DESENVOLVIMENTO DE UM SISTEMA DETECTOR DE DISTRAÇÃO PARA USO EM VEÍCULOS UTILIZANDO ESP32 E VISÃO COMPUTACIONAL

Jonas Pereira Geremias[1]

Vagner Da Silva Rodrigues[2]

Resumo: Com a necessidade de minimizar os acidentes no trânsito, diversos dispositivos foram desenvolvidos para a indústria automobilística. O cinto de segurança, espelhos retrovisores, *airbags*, freios ABS, para-brisas laminados são alguns exemplos de equipamentos de segurança veicular. Porém, tais dispositivos não evitam o maior causador de acidentes: a distração. Desta forma, o presente artigo apresentou o desenvolvimento de uma solução de hardware utilizando o chip ESP32 como base e conceitos de visão computacional para aplicação em veículos com objetivo de detetar a distração do motorista. Foram apresentadas as etapas de definição do hardware, coleta e análise dos dados, criação de firmware e testes funcionais. Foi mensurado através dos testes o tempo de reposta para a aquisição de imagem de diversos tamanhos, processamento utilizando visão computacional e *deep learning*, com biblioteca disponibilizada pelo fabricante do chip, e criação da lógica da aplicação final usando os periféricos como câmera, wi-fi e cartão de memória. Os resultados mostraram que o ESP32 em sistemas mais complexos, envolvendo processamento de imagem ou *deep learning*, apresentou-se lento nessas aplicações, sendo possível ser utilizado na aplicação desenvolvida neste projeto, mas impossibilitando melhorias na solução, como a identificação de motoristas e deteção de uso de celular ou fumaça. Durante os testes de deteção de distração, foram observados os principais fatores que devem analisados nas aplicações futuras, são eles: o tamanho e a qualidade da imagem, o tempo de salvamento dos dados em memória, o tempo de deteção, a luminosidade, além das tarefas em execução paralela. Mostra-se desta forma o potencial a ser explorado nesta área de desenvolvimento tecnológico e a possibilidade de aprofundamento nos campos de integração de sistemas embarcados e visão computacional para a implementação com maior assertividade desta solução.

Palavras-chave: Distração. Sistemas embarcados. Visão Computacional. ESP32-CAM.

1 INTRODUÇÃO

Umas das mais importantes invenções já criadas pelo ser humano foi o automóvel. O primeiro automóvel com tamanho suficiente para transportar pessoas e cargas foi criado no final do século XVIII (ECKERMANN, Erik, 2001). As

[1] Graduando em Engenharia de Computação. Ano 2022-1. E-mail: jonasgeremias@hotmail.com
[2] Professor do Centro Universitário UniSATC E-mail: vagner.rodrigues@satc.edu.br

transformações ao longo dos séculos tornaram o automóvel um dos principais meios de transporte. Porém, com ele surgiram algumas consequências como os acidentes de trânsito. A chegada do telefone celular, por exemplo, colaborou para diminuir a atenção do motorista, o que potencializa as chances de acidentes por falta de atenção. Uma das métricas de informação sobre acidentes nas estradas nacionais pode ser obtida por meio do Atlas da Acidentalidade no Transporte, que foi desenvolvido pelo Programa Volvo de Segurança no Trânsito (PVST) para determinar o tamanho da acidentalidade nas rodovias federais brasileiras. Em 2020, cerca de 38% dos acidentes registrados no Brasil aconteceram por falta de atenção, sendo a principal causa dos acidentes de trânsito (Atlas da Acidentalidade no Transporte Brasileiro, 2021).

Essa realidade nacional também é relatada a nível mundial, tanto que a Organização das Nações Unidas (ONU), juntamente com a Organização Mundial da Saúde (OMS), proclamou em 2010 a Década de Ação pela Segurança no Trânsito 2011-2020 (WAISELFISZ, 2013; WORLD HEALTH ORGANIZATION, 2011). O plano buscou estabilizar a mortalidade no trânsito e reduzir as projeções de mortalidade, atuando em cinco pilares de intervenção: fortalecimento da gestão, investimento em infraestrutura, segurança veicular, comportamento e segurança dos usuários do trânsito, e atendimento pré e intra-hospitalar ao trauma. O assunto é tão importante que a ONU definiu o período de 2021 a 2030 como Segunda Década de Ação pela Segurança no Trânsito (ABRAMET, 2021).

De acordo com a ONU, “a grande maioria das mortes e ferimentos graves no trânsito são evitáveis e, apesar de algumas melhorias em muitos países incluindo países em desenvolvimento, as mortes e ferimentos permanecem um grande problema de saúde pública que tem amplas consequências sociais e econômicas”. (ABRAMET, 2021).

Segundo o Sindicato Nacional da Industria de Componentes para Veículo Automotores, a idade média dos automóveis é de aproximadamente 10 anos. A frota em circulação no Brasil era compreendia em mais de 46 milhões de veículos em 2020, dos quais, 24% apresentavam idade média de até 5 anos, o restante possui idade superior (SINDIPEÇAS, 2021). Ou seja, pelo menos 11 milhões de veículos continuarão em circulação pelos próximos 5 anos, e a grande maioria não possui nenhum dispositivo para evitar a distração do motorista.

A fim de minimizar os acidentes no trânsito, vários dispositivos foram criados e implementados nos automóveis. O cinto de segurança, espelhos retrovisores, *airbags*, freios ABS, para-brisas laminados são alguns exemplos de equipamentos de segurança veicular. Porém, estes não evitam o maior causador de acidentes: a distração.

Para diminuir a distração do motorista, as montadoras de veículos desenvolveram sensores para monitorar os padrões de comportamento no uso da direção e dos pedais do carro, emitindo alertas quando detetado o cansaço do motorista. Entretanto, estes dispositivos são tratados como opcionais em modelos de veículos de custo mais elevado. Também é possível utilizar sensores para monitorar os batimentos cardíacos e detectar sonolência do motorista interligando-os aos celulares, mas a aceitação do dispositivo pelos motoristas pode ser reduzida pelo fato de não poder utilizá-lo durante o deslocamento.

Como alternativa, a visão computacional também pode ser utilizada para solucionar o problema descrito anteriormente. Identificar padrões como a orientação da visão do motorista, se os olhos estão abertos ou fechados e até mesmo detectar o uso do celular enquanto dirige, ou ainda fumaça dentro do veículo, são possíveis através desta técnica.

De modo geral, visão computacional é responsável pela forma com que o computador consegue interpretar o ambiente que está ao seu redor, extraindo informações através de câmeras de vídeos e sensores, e assim, possibilitando que o computador reconheça e manipule os objetos que compõem as imagens capturadas (LI; SHI, 2018).

Visto que já existem equipamentos capazes de identificar e alertar sobre a distração dos motoristas, e que os preços dos sensores são relativamente altos, o estudo e desenvolvimento de um protótipo de sensor de distração usando os conceitos de visão computacional, somados ao incentivo de uso por parte do Governo, pode trazer como resultados positivos à sociedade, diminuindo a distração do motorista no trânsito e melhorar a segurança das pessoas no transporte rodoviário e urbano.

