PROTÓTIPO DE CENTRAL DE MONITORAMENTO E SEGURANÇA AUTOMOTIVA

Ellian C. O. Costa[1], Gabriely M. T. Oliveira[1], Guilherme V. Silva[1], Huyla A. P. Lima[1], Igor C. J. Silva[1],Leandro C. Brasão[2], Maria L. S. Teixeira[1], Raquel D. M. dos Santos[1].

[1]Instituto Federal do Rio de Janeiro campus Volta Redonda

[2]Professor do Instituto Federal do Rio de Janeiro campus Volta Redonda

IFRJ campus Volta Redonda

Rua Antônio Barreiros, 212, Nossa Senhora das Graças, Volta Redonda

27.213-350 Rio de Janeiro – RJ

Resumo — O protótipo aqui apresentado consiste na criação de uma central para monitoramento e controle de dados para supervisionar, dentre outros, o tempo que o motorista passa com as mãos fora do volante. Por meio de uma fita condutiva acoplada ao volante, é possível detectar a presença das mãos, por meio do envio de dados para tratamento em uma central no veículo e na nuvem de dados. A programação em Arduino da central de monitoramento, tem o objetivo de averiguar a permanência das duas mãos do usuário no volante e, dessa maneira, garantir que o motorista mantenha atenção total ao dirigir, disparando um alarme quando o tempo limite para ausência das mãos é ultrapassado. Foi possível configurar a central para realizar chamadas e enviar SMS para um número configurado, em caso de disparo do *airbag* detectado pela rede CAN do veículo ou através de um botão de emergência desenvolvido para situações de assalto ou acidente.

Palavras chave — central de monitoramento, GSM, Rede CAN, segurança automotiva, sensor de mãos no volante

Abstract — *The project presented consists of the development of a central unit that monitors and controls data to supervise, among others, the time a driver stays with his hands out of the steering wheel. Using a conductive tape that can be added to the structure of the steering wheel it is possible to detect the presence of the hands on it, sending these data to be treated by the central unit in the cloud. The central monitoring unit programming in Arduino, has the goal to reassure that the driver stays with both hands on the steering wheel and by that maintaining total focus while driving, firing an alarm when the limit time of absent hands is exceeded. It was also possible to configure the central unit to make calls and send SMS messages to a configured number in a situation when the airbag is set off or when the emergency button developed for cases of robbery or accident is pressed.*

Keywords — *automotive safety, CAN Network, cloud, GSM, hands on the steering wheel sensor, monitoring central unit*

I. INTRODUÇÃO

É notável o aumento do número de veículos no Brasil, e, consequentemente, do número de acidentes e infrações que ocorrem diariamente nas rodovias e centros urbanos. Um dos fatores que contribui para os frequentes casos de acidente é o uso das redes sociais que mantém os usuários conectados pelo maior tempo possível, o que inclui o período em que estão dirigindo. De acordo com pesquisas realizadas pela Associação Brasileira de Medicina do Tráfego (Abramet), o uso de celular enquanto se dirige é a terceira maior causa de acidentes no trânsito brasileiro, levando à morte de cerca de 150 pessoas por dia [1].

Com base nos alarmantes dados apresentados, surgiu a ideia do desenvolvimento de uma central de segurança e controle de dados automotivos, com o objetivo de verificar se o usuário permanece com as mãos no volante, evitando, ao máximo, a interação com aparelhos celulares durante a condução.

Tecnologias para segurança automotiva vêm sendo desenvolvidas por diversas montadoras, sobretudo em modelos alta gama, no entanto, demoram para chegar ao mercado brasileiro. Por esse motivo, o presente trabalho tem o intuito de propor uma nova tecnologia de segurança em veículos que não possuem esse recurso de fábrica.

No desenvolvimento deste protótipo, foram utilizadas tecnologias como o microcontrolador Arduíno, dois módulos acoplados para realizar a comunicação com um banco de dados na Nuvem e a Rede CAN do carro, um sensor capacitivo no volante e um aplicativo para receber e indicar os dados captados por esse sensor.

II. OBJETIVOS

A. Objetivos Gerais

O protótipo, como dito anteriormente, consiste em uma central que permitirá o acoplamento de inicialmente dois acessórios desenvolvidos para assegurar a segurança de condutores. Eles terão como função principal monitorar e, se necessário, alertar o motorista

caso estejam em situação insegura. Estas funções visam aumentar o nível de concentração no trânsito e também fornecer medidas protetivas de ajuda rápida quando necessário, buscando amenizar e diminuir o número de acidentes ocorridos no trânsito.

B. Objetivos Específicos

A principal finalidade do projeto é o desenvolvimento de um sensor capacitivo de toque, instalado no volante do carro, com a finalidade de detectar se as mãos do motorista estão ou não posicionadas no volante.

O segundo objetivo é o desenvolvimento de um botão de emergência que terá como função acionar um número de celular pré-determinado pelo condutor quando pressionado, como forma de ajudar os motoristas em uma situação de assalto, defeito no veículo ou em um acidente, de forma automática.

O terceiro objetivo do projeto é a criação de um aplicativo que será responsável por reunir e indicar os dados coletados, por meio de valores numéricos e alertas, que ficarão salvos em um banco de dados para consulta posterior do usuário ou por uma gestão de frota.

III. METODOLOGIA

A. Pesquisa

O protótipo foi elaborado considerando os dados do trânsito brasileiro que cotidianamente é ameaçado por imprudências e distrações de seus condutores. Segundo levantamento do Observatório Nacional de Segurança Viária [2], 400 mil pessoas por ano são afetadas por acidentes de trânsito no Brasil. A pesquisa, realizada em parceria com o Instituto Datafolha, fez uma avaliação da percepção da população sobre o comportamento do brasileiro no trânsito, baseando-se no depoimento de 2.606 homens e mulheres. Constatou-se que o principal motivo dos acidentes está no uso do celular ao dirigir.

