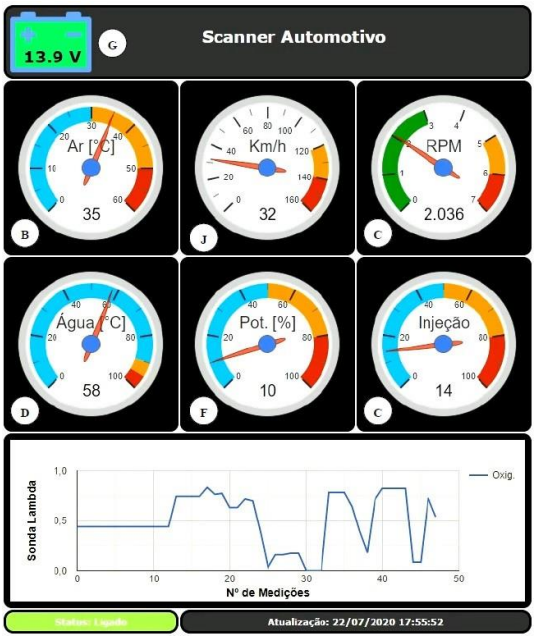

Figure 5: Leitura e apresentação gráfica dos dados no navegador.

5 CONCLUSÃO

A padronização de conectores através de entradas OBD2 em automóveis facilita o uso de escâneres para a requisição de dados de sensores. Neste trabalho foi apresentado um sistema que captura e disponibiliza automaticamente os dados dos sensores de veículos para os usuários. Um módulo OBD2 foi desenvolvido para fazer a interface com o escâner e um celular, e enviar os dados para a Internet.

Os dados podem ser visualizados através de um aplicativo e uma interface Web. Nos testes foi possível obter via aplicativo informações sobre um problema no sistema do cânister. O aviso foi feito através de voz sintetizada, o que torna o sistema utilizável enquanto o motorista dirige seu carro.

O envio e recebimento em tempo real dos dados da situação do veículo enquanto o motorista dirige pode evitar acidentes e facilitar a manutenção preditiva. O sistema tem baixo custo e os usuários devem apenas adquirir um escâner que possibilite a leitura dos sensores do carro.

REFERENCES

[1] C. R. Cruz. 2015. *Desempenho de Sondas Lambda no Monitoramento de Motores do Ciclo Otto Alimentados por Etanol e GNV*. Ph.D. Dissertation. Faculdade de Engenharia de Bauru - SP.

[2] Google Developers. [n.d.]. *Using Google Charts.* Retrieved Julho, 2019 from https://developers.google.com/chart/interactive/docs

[3] Elm Eletronics. 2017. *ELM327 OBD to RS232 Interpreter.* Retrieved Julho, 2020 from http://elmelectronics.com/DSheets/ELM327DS.pdf

[4] Juan Paúl Ortiz GONZÁLEZ, Alejandro Israel Gálvez RODRÍGUEZ, MORALES, and Daniel Antonio León. 2017. Sistema en tiempo real para el monitoreo y geoposicionamiento de un vehículo eléctrico usando un módulo obd2 y un sistema de medida inercial. *Infociencia* 11 (04 2017), 42–47.

[5] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2012. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *CoRR* abs/1207.0203 (2012). arXiv:1207.0203 http://arxiv.org/abs/1207.0203

[6] K. M. Guimarães. [n.d.]. *Dispositivo para Visualização de Informações, Detecção e Prevenção de Falhas em Veículos com Conector ODB II.* Retrieved Julho, 2017 from http://ilhadigital.florianopolis.ifsc.edu.br/

[7] Emir Husni, Galuh Hertantyo, Daniel Wicaksono, Faisal Hasibuan, Andri Rahayu, and Muhamad Triawan. 2016. Applied Internet of Things (IoT): Car monitoring system using IBM BlueMix. 417–422. https://doi.org/10.1109/ISITIA.2016.7828696

[8] Philipp Jahoda. 2020. *MPAndroidChart.* Retrieved Julho, 2020 from https://github.com/PhilJay/MPAndroidChart

[9] Janine Kniess and Vinicius de Figueiredo Marques. 2020. MARPL: A crosslayer approach for Internet of things based on neighbor variability for mobility support in RPL. *Transactions on Emerging Telecommunications Technologies* n/a, n/a (2020), e3931. https://doi.org/10.1002/ett.3931 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3931

[10] MQTT.org. 2020. *MQTT: The Standard for IoT Messaging.* Retrieved Novembro, 2020 from https://mqtt.org/

[11] ELM327 OBD. 2020. *OBD2-ELM327 Software e Hardware.* Retrieved Julho, 2020 from https://obd2-elm327.com/elm327-devices-information

[12] OnBoard Diagnostics (OBD). 2020. *OBD PID Calculator.* Retrieved Novembro, 2020 from https://www.go-ev.com/PID_Calculator.html

[13] M. Oss. 2018. Leitura OBD2 Através de Smartphone. Monografia de especialização, p79. Universidade Tecnológica Federal do Paraná, Curitiba - PR.

[14] Armando Rodríguez, José Álvarez, and Ricardo Rodriguez. 2018. Implementation of an OBD-II diagnostics tool over CAN-BUS with Arduino. *Sistemas y Telemática* 16 (04 2018). https://doi.org/10.18046/syt.v16i45.2747

[15] Uferah Shafi, Asad Safi, Ahmad Shahid, Sheikh Ziauddin, and M.Q. Saleem. 2018. Vehicle Remote Health Monitoring and Prognostic Maintenance System. *Journal of Advanced Transportation* 2018 (01 2018), 1–10. https://doi.org/10.1155/2018/8061514

[16] R. A. Staroski. 2019. *OBD-JRP: Monitoramento Veicular com JAVA e Raspberry PI*. Trabalho de Conclusão de Curso. Universidade Regional de Blumenau, Centro de Ciências Exatas e Naturais.