Neste contexto, o objetivo deste trabalho é desenvolver um protótipo de um sistema embarcado que utilize visão computacional para detectar a distração de motoristas. Como objetivos específicos, validar o chip ESP32 como o microcontrolador

para aplicação, definir o *hardware* embarcado: contendo fonte, câmera, relógio, entrada para cartão de memória e *buzzer* para sinal de alerta, desenvolver o firmware aplicando visão computacional para identificar o estado dos olhos, medir a inclinação da visão do motorista e avaliar a acurácia e o tempo de resposta do sistema.

2 REVISÃO BIBLIOGRÁFICA

Para o desenvolvimento de soluções robustas e de resposta em tempo real, é necessário conhecer os conceitos de sistemas embarcados e *edge computing*, ou computação de borda, que significa que a solução desenvolvida será uma placa eletrônica dedicada e baseada na necessidade de processar dados localmente e em tempo real, ou seja, situações em que a transmissão de dados a um Datacenter para processamento causa níveis inaceitáveis de latência (RED HAT, 2021).

2.1 SISTEMAS EMBARCADOS

Um sistema embarcado é a combinação entre software, hardware e outras partes adicionais. Ele possui memória, processador, dispositivos de armazenamento e difere de um computador de uso geral por executar bem uma única tarefa, de maneira contínua e em muitas vezes, sem falhas ou panes, podendo ter alguma rede de comunicação com o exterior (MORIMOTO, 2007).

Os sistemas embarcados possuem algumas características específicas que os diferenciam de outros sistemas computacionais (VAHID; GIVARGIS, 2002):

Monofuncional: um sistema embarcado geralmente executa apenas um programa, repetidamente, ao contrário de um sistema de computador, que executa várias tarefas conforme o usuário instala novos programas.

Restrições de projeto: todo sistema de computador tem restrições em suas métricas de projeto, porém, em sistemas embarcados, essas limitações são maiores. Os sistemas normalmente são de baixo custo, precisam ter desempenho rápido o suficiente para processamento em tempo real e baixo consumo de energia para maximizar a vida útil da bateria;

Robustez: muitos dos sistemas embarcados devem reagir às mudanças no ambiente e processar dados em tempo real onde o sistema não pode, por exemplo,

perder dados ou simplesmente parar de funcionar. Um exemplo típico é um marca-passo que nunca deve parar de enviar sinais de pulso ao coração de um determinado paciente.

Para que o sistema embarcado execute a tarefa conforme programado, o sistema deve possuir chips eletrônicos como microcontroladores ou microprocessadores sua arquitetura.

2.2 MICROPROCESSADORES E MICROCONTROLADORES

Vários fabricantes de processadores comercializam dispositivos especificamente para o domínio de sistemas embarcados. Esses dispositivos podem incluir diversos recursos. Primeiro, eles podem incluir vários dispositivos periféricos, como temporizadores, conversores analógico-digital e dispositivos de comunicação serial no mesmo chip que o processador. Em segundo lugar, eles podem incluir um conjunto de instruções pré armazenado em sua memória. Terceiro, eles podem fornecer ao programador acesso direto a vários pinos de hardware. Quarto, eles podem fornecer instruções especializadas para operações comuns de sistema embarcado, como operações de manipulação de bits. Um microcontrolador é um dispositivo que possui alguns ou todos esses recursos (VAHID; GIVARGIS, 2002).

A incorporação de periféricos e memória no mesmo chip reduz o número de componentes externos necessários, resultando em implementações compactas e de baixo consumo de energia. Fornecer acesso por pinos de I/O permite que os programas monitorem os sensores, definam atuadores e transfiram dados com outros dispositivos. O fornecimento de instruções especializadas melhora o desempenho de aplicativos de sistemas embarcados.

Existem diversos tipos de arquiteturas de microcontroladores no mercado, como o PIC, AVR, ARM, 8051 e o ESP32. Assim como em placas eletrônicas com os circuitos mínimos necessários, conhecidas como “*Shields*”, como é o caso do ESP32-WROOM-32, que foi utilizado neste projeto por possuir os periféricos necessários já integrados em uma mesma placa.

2.3 MICROCONTROLADOR ESP32WROOM32

O ESP32-WROOM-32 é um módulo genérico, com Wi-Fi e Bluetooth integrado, que visa uma ampla variedade de aplicações, desde redes de sensores de baixa potência até as tarefas mais exigentes, como codificação de voz, *streaming* de música e decodificação de MP3.

No núcleo deste módulo está o chip ESP32-D0WDQ6 que é projetado para ser escalonável e adaptável. Existem dois núcleos de CPU que podem ser controlados individualmente, sendo a frequência de *clock* da CPU ajustável de 80 MHz a 240 MHz. O chip também possui um coprocessador de baixo consumo de energia que pode ser usado no lugar da CPU para economizar energia enquanto executa tarefas que não requerem muita capacidade de computação como monitoramento de periféricos. O ESP32 integra um variado conjunto de periféricos, como sensores de toque capacitivos, sensores Hall, interface *SDcard*, Ethernet, SPI de alta velocidade, UART, I²S e I²C (ESPRESSIF SYSTEMS, 2021).

2.4 INTELIGÊNCIA ARTIFICIAL

A inteligência artificial é um agrupamento de várias tecnologias, como redes neurais artificiais, algoritmos, sistemas de aprendizado, entre outros que conseguem simular capacidades humanas ligadas à inteligência. Por exemplo, o raciocínio, a percepção de ambiente e a habilidade de análise para a tomada de decisão.

2.4.1 Aprendizagem de máquina

O Aprendizado de Máquina (*“Machine Learning”*) é um dos principais pilares da nova era industrial, pois permite a extração de informação utilizando dados de forma eficiente e eficaz. Do ponto de vista da eficiência, é alavancada pelos dispositivos de baixo custo voltados à computação de alto desempenho. Já a eficácia depende da qualidade e quantidade dos dados disponíveis e dos modelos de aprendizagem (FREITAS, A. L.; SANTANA JUNIOR. O. V, 2019).

O objetivo do Aprendizado de Máquina é reconhecer padrões a partir das entradas e saídas de dados de um sistema, para que o mesmo seja capaz de tomar

decisões de forma independente ou com a menor intervenção humana possível.

Os algoritmos de aprendizado de máquina podem ser classificados em aprendizado supervisionado, e aprendizado não supervisionado (PEREZ, 2017). Segundo Simon Haykin (2008), existe ainda uma subcategoria do aprendizado não supervisionado chamada de aprendizado por reforço.

De modo geral, a aprendizagem não supervisionada envolve a observação de vários exemplos de um vetor aleatório tentando aprender de maneira direta ou indireta a distribuição de probabilidade ou propriedades interessantes dessa distribuição; enquanto aprendizagem supervisionada envolve observar várias amostras de um vetor (x) e um valor complementar (y), aprender o padrão e conseguir prever o resultado para as novas amostras (GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A, 2016).

O termo aprendizagem supervisionada origina-se da observação de um conjunto de dados e seus resultados na busca por um padrão, assim, com a ajuda de uma pessoa, é treinado o algoritmo para que saiba o que o padrão encontrado significa. Já na aprendizagem não supervisionada, o algoritmo deve aprender a dar sentido aos dados sem que haja a interação humana (GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A, 2016).