Este trabalho foi motivado pelos índices de acidentes e infrações que resultam em maiores taxas de acidentes. O projeto tem ênfase na segurança de motoristas e passageiros no trânsito, levando em conta a importância e a utilidade do sistema como contribuição para a sociedade, a fim de reduzir o número de acidentes ao colaborar para a segurança de passageiros e pedestres.

Outro ponto levado em consideração para a aplicabilidade do projeto no país foi o envelhecimento dos veículos brasileiros. De acordo com pesquisa realizada pelo Sindipeças [3], 52% da frota tem entre 6 e 15 anos de uso. Isso indica que, no Brasil, grande parte não possui tecnologias para segurança automotiva, desenvolvidas por montadoras e entregue em seus modelos mais recentes.

Um projeto semelhante, intitulado "*A Wireless Steering Wheel Gripping Sensor For Hands On/Off Detection*" foi desenvolvido em 2016[4] com o objetivo de fundamentar estudos psicológicos sobre o estado dos motoristas. Seu funcionamento ocorre a partir de um sensor capacitivo, um *hands on/off* (detector de presença de mãos no volante) sem fio e um detector para volantes, podendo ser integrado dentro do mesmo ou adaptado de forma que seu uso seja possível em carros.

Os conhecimentos prévios necessários para o planejamento do projeto serão esclarecidos e explicados a seguir, bem como os conceitos bases para sua elaboração.

1. ARDUINO MEGA 2560

O microcontrolador utilizado no projeto, devido à sua praticidade e funcionalidade, foi o Arduino Mega 2560 [5]. Trata-se de uma placa de circuitos mais elaborada e com maior quantidade de memória que pode ser conectada a um computador através de um cabo USB e programada, utilizando uma linguagem baseada em C/C++, via IDE (*Integrated Development Environment*) [6].

2. MÓDULO GSM SIM900 GPRS

Um dos componentes utilizados na elaboração do projeto foi um módulo GSM *Shield* [7] acoplável ao Arduino UNO e ao Arduino MEGA para conectá-lo à internet através de uma rede GPRS (uma tecnologia de comutação de pacotes que pode fornecer taxas de dados entre 56-114 kbit/s). O *Shield* age como um modem GSM, assemelhando-se a um celular. O objetivo deste módulo é realizar trocas de SMS e chamadas de voz com o uso de um cartão SIM de operadora móvel.

3. PROTOCOLO E REDE CAN

No protocolo de comunicação síncrono CAN [8] todos os módulos da rede podem trabalhar como mestre e escravo de acordo com a necessidade, seguindo o conceito de multi-mestre. Também pode ser classificado como comunicação *multicast* já que todos os módulos da rede recebem todas as mensagens enviadas.

Usualmente, a rede CAN conta com dois fios trançados, sendo um o CAN *HIGH* e o outro o CAN *LOW* que enviam informação de forma redundante, o que aumenta a confiabilidade do sistema. Os dados são interpretados por meio da medição da diferença de tensão entre esses dois fios (gráfico da Figura 1). As informações neste protocolo são enviadas em endereços fixos, *bit* a *bit* onde alguns *bits* são dominantes, tendo prioridade na rede, e outros são recessivos. Antes de enviar cada *bi*t de seu endereço, o módulo verifica se há um *bit* dominante tentando comunicar e aguarda até que este termine sua mensagem, evitando a colisão de informações por meio do CSMA/CD *with* NDA (*Carrier Sense Multiple Access/Collision Detection with NonDestructive Arbitration*).

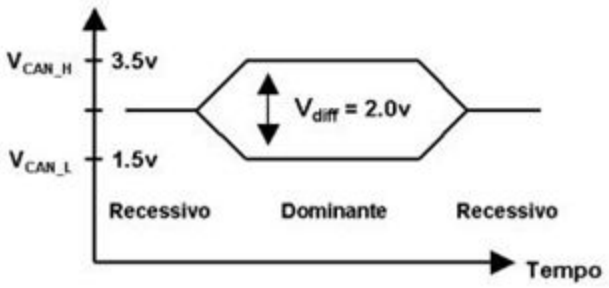

Figura 1: Gráfico da variação de tensão em uma Rede CAN.

Como apresentado na Figura 2, a velocidade de transmissão da Rede CAN é inversamente proporcional ao tamanho do chicote do carro, ou seja, quanto menor o tamanho do barramento, mais rápido os dados serão transmitidos através da rede.

Atualmente, o protocolo CAN é utilizado pelos grandes construtores automotivos devido à sua alta confiabilidade e acessibilidade. Diversas redes CAN podem coexistir em um mesmo veículo, realizando a comunicação motor, habitáculo e conforto.

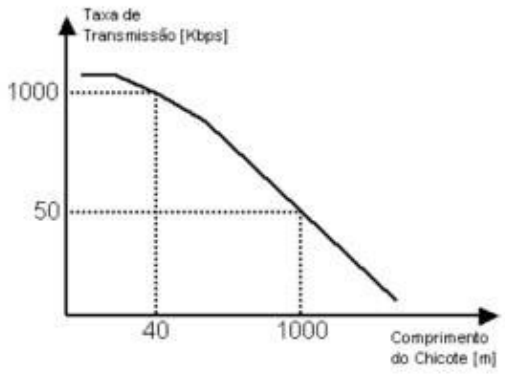

Figura 2: Gráfico Velocidade de Transmissão em função do Comprimento do Chicote

Na figura 3, é possível observar algumas das diversas centrais que existem em um carro e como elas se comunicam via Rede CAN.