2.4.2 Redes Neurais e Aprendizagem profunda

As redes neurais artificiais foram inspiradas pela maneira como o cérebro humano calcula as informações. O cérebro é um computador altamente complexo que pode organizar seus componentes estruturais (chamados neurônios) para realizar certos cálculos, como reconhecimento de padrões, percepção e controle de movimento. Geralmente é mais rápido do que os melhores computadores disponíveis hoje. De modo geral, uma rede neural é uma máquina projetada para simular o caminho que são percorridos no cérebro para uma determinada tarefa (HAYKIN, 2008).

As redes neurais artificiais são compostas por neurônios artificiais que processam e trocam informações. O primeiro neurônio artificial foi desenvolvido por Frank Rosenblatt. Esse neurônio recebe informações, processa e envia o resultado booleano para a próxima camada de neurônios a ele conectada (NIELSEN, 2015).

Após o ano de 2006, técnicas de aprendizado profundo, chamadas de *deep learning*, e modelagem de redes neurais profundas conseguiram melhorar o desempenho em muitos problemas importantes de visão computacional (NIELSEN, 2015). Áreas como mercado de ações, websites adaptativos, jogos de estratégia, detecção de fraude virtual, classificação de imagens, reconhecimento de fala e processamento de linguagem natural utilizam de redes neurais profundas e *deep learning* em suas soluções.

O reconhecimento facial é um sistema biométrico de funcionamento similar aos sistemas de reconhecimento pela impressão digital e ou pela retina. Isso porque cada pessoa possui um rosto único. A grande vantagem da utilização biométrica pelo reconhecimento facial é fato de ela ser uma técnica natural, não intrusiva e de fácil utilização (LI; JAIN, 2005).

2.5 VISÃO COMPUTACIONAL

De acordo com Gary Bradski e Adrian Kaebler, visão computacional é a transformação de dados de uma câmera fotográfica ou de vídeo em uma decisão ou em uma nova representação. Todas essas transformações são feitas para atingir algum objetivo específico. Os dados de entrada podem incluir algumas informações contextuais, como “a câmera está montada em um carro” ou “o visor de alcance do laser indica que um objeto está a 1 metro de distância”. (BRADSKI; KAEHLER, 2008).

Os sistemas de visão computacional seguem um conjunto predefinido de etapas, a saber: aquisição, pré-processamento, segmentação, representação e descrição, reconhecimento e interpretação (GONZALEZ, WOODS, 2008).

No que se refere ao sensoriamento, são necessários dois elementos para a coleta da imagem digital. O primeiro é um dispositivo físico sensível à energia irradiada pelo objeto cuja imagem deseja-se capturar. O segundo é o digitalizador, que é responsável por converter a saída do dispositivo físico em formato digital.

Em geral, a aquisição da imagem já envolve a etapa de pré-processamento. Nessa fase é realizada a alteração de propriedades, como o tamanho ou correções de gama. A etapa de segmentação é a etapa que mais influência no resultado, pois é nela que a imagem é transformada em uma representação mais significativa e mais fácil de analisar.

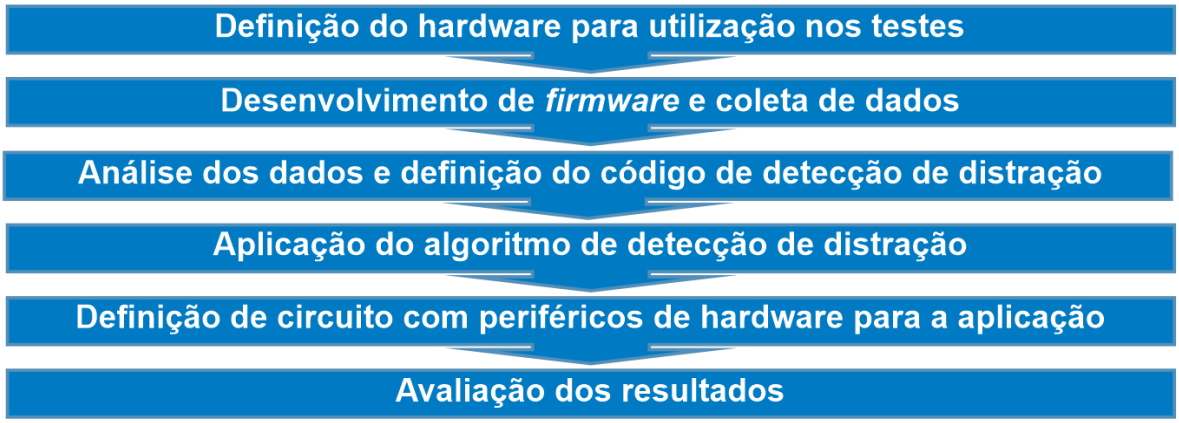
O processo de representação da imagem significa formatá-la como uma matriz de valores numéricos para ser interpretada. Já a descrição, é conhecida como etapa de seleção das características, responsável por extrair os principais atributos que resultam em alguma informação quantitativa de interesse ou que possam ser utilizadas para diferenciar uma classe de objetos de outra (GONZALEZ, WOODS, 2008).

O reconhecimento é o processo que atribui um rótulo para os descritores e a interpretação é a análise dos resultados para a tomada de decisão. Para a análise dos resultados é necessária a utilização de métricas de acordo com o modelo usado. Métricas como precisão, acurácia e revocação são utilizadas neste contexto.

3 PROCEDIMENTOS EXPERIMENTAIS

Este trabalho foi organizado em 6 etapas para seu desenvolvimento, conforme observado na Figura 1.

Figura 1: Etapas de desenvolvimento



Fonte: Do autor (2022).

3.1 DEFINIÇÃO DO HARDWARE PARA UTILIZAÇÃO NOS TESTES

Nesta etapa inicial, foi definido o circuito eletrônico com os periféricos (partes para comunicação com o meio externo) necessários para o hardware, que pudesse tornar possível a realização dos procedimentos experimentais.

Considerando as funcionalidades para realização dos experimentos, o

hardware necessitava ter os seguintes dispositivos:

- Microcontrolador;
- Câmera embarcada;
- Periférico para visualização da imagem em tempo real para verificação da posição de instalação no veículo;
- Periférico para registro das imagens e eventos;

Para a aplicação em veículos, será necessário também dos seguintes itens:

- Conversor DC/DC para atender a tensão de alimentação dos veículos;
- Periférico visual e sonoro para alerta do motorista (diodo emissor de luz e *buzzer*);
- Periférico para contagem de tempo;
- Periférico para acionamento de luz infravermelha para modo noturno, com o objetivo de iluminar o rosto do motorista.

Foi designado o chip ESP32 do fabricante Espressif como microcontrolador principal, por possuir maiores memórias RAM, ROM e Flash, velocidade de processamento, módulo Wi-Fi integrado, ferramentas de desenvolvimento gratuitas, código aberto e bibliotecas de *machine learning* disponibilizada pelo próprio fabricante do chip, além de possuir preço acessível, conforme apresentado no Quadro 1.

O Quadro 1 faz comparação entre o ESP32 e outros dispositivos como ESP8266 (linha com menos recursos do próprio fabricante), Arduino UNO e Arduino MEGA que são placas de desenvolvimento utilizadas na academia para ensino e prototipagem.

Quadro 1: Comparação entre chips

Descrição	**Arduino UNO**	**Arduino MEGA 2560**	**ESP32**	**ESP8266**
Alimentação	5V	5V	2,2V ~ 3,3V	2,2V ~ 3,3V
VIN	7 ~ 12V	7 ~ 12V	5 ~ 9V	5 ~ 9V
Corrente de Consumo:	Média de 15mA	Média de 70mA	Média de 80mA	Média de 80mA
Temperatura de Operação:	-40°C ~ +85°C	-40°C ~ +85°C	-40°C~ +85°C	-40°C ~ +85°C
Processador:	AVR® 8-bit RISC	ATmega2560 RISC com até 16 MIPS	Xtensa® Dual-Core 32-bit LX6	Tensilica® L106 ultra-low power 32-bit
Frequência de Operação:	0 ~ 16 MHz	0 ~ 16MHz	80MHz~240MHz	80MHz~160MHz
FLASH:	32KB	256 KB	4MB	4MB
RAM/SRAM:	2KB	8 KB	520KB	36kB
ROM/EEPROM	4KB	4 KB	448KB	64KB

Pinos de I/O:	23 pinos com 6 PWM	54 pinos com 14 PWM	34 pinos com 16 PWM	13 pinos com 9 PWM
Conversores ADC	6 ADC de 10 bits	16 ADC de 10 bits	18 ADC de 12 bits	1 ADC de 10 bits
Conversores DAC	Nenhum	Nenhum	2 DAC	Nenhum
WiFi:	Somente com Shield	Somente com Shield	2,4 GHz, 802.11 b/g/n/e/i	2.4GHz 802.11 b/g/n
Bluetooth:	Não Possui	Não Possui	Bluetooth Low Energy v4.2	Não Possui
Preço Médio (Modelo Original)	R$69,99	R$99,99	R$38,90	R$31,00

Fonte: OLIVEIRA (2021).

Para a precificação no cenário de 2022, foram convertidos os preços considerando a conversão de um dólar para cinco reais. Seguem os preços atualizados:

- Arduino UNO: R$ 110,00;
- Arduino MEGA 2560: R$ 192,50;
- ESP32: R$ 74,75;
- ESP8266: R$ 29,95;

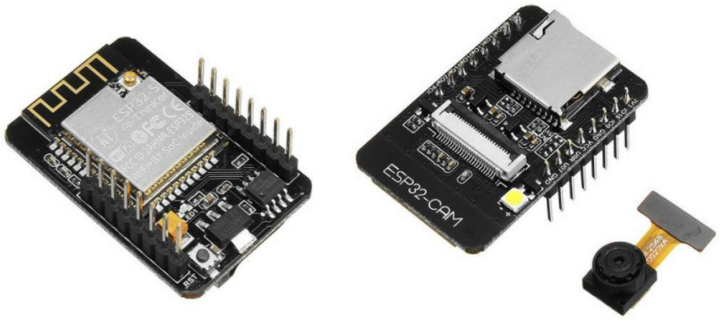
Nota-se a grande diferença do ESP32 relacionada a capacidade de memória, tanto volátil como não volátil, processamento e periféricos de comunicação, sendo o definido para o projeto.

Foram considerados apenas placas eletrônicas com sistema de tempo real. Outras como microcomputadores, por exemplo o *Raspiberry Pi*, foram desconsideradas neste trabalho devido ao fato de usarem sistema operacional não determinístico e possuir o código de execução no cartão SD, sendo fácil o corrompimento dos dados e assim parar de funcionar.

Para o desenvolvimento do trabalho, foi utilizada a placa ESP32-CAM do fabricante Ai-thinker (Figura 2), pois a mesma possui o chip ESP32, câmera e encaixe para cartão de memória em uma única placa.

Para a definição da câmera, foi optado pelo modelo OV2640 do fabricante Ominivision, pois vem com o módulo ESP-CAM32 utilizado nos testes. Ela possui resolução 2 Megapixel conseguindo imagens de 1632 x 1232 pixels e possui tensão de alimentação e periférico de comunicação compatíveis com o ESP32.

Figura 2: Placa de desenvolvimento.



Fonte: GUSE (2019).

3.2 DESENVOLVIMENTO DO FIRMWARE E COLETA DE DADOS

Para os testes aplicando visão computacional e *deep learning*, foi utilizada a biblioteca ESP-DL disponibilizada pelo fabricante do chip. A mesma fornece recursos de aprendizado profundo de alto desempenho dedicados a ESP32, como inferência de Rede Neural (NN), processamento de imagens e operações matemáticas.

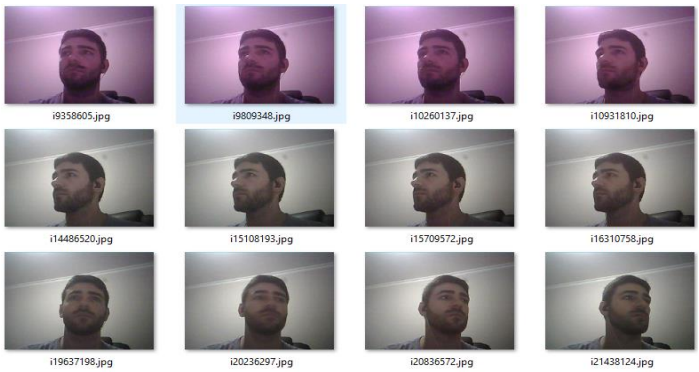
O projeto ESP-DL possui um exemplo de detecção de face humana. A entrada para esta interface é uma imagem estática, e os resultados da detecção são pontuações de confiança e valores de coordenadas dos olhos, nariz e boca. O exemplo usado foi o “*human_face_detect*”.

Utilizando a placa ESP32-CAM, foi desenvolvido um algoritmo capaz de tirar fotos usando a câmera OV2640 para então aplicá-las no algoritmo de detecção e posteriormente salvar no cartão de memória as imagens e os resultados.

Para adequar a imagem ao tipo de entrada esperado pelo algoritmo de detecção, foi necessário configurar o tipo de imagem da câmera para o tipo RGB565, convertê-la para JPG e salvá-la no cartão de memória, sendo assim possível visualizar as imagens em um computador.

Foram coletadas 100 amostras de imagens com várias posições de rostos e variação de luminosidade. Em seguida as amostras foram classificadas individualmente como válida ou distração. Parte dos dados coletados são apresentados na Figura 3.

Figura 3: Dados de imagens e pontos faciais coletados.



Fonte: Do autor (2022).

3.3 ANÁLISE DOS DADOS E DEFINIÇÃO DO CÓDIGO DE DETEÇÃO DE DISTRAÇÃO

Utilizando a linguagem Python no ambiente colaborativo da Google, o Google Colab, foram realizadas todas as análises nos dados e a validação dos resultados, comparando, assim, com técnicas de *machine learning*.

A partir dos pontos dos olhos nariz e boca, foi calculada a distância euclidiana e ângulos entre o olho direito e a boca, e entre os dois pontos da boca com o nariz. Com isso foi possível criar limites de inclinação do rosto. Os resultados foram salvos em um conjuntos de dados (‘*dataset*’) e, então, usando a biblioteca de visualização de imagem, foi possível imprimir os pontos e as linhas geradas na imagem. A Figura 4 apresenta uma parte do código desenvolvido na linguagem Python juntamente com o resultado aplicado a uma imagem. Observa-se que o código gera pontos de identificação dos olhos, nariz e canto da boca. A partir destes pontos de indicação é possível realizar a verificação da distração do usuário.