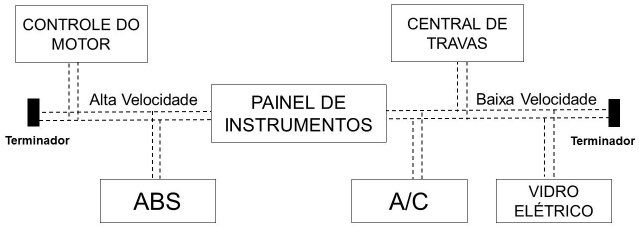

Figura 3: Diagrama das centrais de um carro

4. CAN BUS SHIELD V2

O CAN-BUS *Shield* é um módulo para Arduino cuja função é conectar microcontroladores com a Rede CAN que existe no carro. O módulo transforma a alta taxa de transferência da Rede CAN em uma baixa taxa de dados além de transformar o protocolo CAN para um protocolo *Serial* de forma que qualquer microcontrolador entenda.

B. Projeto

As atividades do projeto foram divididas entre cinco etapas: a elaboração e definição de requisitos; a implementação em cada módulo; integração dos módulos, testes, correções e a finalização. O grupo optou por dividi-lo em microprojetos e subtarefas para um melhor desenvolvimento, a partir da divisão de responsabilidades e deveres embasados nas habilidades de cada integrante, levando em consideração a visão do grupo para gerenciar os recursos humanos.

Sendo assim, as tarefas foram distribuídas da seguinte forma:

1. Criação da central a partir do Arduino MEGA;
2. Programação do servidor com lógica Python e Arduino;
3. Programação módulo GSM em Arduino;
4. Interface e desenvolvimento do aplicativo de monitoramento do projeto, com *Web Design* e *Development*, comunicação em redes e bancos de dados;
5. Programação do módulo para envio e recebimento de dados em rede CAN;
6. Construção e programação do sensor de toque.

O diagrama de blocos na Figura 4 tem como objetivo representar o fluxo de informações entre cada bloco e sua interação.

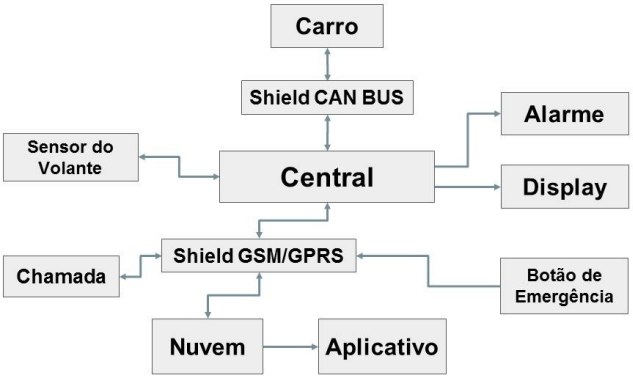

Figura 4: Diagrama de Blocos

Um fluxograma é um diagrama que, por meio de representação gráfica, ilustra um processo de modo a auxiliar o entendimento do funcionamento do mesmo. O fluxograma do projeto descrito está representado na Figura 5. Este permite identificar as principais decisões e ações realizadas pelo conjunto.

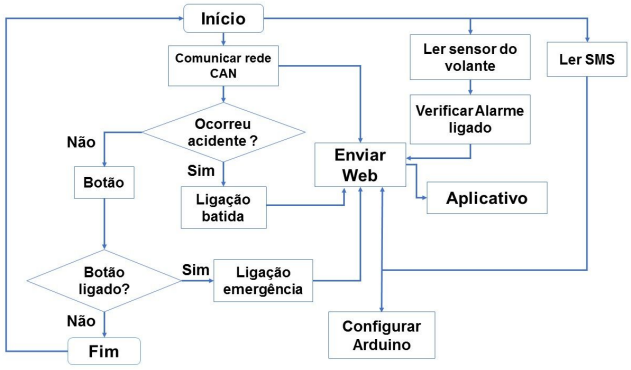

Figura 5: Fluxograma do Processo

C. Materiais

No desenvolvimento do projeto foram utilizados os materiais descritos na Tabela 1.

Tabela 1: Tabela de Materiais e Custos

Material	Quantidade	Preço
Arduino MEGA 2560	1	R$50.00
Shield GSM Sim900 GPRS	1	R$115.00
CAN BUS Shield V2	1	R$28.83
Capa de Volante	1	R$15.00

Fita Condutiva	1	R$29.00
Tomada BD2	2	R$2.40
Buzzer	1	R$2,00
Módulo IC2 para LCD	1	R$12,00
Display LCD	1	R$16,90
Microfone	1	R$10,90
Resistores	6	R$0.60
Placa de Fenolite	1	R$9,00
Soquete	8	R$20,00
ESP32	1	R$60,00
Caixa de MDF	1	R$8,00
Botão	1	R$2,50
Total		R$383,13

IV. DESENVOLVIMENTO

A. Construção do Sensor Capacitivo

O principal acessório desenvolvido é o sensor capacitivo, acoplado ao volante por meio de uma capa onde este foi introduzido. Como mencionado anteriormente, a função do sensor é detectar a presença de cada uma das mãos do condutor no volante e quanto tempo cada mão permanece nele.

A fita condutiva foi dividida em duas metades, uma do lado direito e outra do lado esquerdo da capa do volante, para que fosse possível detectar a mudança da capacitância e, consequentemente, a presença das mãos direita e esquerda em cada um dos lados do volante.

No esquema da Figura 6, é possível observar como são feitas as conexões do sensor do volante ao Arduino. O circuito acrescentou um resistor de 10 MΩ entre o pino de envio e o pino de recepção. O pino de recepção, onde está o resistor, é o terminal do sensor, ou seja, a ponta da fita condutiva, que está ligada a um pino de entrada digital do arduino.

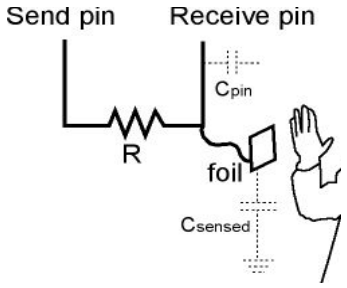

Figura 6: Circuito Esquemático do Sensor Capacitivo

A escolha do valor do resistor se deve à necessidade de uma sensibilidade específica para o sensor, já que um resistor de valor baixo resultaria em um sensor capacitivo de baixa sensibilidade, dificultando a detecção das mãos. Um resistor com um alto valor geraria uma sensibilidade excessivamente alta o que aumentaria a probabilidade de interferências no sinal lido.

Uma capa de volante foi utilizada como suporte da fita para melhor adaptação ao veículo. O protótipo do sensor está representado na figura 7.

Figura 7: Protótipo do Sensor Capacitivo Acoplado ao Volante

B. Comunicação

1. CAN

A comunicação com a Rede CAN é realizada através do *Shield* CAN BUS V2 que é capaz de ler e enviar informações para a rede de um carro, viabilizando o acesso a informações necessárias para a continuação do projeto.

Como não seria viável testar o projeto em um carro a cada alteração, utilizou-se um simulador de Rede CAN que permite ao usuário criar tramas e nomear sua função, por exemplo, a trama 0x01 contém a informação de velocidade do carro. O simulador foi programado para enviar tramas com 8 *bytes* de informações, em base hexadecimal. Esta programação deve ser adaptada a cada veículo individualmente, pois o endereço e a posição das informações não são padronizados em diferentes construtores automotivos. Foram criados endereços e informações diversos, simulando o ambiente veicular tais como: velocidade, posição da chave do veículo, disparo do airbag e tipo de choque.

Para filtrar e ler as tramas do carro foi utilizada a biblioteca MCP_CAN por ser compatível com o módulo CAN MCP2515. A função para inicializar a taxa de transmissão do CAN é definida como #define CAN_125KBPS, além de definir um *clock* de 8MHz.

É possível escolher qualquer taxa de transmissão desejada desde que todos os remetentes e receptores estejam configurados da mesma forma. Taxas mais rápidas permitem que se transfira mais dados, porém as mais lentas são mais confiáveis e permitem a transferência entre distâncias maiores.

A função readMsgBuf permite saber o tipo de ID (se é estendido ou padrão), além de retornar o *bit* de *status* (mensagem recebida). Se o ID for 0x80000000, é do tipo estendido, caso contrário, padrão. O ID for 0x40000000 significa que a mensagem é remota.

Feita a configuração das máscaras e filtros, a função é retornada de acordo com o controle destes, de modo que a mensagem não será recebida se estiver fora do controle.

Para recuperar os dados obtidos a partir do simulador foi usado o *Shield* CAN BUS que é capaz de receber essas informações e enviá-las para o Arduino MEGA. Essa conexão entre simulador, *shield* e arduino foi feita a partir de comunicação SPI (*Serial Peripheral Interface*) que consiste em um modo de comunicação *serial* síncrona capaz de atingir velocidades maiores de troca de dados em um sistema *full-duplex* com direções de transmissão pré-definidas.

2. GSM/GPRS

A comunicação entre o Arduino e o cliente foi realizada através do módulo Sim900 GSM GPRS. A comunicação entre o microcontrolador e o *shield* é feita através de um protocolo *serial* que utiliza linguagem de comando AT. Esses são comandos capazes configurar o SIM900 e definir quais serão as ações a serem tomadas. A biblioteca GPRS_SIM900-master utilizada no projeto é capaz de encapsular os AT em comandos de mais alto nível e substituir a necessidade de enviar comandos AT constantemente.

O Arduino precisa enviar diversos comandos AT seguidos para verificar a comunicação e mandar ordens para que o SIM 900 ligue e envie mensagens de texto. A biblioteca resume toda comunicação AT feita entre o Arduino e o módulo em uma única função.

C. Funções

1. Função de Monitoramento das Mãos no Volante

O sensor capacitivo tem como função detectar a presença das mãos do condutor no volante, para isso, a fita condutiva do sensor e o corpo humano agem como duas placas de um capacitor que armazena carga. A capacitância depende de quão perto a mão do condutor se encontra da placa, de modo que, ao aproximar a mão do sensor, o valor da capacitância vai aumentando até que atinja o valor máximo quando o toque é efetuado.

O arduino tem a função de medir, de forma precisa, o tempo necessário para que o capacitor carregue, estimando sua capacitância. Neste contexto, a biblioteca *capacitiveSensor* é a responsável por transformar os pinos do Arduino em um sensor capacitivo, alternar o pino de envio do microcontrolador para um novo estado e, em seguida, aguardar o pino de recebimento mudar para o mesmo estado do pino de envio. Esta mudança é cronometrada através de uma variável incrementada dentro de um *loop while*.

A programação realizada em Arduino tem como função filtrar os dados capacitivos captados pelo sensor e determinar se uma mão está no volante, se as duas estão ou se nenhuma está.