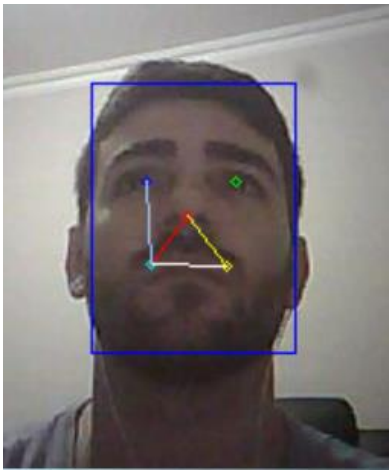
Figura 4: Código para cálculo de ângulo e distância euclidiana e imagem resultado.

```python
def distance_between(p1,p2):
    dx = p2[0] - p1[0]
    dy = p2[1] - p1[1]
    distance = int(math.sqrt( dx**2 + dy**2 ))
    return distance

def angle_between(p1,p2):
    dx = p2[0] - p1[0]
    dy = p2[1] - p1[1]

    angle = 0
    if p2[0] != p1[0]:
        angle= round((math.atan2(dx, dy)*180)/math.pi, 2)
        # if angle>=180 :
        #    angle = angle-180
        if angle<0 :
           angle = -(angle)
    if p2[0] == p1[0]:
        angle = 90

    return angle
```



Fonte: Do autor (2022).

A próxima etapa executada foi a separação dos dados em duas categorias: de treinamento e de teste, sendo que foram utilizados70% dos dados para treinamento e 30% para teste.

Baseando-se nos resultados aplicados com *machine learning*, e que todos obtiveram resultados superiores a 75%, o algoritmo definido para implementação no hardware foi a utilização apenas dos ângulos relativos entre os pontos, pois apresentou resultados bons e a lógica foi implementada no hardware sem maiores dificuldades. Foi desenvolvido o algoritmo em Python e comparado os resultados no Quadro 2, que apresenta uma relação entre os algoritmos executados e suas respectivas acurácias.

Quadro 2: Resultados dos algoritmos

Algoritmo executado	**Acurácia**
sklearn.naive_bayes. GaussianNB	96,70 %
sklearn.neural_network.MLPClassifier	75,00 %
sklearn.svm.SVC	91,90 %
sklearn.linear_model.LogisticRegression	91,99 %
sklearn.tree.DecisionTreeClassifier	87,50 %
Algoritmo de ângulo entre os pontos	80,19 %

Fonte: Do autor (2022).

A Figura 5 apresenta a função na linguagem de programação Python criada para a detecção a partir dos pontos na imagem, utilizando a inclinação relativa dos olhos, nariz e boca. O cálculo de distância euclidiana não foi utilizado pois os valores mudavam dependendo da distância entre o rosto e a câmera.

Figura 5: Algoritmo de detecção por ângulo na linguagem Python

```python
def detect(angle_nose_mouth_l, angle_nose_mouth_r, angle_eye_l_mouth_l):
  if (angle_nose_mouth_l < 25) or (angle_nose_mouth_r < 25):
    return False
  elif (angle_nose_mouth_l < 30) and (angle_nose_mouth_r < 30):
    return False
  elif (angle_eye_l_mouth_l > 10) and (angle_eye_l_mouth_l < 80):
    return False
  else:
    return True
```

Fonte: Do autor (2022).

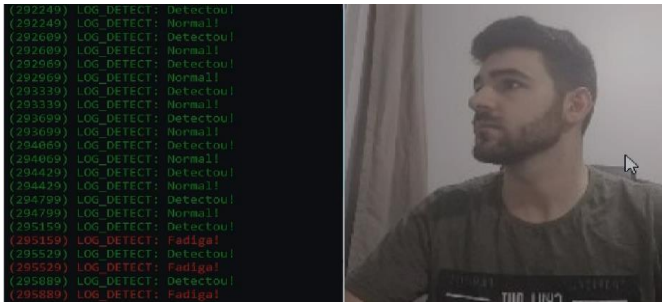
3.4 APLICAÇÃO DO ALGORITMO DE DETECÇÃO DE DISTRAÇÃO

Após a definição do algoritmo para implementação no hardware, foi necessário transcrever o algoritmo de Python (Figura 5) para a linguagem C, linguagem de programação usada no microcontrolador ESP32.

Nessa etapa foi adicionada a lógica de funcionamento já voltada para a aplicação que define o tempo de 3 segundos para considerar uma distração. Esse tempo é configurado como um parâmetro na aplicação final. A deteção acontece quando é observada a inclinação da cabeça ou a falta de deteção de face. Foi utilizada a comunicação serial e um LED da placa de testes para apresentar o resultado das deteções de distração.

Como não foi possível visualizar o *frame* em tempo real através do serviço Wi-Fi para que os resultados não fossem influenciados pelo serviço de Wi-Fi, foi posicionado um celular ao lado da câmera do ESP32 e assim foi possível verificar o funcionamento. A Figura 6 apresenta o log com a mensagem “Distração“ no momento em que o motorista olha para o lado, exibindo em vermelho no log a mensagem de deteção.

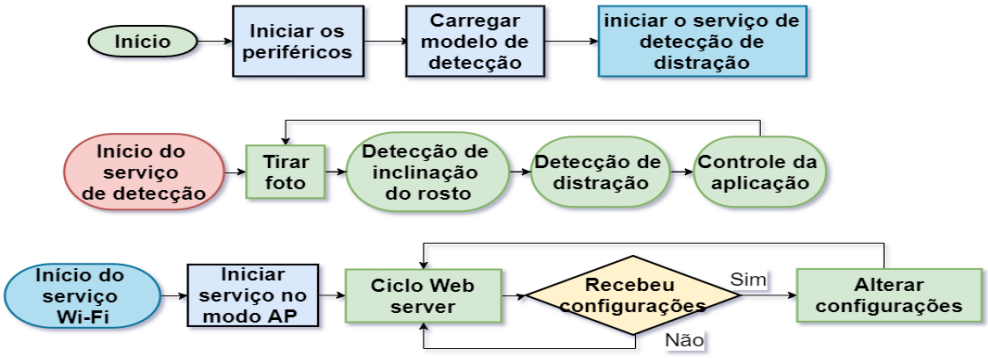
Figura 6: Algoritmo em funcionamento no hardware



Fonte: Do autor (2022).

O fluxo geral do sistema de detecção de distração é apresentado na Figura 7.

Figura 7: Fluxo geral do sistema.



Fonte: Do autor (2022).

Como principais etapas no desenvolvimento do *firmware,* tem-se:

- Utilização da câmera e cartão de memória: foi configurado a comunicação serial (SPI) para comunicação com estes periféricos.
- Registro dos dados na memória flash: o ESP32 possui uma memória flash de 4 MB, na qual é usada para salvar o código do firmware e dados de configuração da câmera para não se perderem após um desligamento da aplicação.
- Detecção dos olhos e inclinação da face: com o algoritmo ESP-DL da Espressif foi possível detectar os olhos, nariz e boca em uma imagem, entretanto, o único formato suportado foi o RGB565, sendo necessário converter para o formato JPG para ser visível no serviço web local e ser salvo no cartão de memória.
- Algoritmo de detecção de distração: criação de lógica para identificar inclinação da face e identificar uma possível distração, foi colocado uma proteção de 3 segundos para ligar o buzzer.
- Utilização do módulo Wi-Fi para visualização da imagem: este modo é usado apenas para testes, pois o ESP não apresentou memória suficiente para fornecer os frames para os dois serviços (detecção e servidor web local).