Também pode indicar, por meio da programação, o tempo de ausência das mãos do motorista no volante em relação à velocidade do veículo, acionando um alerta visual e até um sonoro, caso o período limite seja ultrapassado. A partir do momento em que for detectado que o condutor se encontra com apenas uma mão no volante, a velocidade do carro é analisada em três critérios: quando o veículo estiver sendo conduzido nestas circunstâncias por mais de 15 segundos em uma velocidade superior a 5 km/h e inferior a 25 km/h, o alarme é acionado; se a velocidade estiver entre 25 km/h e 50 km/h por mais de 10 segundos, o motorista será alertado; o motorista também é alertado quando estiver conduzindo o veículo com apenas uma mão por mais de 7 segundos a mais de 50 km/h. Nos casos em que não for detectada a presença de nenhuma mão no volante, o acionamento do alarme procederá sob os seguintes critérios: quando o presença de mãos não for detectada por mais de 5 segundos a uma velocidade mínima de 5 km/h e máxima de 25 km/h; a ausência ultrapassar 3 segundos em uma velocidade entre 25 km/h e 50 km/h; nos momentos em que, por mais de 2 segundos, o veículo estiver a mais de 50 km/h.

O fluxograma da Figura 8 demonstra o funcionamento desta função.

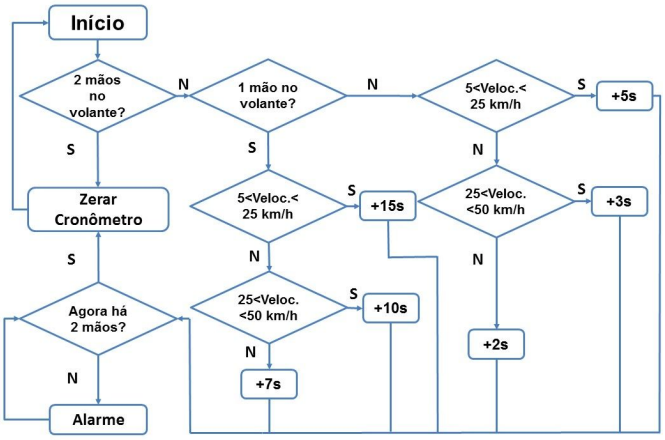

Figura 8: Fluxograma de Mãos no Volante

2. Função de Situação de Emergência

O papel dessa função é realizar uma ligação e enviar uma mensagem de texto caso o botão de emergência tenha sido acionado ou ocorrer disparo do *airbag*, a partir de informação proveniente da Rede CAN.

Primeiramente, a programação do Arduino MEGA define o tempo de comunicação *serial* entre ele e o módulo SIM900 GPRS para o valor padrão de 19200. O programa Arduino envia alguns comandos em comunicação AT para o SIM900 de forma a verificar se o módulo inicializou corretamente.

Após a inicialização ser realizada, o Arduino verifica a existência de um número de telefone gravado anteriormente em sua memória EEPROM. Caso exista, este será definido como o número que receberá ligações e mensagens de texto em caso de emergência. Quando não houver nenhum número de telefone previamente gravado ou o cliente desejar trocá-lo, este poderá ser adicionado por uma mensagem de texto enviada pelo cliente para o SIM900, assim, o novo número substituirá o anterior e se tornará o novo contato para emergências.

Para que uma mensagem de texto altere o número de emergência, o cliente deve iniciá-la com a letra "A" e escrever o novo número de telefone em seu formato internacional (+5524).

Quando o botão de emergência for acionado, uma mensagem de texto será enviada e uma ligação realizada para o número configurado pelo cliente. Utiliza-se da biblioteca GPRS_SIM900-master para a realização destes comandos.

Esses comandos também serão executados quando a Rede CAN apresentar mudança na trama correspondente ao *airbag*, indicando seu disparo, de acordo com o fluxograma da Figura 9. Essa mudança será detectada pelo *Shield* CAN BUS que enviará as informações para o Arduino sobre essa mudança de estado.

Outra ação a ser realizada através do Arduino quando a função de emergência for acionada será a abertura do microfone acoplado à central, permitindo que a pessoa contatada neste caso ouça tudo o que acontece no carro enquanto a função estiver ativada.

Devido ao fato do sinal telefônico ser insuficiente em algumas áreas, será feita uma redundância no comando para que, em casos de falha, o Arduino repita o comando até o êxito no envio do SMS e da ligação.

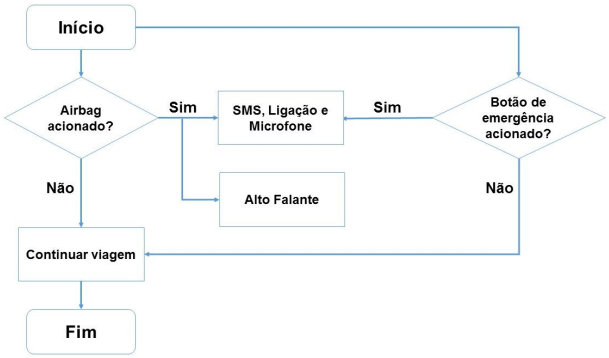

Figura 9: Fluxograma da Situação de Emergência

D. Função de Monitoramento do Cinto de Segurança

A partir das tramas enviadas pelo veículo na Rede CAN é possível avaliar a situação do cinto de segurança do motorista, ou seja, se ele está colocado ou não quando o carro está ligado. Caso o motorista esteja negligenciando o uso do cinto de segurança, uma mensagem aparecerá na tela e um alerta sonoro será ativado até que a situação esteja normalizada.

E. Integração

Considerando a totalidade do projeto descrito, torna-se necessária a criação de uma central capaz de reunir todos os dados coletados e distribuí-los para as partes que precisam ter acesso a eles. Tanto os dados coletados pelo sensor capacitivos como os provenientes da Rede CAN e do botão de emergência se concentram no Arduino MEGA.

O primeiro passo para integrar os dados foi conectar os shields CAN BUS e o SIM900 de maneira eficiente no Arduino para que as informações de ambos fossem acrescentadas às do sensor, já acoplado ao microcontrolador. Para os dados coletados da Rede CAN, o shield CAN BUS foi colocado no topo do Arduino MEGA como mostrado na Figura 10. O *shield* SIM900 usou dois pinos seriais e o sensor capacitivo dois pinos de entrada digital. Na Tabela 2, estão apresentados quais pinos foram utilizados e suas finalidades.

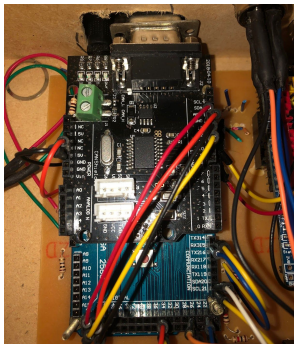

Figura 10: *Shield* CAN BUS conectado ao Arduino MEGA

Tabela 2: Pinagem do Projeto

PINAGEM TOTAL DO PROJETO

Pinos do Arduino MEGA	**Pinos Conectados ao Arduino**
VCC	SIM900(VCC), Display(VCC), ESP32(VIN)
GND	Botão de Emegência, Buzzer, SIM900(GND), Display(GND), ESP32(GND)
10	SIM900 (Pino 7)
11	SIM900 (Pino 8)
22	Sensor do Volante
23	Sensor do Volante
26	Sensor do Volante
27	Sensor do Volante
34	Botão de Emergência
35	Resistor + Buzzer
50	CAN BUS(Pino12)
51	CAN BUS(Pino11)
52	CAN BUS(Pino13)
53	CAN BUS(Pino 10)
sda	Display
scl	Display
RX3	ESP32(TX2)
TX3	Divisor de Tensão + ESP32(RX2)

No intuito de captar as informações das diversas partes do projeto e enviá-las para a nuvem um ESP32 foi acoplado ao microcontrolador. Ele serve como escravo do Arduino MEGA, ou seja, coleta os dados concentrados nele via protocolo serial e os envia via *Wi-fi* para um banco de dados na nuvem.

Visando à melhora estética do protótipo e a garantia de que possíveis danos não viessem a ocorrer, o *shield* SIM900 e o ESP32 foram soldados a uma placa com um soquete, e colocados dentro de uma caixa junto com o Arduino MEGA. É importante ressaltar que o *shield* SIM900 foi colocado de maneira que sua antena pudesse ficar posicionada para o lado externo da caixa, caso contrário, o sinal telefônico poderia sofrer interferência, como mostrado na Figura 11.

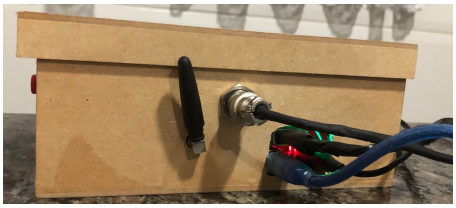

Figura 11: *Shield* SIM900 com sua antena posicionada fora da caixa

No lado externo da caixa com os componentes, foi adicionado um *display* para apresentação de dados ao motorista durante a condução, e resumo das estatísticas ao final de cada percurso. Esta tela apresenta os dados de velocidade do veículo, temperatura do

motor, situação do cinto de segurança do motorista, em virtude das informações presentes na Rede CAN, além da quantidade de mãos no volante e mensagens para situações de alerta ou emergência.

F. Aplicativo

Para armazenar os dados enviados pelo ESP32, sejam eles a respeito do sensor capacitivo, botão de emergência ou provenientes da Rede CAN, foi desenvolvido um banco de dados em um servidor na *Web*. O servidor utilizado é o *WebHost*, oferecido gratuitamente pela HostNet, porém poderia ser substituído por qualquer outro serviço de hospedagem, visto que a maioria possui as funções necessárias de PHP(*PHP: Hypertext Preprocessor*)[9] e SQL(*Structured Query Language*). O banco de dados tem como função integrar as informações das múltiplas partes do projeto para serem apresentadas ao usuário pelo aplicativo.

O banco de dados é do tipo MySQL, ou seja, tem a licença livre, que pode ser manipulado puramente por comandos SQL ou inseridos em linguagem PHP. A partir do método HTTP(*Hypertext Transfer Protocol*)/POST, é possível realizar a comunicação do *shield* com a página na *Web* que compartilha o servidor do banco de dados. A página recebe os dados enviados em HTTP/POST e os interpreta em PHP, depois essas informações são enviadas via SQL para o banco em MySQL. A relação de envio e recebimento de dados está representada pela Figura 12.

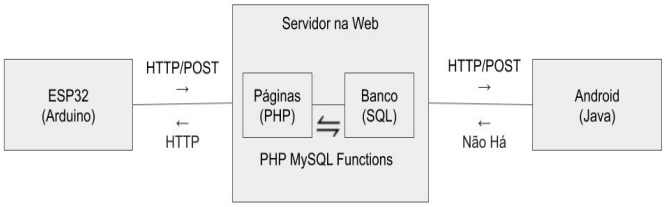

Figura 12: Esquema de Envio e Recebimento de Dados

Uma API (*Application Programming Interface*) foi desenvolvida para facilitar, de maneira segura, a aquisição de dados do banco em SQL. A API é básica pois aguarda por um código do usuário, inserido através de HTTP/POST, para retornar um arquivo em JSON(*JavaScript Object Notation*)[10]. Este armazena todos os arquivos que compõem o banco de dados.