3.4.1 Alteração da câmera infra vermelha

O sistema deve funcionar tanto em ambientes iluminados quanto em ambientes escuros. Por este motivo, foi necessário remover o filtro infravermelho da

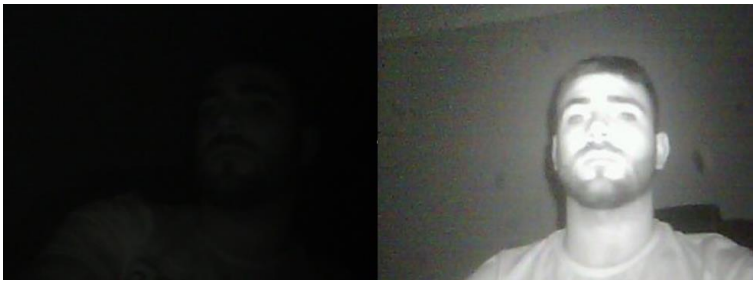
lente para a câmera conseguir captar as luzes infravermelhas. Também foi necessário adaptar um canhão de LED infravermelho (pois a luz infravermelha não é visível ao olho humano) nos testes para que fosse possível iluminar o rosto sem atrapalhar o motorista. Esta adaptação do hardware é apresentada na Figura 8.

Figura 8: Alteração do hardware para remover o filtro infravermelho

Fonte: Do autor (2022).

A Figura 9 mostra a comparação entre imagens antes e depois da adição do canhão de luz infravermelho, nota-se melhoras significativas na imagem, que antes possuía um rosto imperceptível, passou a ser perceptível e, consequentemente, detetável na aplicação.

Figura 9: Imagem com e sem filtro infravermelho na lente



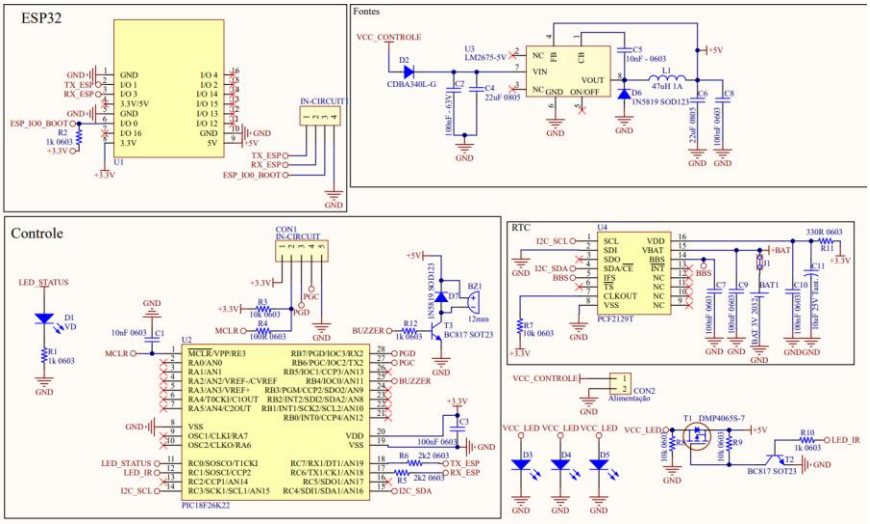
Fonte: Do autor (2022).

3.5 DEFINIÇÃO DE CIRCUITO DE HARDWARE PARA A APLICAÇÃO

Baseado nos requisitos do projeto, foi desenvolvido um esquema eletrônico e uma placa eletrônica no software Altium Designer. A placa possui os periféricos para atender a aplicação final, contendo relógio de tempo real interno, bateria CR2032, LEDs infravermelhos, LED de status do sistema, buzzer, fonte, microcontrolador secundário para expansão de entradas e saídas e conectores de gravação, além do

ESP32-CAM com a câmera e entrada para cartão SD. A Figura 10 apresenta o esquemático final de desenvolvimento do sistema.

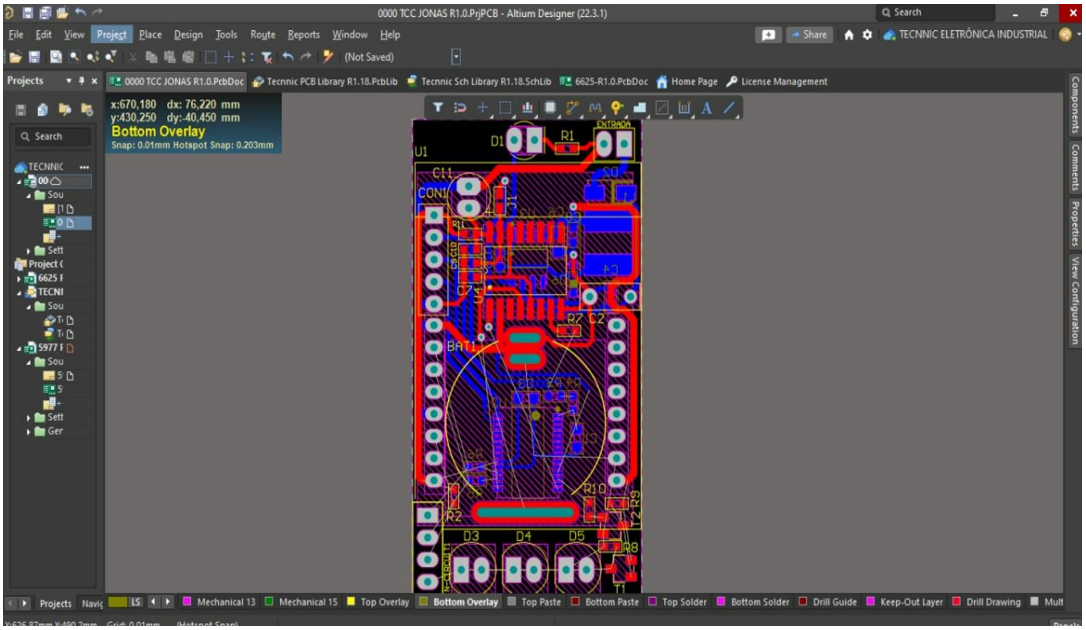
Figura 10: Esquemático da placa final da aplicação.



Fonte: Do autor (2022).

A Figura 11 mostra uma captura de tela do layout ainda em desenvolvimento no Altium Designer. A placa eletrônica possui dimensões de 30x60 mm.