O aplicativo foi desenvolvido em Java no Android Studio e tem como objetivo apresentar, na tela do celular do condutor ou mesmo do gestor de uma frota, os dados das viagens realizadas.

Nas informações apresentadas estão incluídos: temperatura do motor, porcentagem do tempo sem nenhuma mão no volante, porcentagem do tempo só com uma mão no volante, descartados os períodos de tolerância em função da velocidade, velocidade média da viagem e alerta de emergência. Os dados ficam dispostos na tela do aplicativo como mostrado na Figura 13.

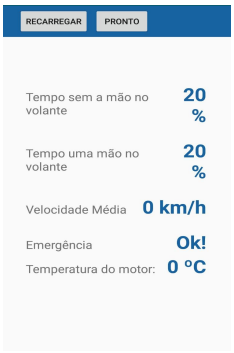

Figura 13: Tela principal do aplicativo com informações gerais do projeto.

Outras informações presentes na rede CAN do veículo podem ser facilmente apresentadas ao cliente com pequenas adaptações no aplicativo.

O aplicativo acessa os dados do banco e os apresenta ao usuário. A comunicação é baseada em um *WebService* com uma *API* escrita em PHP. Os *scripts* em PHP, quando acionados por uma chave exclusiva para cada usuário, retornam os dados necessários para o aplicativo em forma de um arquivo JSON que é facilmente compreendido pelo aplicativo.

Para situações em que não há sinal *Wifi,* ou acesso à *internet*, e não é possível enviar os dados que o ESP32 recebe para o banco de dados, um *buffer* é realizado para salvar esses arquivos não enviados, permitindo seu envio posterior, quando for possível acessar a *internet* novamente. Um *buffer* é uma parte da memória SKRAM do ESP32, ou seja, uma matriz que vai sendo ocupada a partir de cada POST enviado. Essa memória é de um tipo volátil, o que significa que com o desligamento do microcontrolador os dados que estavam salvos nela serão perdidos. Com isso é possível contemplar situações com perda de sinal de *internet,* mas não situações de perda de energia.

No que diz respeito ao acesso ao aplicativo, um sistema de senha foi desenvolvido para que somente o responsável pela gestão dos dados captados em cada viagem tenha acesso aos mesmos.

V. RESULTADOS

O sensor capacitivo de toque ao volante cumpre o objetivo principal do projeto, detectando quando o condutor solta o volante ou se encontra com apenas uma ou ambas as mãos no volante. O programa consegue ativar um alarme visual e um alarme sonoro, caso o motorista esteja há muito tempo com uma ou duas mãos fora do volante em função da velocidade do veículo informada pela rede CAN.

O objetivo secundário de projetar uma função de emergência capaz de acionar um SMS e realizar uma ligação para um número definido foi atingido também. Essa função é ativada quando o botão de emergência é pressionado pelo motorista ou quando é detectada alguma mudança na trama do *airbag*. No entanto, o alto falante não ficou inibido em caso de uma chamada de emergência.

Outro resultado alcançado foi a comunicação entre o *shield* CAN BUS e as tramas enviadas pelo simulador de Rede CAN. Por meio dessa troca de informações foi possível simular, com sucesso, eventos que seriam enviados em um veículo pela Rede CAN, como o acionamento do *airbag*. Uma adaptação deve ser realizada para aplicação específica em cada veículo conforme mencionado.

Em relação ao *display* de apresentação, este mostra uma série de informações em seu estado nominal, ou seja, sem nenhum alerta ou emergência, como visto na Figura 14. Por meio dos dados do

sensor, são apresentadas quantas mãos estão no volante e, por meio dos dados da Rede CAN, mostra-se a situação do *airbag*, do cinto de segurança do carro, a temperatura do motor e a velocidade do carro.

Figura 14: *Display* de apresentação em seu estado nominal

Em caso de alerta de mãos fora do volante, em conjunto ao acionamento de um alerta sonoro, o display muda sua mensagem para "Volante", como demonstrado na Figura 15.

Figura 15: *Display* de apresentação em caso de alerta para mãos fora do volante

Para caso de ativação da função de emergência, aparecerá a mensagem "*Airbag* Disparado Emergência" para quando o airbag for o responsável por ativar a função, como na Figura 16.

Figura 16: *Display* de apresentação em caso de disparo do *airbag*

Ainda há uma terceira mensagem é "Cinto!" que aparece quando o cinto de segurança do motorista não está colocado, como mostrado na Figura 17, além da ativação de um alerta sonoro até que o motorista coloque o cinto.

Figura 17: *Display* de apresentação em caso de cinto de segurança não colocado

Em relação ao aplicativo, foram alcançados os objetivos de comunicá-lo ao resto do projeto e trazer para o usuário a facilidade de acessar os dados resumidos de suas viagens no celular. Como mostrado na Figura 13, os dados mais importantes do projeto ficam dispostos na tela do aplicativo. Em relação à senha de acesso, o resultado foi positivo e o usuário só consegue acessar a tela principal com a senha correta. É possível ver na Figura 18 uma situação em que a senha incorreta foi inserida, validando o objetivo buscado.

Figura 18: Situação do Aplicativo quando a senha incorreta é introduzida.