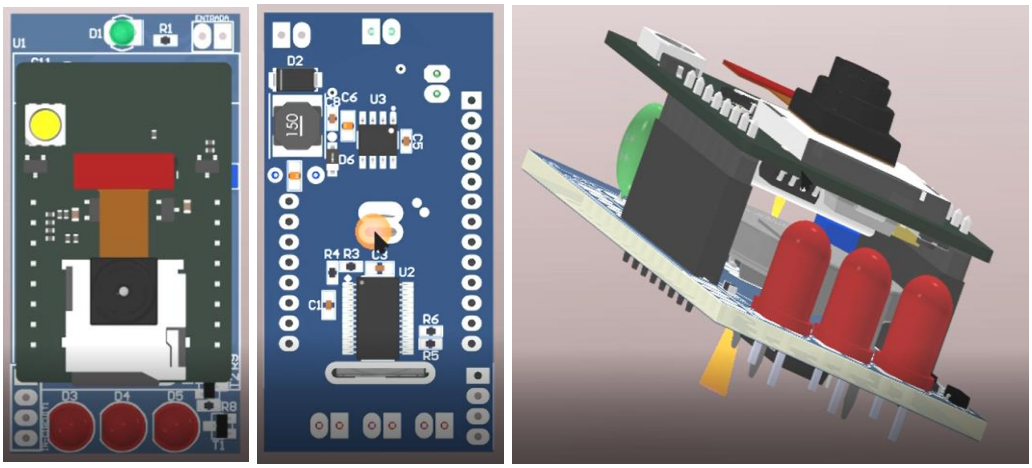
Figura 11: Placa de circuito impresso em visão 2D no Altium Designer.



Fonte: Do autor (2022).

Já a Figura 12 apresenta uma visão 3D da placa em desenvolvimento. Esta visualização 3D é fornecida pelo próprio software de desenvolvimento do esquemático e do layout.

Figura 12: Placa de circuito impresso em visão 3D no Altium Designer.



Fonte: Do autor (2022).

4 RESULTADOS E DISCUSSÕES

4.1 MEDIÇÃO DOS TEMPOS DE RESPOSTA COM TAMANHOS DE IMAGENS

Com o hardware e o software desenvolvido, iniciaram-se os testes de validação da solução proposta. O primeiro teste realizado foi a medição dos tempos de resposta com tamanhos de imagens. O objetivo deste teste foi definir a configuração de tamanho de imagem para que o tempo de resposta do sistema atendesse a aplicação final.

Para o teste foram utilizadas 5 amostras de imagens de cada tamanho configurado. Em todas as amostras o motorista estava olhando para a frente. Para cada amostra foi registrado o tempo de aquisição e de salvamento no cartão de memória.

Foi desenvolvido um programa de teste para mensurar o tempo de aquisição, detecção com a biblioteca ESP-DL e salvamento das imagens no cartão SD. Foram utilizados os tamanhos de imagem HQVGA (240 x 176), QVGA (320 x 240), CIF (400 x 296), HVGA (480 x 320) VGA (640 x 480), SVGA (800 x 600), disponíveis no modelo de câmera definido. O Quadro 3 abaixo apresenta os resultados obtidos.

Quadro 3: Tempos de resposta com tamanhos diferentes de imagem

Tamanho da imagem (px)	Tempo de aquisição (ms)	Tempo de Deteção (ms)	Tempo de Salvamento (ms)
HQVGA (240 x 176)	150 - 200	350 – 400	400 - 600
QVGA (320 x 240)	140 – 250	250 - 400	400 - 600
CIF (400 x 296)	180 – 250	300 – 400	650 – 850
HVGA (480 x 320)	750 - 1000	330 - 450	650 – 850
VGA (640 x 480)	800 - 1100	528 – 800	650 – 850
SVGA (800 x 600)	1000 – 1200	700 – 900	650 – 850

Fonte: Do autor (2022).

Nota-se que as imagens VGA e SVGA demoram muito tempo para serem processadas no microcontrolador, tendo um tempo de resposta muito longo para a aplicação. Já a imagem na resolução HQVGA, por ser pequena, dificulta a deteção do rosto. A resolução CIF apresentou bons resultados quanto ao tempo total de processamento, sendo a designada para a aplicação. Já o tempo de salvamento no cartão de memória é alto devido ao processamento do sistema de arquivos do tipo FAT.

4.2 TESTES PARA RECONHECIMENTO DE DISTRAÇÃO

Também foram realizados testes práticos em bancada e em um veículo em movimento para validar o funcionamento do algoritmo de detecção de distração. Estes testes foram realizados simulando as situações de atenção e distração ao dirigir.

As posições da visão do motorista variaram entre olhar para a frente, para cima, baixo, esquerda e direita; sempre por mais de 3 segundos para indicar distração. O Quadro 4 apresenta o resultado dos testes para diferentes posições dos olhos.

Quadro 4: Teste de funcionamento

Posição dos olhos	Resultado	Observação
Olhando para a frente	OK	-
Olhando para cima	Distração	-
Olhando para baixo	Distração	-
Olhando para esquerda	Distração	-
Olhando para direita	Distração	-
Sem rosto por 3 segundos	Distração	Em alguns *frames*, a rosto não foi detectado, então foi considerado como distração.

Fonte: Do autor (2022).

Os testes foram replicados de noite com o auxílio do LED infravermelho para iluminação da face do motorista. Os resultados são mostrados no Quadro 5. Foram coletadas 20 amostras para cada posição dos olhos.

Quadro 5: Teste de funcionamento no escuro

Posição dos olhos	Resultado	Observação
Olhando para a frente	OK	-
Olhando para cima	30% Falhas	Apresentou falhas de detecção do rosto.
Olhando para baixo	30% Falhas	Apresentou falhas de detecção do rosto.
Olhando para esquerda	Distração	-
Olhando para direita	Distração	-
Sem rosto por 3 segundos	Distração	Em alguns frames, a rosto não era detectado, então foi considerado como distração.

Fonte: Do autor (2022).

É possível perceber que durante a noite, casos em que a imagem torna-se escala de cinza por falta de iluminação, o algoritmo tem dificuldade em encontrar os pontos faciais. Porém com a coleta de maior quantidade de *frames* é possível tratar essas falhas, trabalhando com a troca de resolução de imagem e lógicas de temporização para evitar deteções falsas.

Foi considerado também que, se não foi detetado um rosto na imagem, o sistema considerará como distração e, se manter a falha por mais de 3 segundos, o motorista será avisado através do buzzer e LED.

4.3 PRINCIPAIS FATORES QUE INFLUENCIARAM NOS RESULTADOS

A partir dos testes realizados, pode-se destacar cinco pontos que influenciados diretamente nos resultados:

- *Tamanho da imagem*: quanto maior o dado de entrada, maior era o tamanho da memória RAM alocada do chip e maior o tempo de análise.
- *Tempo de detecção*: por ser um algoritmo complexo, a detecção se tornou lenta para o processamento no microcontrolador, sendo necessário diminuir o tamanho da imagem de entrada e consequentemente reduzir o tempo de processamento.

- *Tempo de salvamento*: conforme o tamanho do arquivo e a quantidade de dados já existentes no cartão, o tempo de salvamento variou, não podendo ser aplicado na aplicação final frequentemente. Assim, o algoritmo irá salvar a imagem apenas ao detectar distração por 3 segundos.
- *Luminosidade*: foi observado que a luminosidade das imagens alterou os resultados. Em imagens bem iluminadas e colorizadas o algoritmo de detecção apresentou melhores resultados. Já em ambientes escuros, apresentou dificuldade em identificar os pontos do rosto. Com a instalação do LED infravermelho e alteração da lente da câmera foi possível a validação em ambientes escuros também.
- *Serviços em paralelo (web local)*: o serviço pode ser executado em paralelo, porém apenas para configuração dos parâmetros da câmera como o tamanho e qualidade. No modo de visualização da imagem, o serviço de detecção parou de funcionar por falta de memória. A possível maneira de contornar esse problema é desenvolver um serviço de foto a cada 500 milissegundos apenas para auxiliar na instalação do equipamento, por exemplo.