Os códigos do projeto descrito, tanto do aplicativo como do sensor capacitivo e comunicação CAN e GSM, estão disponível para acesso ilimitado e *dowload* na plataforma GitHub através do *link*: https://github.com/EllianCarlos/SegurancaAutomotiva

VI. CONCLUSÕES

Os crescentes índices de acidentes de trânsito motivaram o desenvolvimento do projeto aqui descrito, o que ressalta sua relevância e aplicação na sociedade haja vista que garante a completa atenção do motorista no trajeto, assim como oferece recursos de emergência.

Em relação aos objetivos gerais e específicos, todos os resultados esperados foram atingidos com êxito, incluindo a central desenvolvida para acoplar os acessórios relacionados ao volante e ao botão emergencial e, ainda, o aplicativo para o controle e gerenciamento de dados.

A partir dos resultados obtidos pode-se verificar que a concentração do motorista passa a ser redobrada, pois o alarme sonoro, ao ser ativado pela ausência de mãos no volante, retoma a atenção do condutor para o percurso.

À respeito do botão de emergência, este presta auxìlio quase imediato em caso de acidente ou roubo do veículo permitindo seu acionamento de maneira discreta neste último caso.

O aplicativo atua como uma forma de monitoramento de dados, sejam eles para fins de controle empresarial ou de acesso do próprio usuário.

Tendo em vista as resultantes do projeto, acredita-se que, com as melhorias propostas, este contribuirá para uma sociedade mais segura e que preze, acima de tudo, pela vida.

VII. PROPOSTAS DE MELHORIAS

Por ser um protótipo, muitas melhorias precisam ser feitas para seu uso comercial e instalação em um veículo. Dentre as sugestões de melhoria e trabalhos futuros destacam-se:

- A substituição do ESP32 no papel de escravo do Arduino por um módulo *Wi-fi* próprio para o Arduino ou usar a rede GPRS do *shield* SIM900, diminuindo o custo do projeto;
- Atualmente, o número de emergência é definido através de um SMS que o usuário envia ao *shield* GSM e este envia as informações para o Arduino. A proposta seria fazer essa declaração do número de emergência via aplicativo, integrando a comunicação usuário/central.
- A comunicação com a Rede CAN de maneira bilateral, ou seja, a possibilidade de envio de informações para o carro. Atualmente, o Arduino se comunica com a Rede CAN, mas está apenas lendo as informações enviadas pelo carro.

Ao enviar dados para o carro seria possível, em parceria com uma montadora, utilizar as interfaces e alto falantes do próprio veículo.

- A evolução na complexidade do API do aplicativo para que este seja capaz de selecionar quais dados serão coletados do banco de dados para serem apresentados ao usuário.
- A construção de uma placa integrada para o acoplamento dos shields, do Arduino e do ESP32.
- A substituição dos fios que ligam a capa do volante ao microcontrolador por alguma outra maneira de comunicação que não seria danificada durante o giro do volante.
- Utilização da energia proveniente do carro para alimentar os *shields*;
- Acesso ao histórico do banco de dados, fazendo com que fiquem visíveis no aplicativo, e nenhuma informação seja perdida pelo usuário;

REFERÊNCIAS BIBLIOGRÁFICAS

[1] AMARAL, Paulo. “Uso de celular ao volante aumenta em 400% risco de acidentes”. Disponível em: <https://www.huffpostbrasil.com/entry/usar-celular-dirigir_br_5c46214ae4b0bfa693c63898>. Acesso em: 12 de abril 2019.

[2] M.S. “Brasil registra 47 mil mortes por acidentes no trânsito anualmente.” Dispon´vel em: <https://www.revistaapolice.com.br/2018/05/brasil-47-mil-mortes-acidentes-transito/>. Acesso em 15 de abril em 2019.

[3] RODRIGUES, Lino. “Frota de veículos no Brasil cresce, mas continua envelhecida”. Disponível em: <https://www.em.com.br/app/noticia/economia/2018/04/13/internas_economia,951278/frota-de-veiculos-no-brasil-cresce-mas-continua-envelhecida.shtml>. Acesso em: 09 de abril de 2019.

[4] FALLER, Lisa-Marie; HAMID, Raiyan; MÜHLBACHER-KARRER, Stephan; ZANGL, Hubert. “A wireless steering wheel gripping sensor for hands on/off detection”. Disponível em: <https://ieeexplore.ieee.org/document/7479878>. Acesso em: 17 de abril de 2019.

[5] SOUZA, Fábio. “Arduino MEGA 2560”. Disponível em:<https://www.embarcados.com.br/arduino-mega-2560/>. Acesso em: 23 de fevereiro de 19.

[6] THOMSEN, Adilson. “O que é Arduino?” Disponível em: <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em: 23 de fevereiro de 19.

[7] Arduino: “Getting Started with the Arduino GSM Shield”. Disponível em: <https://www.arduino.cc/en/Guide/ArduinoGSMShield>. Acesso em: 23 de fevereiro de 2019.

[8] GUIMARÃES, Alexandre; SARAIVA, Antônio. “O Protocolo CAN: Entendendo e Implementando uma Rede de Comunicação Serial de Dados baseada no Barramento ‘Controller Area Network’”. Disponível em: <http://www.alexag.com.br/Artigos/SAE2002.pdf>. Acesso em: 12 de abril de 2019.

[9] VINCY. “PHP RESTful Web Service API - Part 1 - Introduction With Step-by-Step Example”. Disponível em: <https://phppot.com/php/php-restful-web-service/>. Acesso em: 15 de junho de 2019.

[10] GHANEM, Hamdy. “Send and Receive JSON Between Android and PHP Web Service”. Disponível em: <https://www.codeproject.com/Articles/267023/Send-and-receive-json-between-android-and-php>. Acesso em: 15 de junho de 2019.