5 CONCLUSÃO

O projeto utilizando o ESP32-CAM apresentou bons resultados, podendo ser aplicado em soluções de visão computacional embarcadas com lógicas simples. O algorítmo apresentado de detecção por inclinação apresentou vulnerabilidades quanto ao uso de óculos escuro e partes do rosto tampada, sendo necessário rever essa abordagem para uma aplicação futura. Se a aplicação necessitar de execução de mais de um modelo de detecção como a identificação de motorista, detecção de fumaça, detecção de uso do celular etc, será necessário definir um microcontrolador com maior poder de processamento. Neste caso, o ESP32-CAM pode ser utilizado apenas como coletor de imagem e acionamento dos periféricos já que através do Bluetooth ou Wi-Fi pode ser integrado com *smartphones* ou outros equipamentos.

Este projeto também ajuda a enfatizar a dificuldade em utilizar Visão Computacional e *Machine Learning* em sistemas embarcados de baixo custo e de tempo real, onde as limitações de hardware, tais como processamento e memória, afetam os tempos de resposta do sistema, podendo até comprometer a sua confiabilidade.

A integração de conhecimento adquirido durante a graduação de Engenharia de Computação, foi crucial para o desenvolvimento da pesquisa por meio dos conceitos e práticas realizadas nas disciplinas do curso. Dentre as disciplinas, cita-se: Sistemas Embarcados, Circuitos Eletrônicos, Visão Computacional, Machine Learning, IoT Aplicada, e Soluções WEB as quais foram de vital importância para este projeto.

Para continuação deste projeto, o desenvolvimento de uma caixa plástica através de manufatura aditiva e a fabricação do protótipo são necessárias, bem como a alteração do firmware para embarcar no MVP, possibilitando a criação de um lote piloto para testes em alta escala.

Como pontos de melhorias, o estudo de um algoritmo de *deep learning* diferente do utilizado pode ser realizado, como o TensorFlow Lite otimizado para o ESP32, a criação de modelos de detecção de pontos faciais e mensurar os tempos de processamento do sistema a fim de melhorar o tempo de resposta da aplicação também podem ser estudados. Além de um chip alternativo com maior processamento para aplicar algoritmos mais complexos ao sistema e possibilitar novos recursos, porém ainda se faz necessário que o chip seja um sistema de tempo real.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE MEDICINA DE TRÁFEGO (ABRAMET). **ONU define período de 2021 a 2030 como Segunda Década de Ação pela Segurança no Trânsito**, 2021. Disponível em: <https://abramet.com.br/noticias/onu-define-periodo-de-2021-a-2030-como-segunda-decada-de-acao-pela-seguranca-no-transito/>. Acesso em: 29 de agosto de 2021.

ATLAS DA ACIDENTALIDADE NO TRANSPORTE BRASILEIRO. **Distribuição dos acidentes por causa, valores absolutos**. 2021. Disponível em: <https://atlasacidentesnotransporte.com.br/consulta?grafico=acidente#graph>. Acesso em: 13 de setembro de 2021.

BRADSKI, Gary. KAEHLER, Adrian. **Learning OpenCV**. Ed. O'Reilly Media, Inc. 2008.

CÓDIGO DE TRÂNSITO BRASILEIRO – CTB – LEI Nº 9.503, de 23 de setembro de 1997.

ECKERMANN, Erik. **World History of the Automobile**. SAE Press. 2001. p.14.

ESPRESSIF SYSTEMS. **ESP32WROOM32 Datasheet**. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>. Acesso em: 01 de outubro de 2021.

FREITAS, A. L.; SANTANA JUNIOR. O. V. Machine learning: desafios para um brasil competitivo. **Computação Brasil**, Porto Alegre, v. 39, p. 8, jan. 2019.

GONZALEZ, Rafael C; WOODS, Richard E. **Processamento de imagens digitais**. 2008.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. [S.l.]:MIT press, 2016.

GUSE, Rosane. **Câmera IP: Cuide do seu bebê com o ESP32-CAM**. FilipeFlop, 2019. Disponivel em: <https://www.filipeflop.com/blog/esp-32-camera-ip>. Acesso em: 04 de novembro de 2021.

HAYKIN, S. **Neural networks and learning machines**. McMaster University, Canadá: Pearson Prentice Hall, 2008.

LI, S. Z.; JAIN, A. K. **Handbook of face recognition**. Springer Science and Business Media,Inc., China, 2005.

LI, X.; SHI, Y. **Computer vision imaging based on artificial intelligence**. In:2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS). 2018. p.22–25.

MORIMOTO, P. C. E.. **Entendendo os sistemas embarcados**. Disponível em: <http://www.hardware.com.br/artigos/entendendo-sistemas-embarcados/>. Acesso em: 01 de outubro de 2021.

NIELSEN, M. A. **Neural networks and deep learning**. Nova York, NY, Estados Unidos: Determination Press, 2015.

OLIVEIRA, Jailson. **Arduino, ESP32 e ESP8266 – Comparação**. XProjetos, 2019. Disponivel em: <https://xprojetos.net/arduino-esp32-e-esp8266-comparacao/>. Acesso em: 02 de novembro de 2021.

PEREZ, R. Algoritmo backpropagation. **Programar, Edição 57 - Julho 2017**, PISAREVSKY, **V.haarcascade_frontalface_alt2.xml**. 2013. Disponível em: <https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_alt2.xml>.Acesso em: 30 de setembro de 2021.

RED HAT. **O que é edge computing?**. Red Hat, 2021. Disponivel em: < https://www.redhat.com/pt-br/topics/edge-computing/what-is-edge-computing#empresas-que-usam-a-edge-computing> Acesso em: 05 de junho de 2021.

SINDICATO NACIONAL DA INDÚSTRIA DE COMPONENTES PARA VEÍCULOS AUTOMOTORES (SINDIPEÇAS). **Relatório da Frota Circulante**, 2021. Disponível em: <https://www.sindipecas.org.br/sindinews/Economia/2021/RelatorioFrotaCirculante_Marco_2021.pdf>. Acesso em: 29 de agosto de 2021.

VAHID, Frank, GIVARGIS, Tony. **Embedded system design: a unified hardware/software introduction**. [S.l.]: John Wiley & Sons, Inc., 2002.

WAISELFISZ, J. J. **Mapa da violência 2013:acidentes de trânsito e motocicletas**. Rio de Janeiro: Centro Brasileiro de Estudos Latino-Americanos (Cebela), 2013.

WORLD HEALTH ORGANIZATION (WHO). **Global plan for the decade of action for road safety 2011-2020**. Genebra: WHO, 2011. Disponível em: <https://www.who.int/roadsafety/decade_of_action/plan/plan_english.pdf?ua=1>. Acesso em: 29 de agosto de 2021.