CENTRO DE PESQUISA E DESENVOLVIMENTO TECNOLÓGICO EM INFORMÁTICA E ELETROELETRÔNICA DE ILHÉUS
RESIDÊNCIA TECNOLÓGICA EM SISTEMAS EMBARCADOS

PRISCILA PEREIRA SUZART DE CARVALHO

PROJETO FINAL
Monitoramento de fadiga em motoristas de caminhão no transporte rodoviário de cargas

ILHÉUS - BAHIA
FEVEREIRO / 2025

CENTRO DE PESQUISA E DESENVOLVIMENTO TECNOLÓGICO EM INFORMÁTICA E ELETROELETRÔNICA DE ILHÉUS
RESIDÊNCIA TECNOLÓGICA EM SISTEMAS EMBARCADOS

PRISCILA PEREIRA SUZART DE CARVALHO

PROJETO FINAL
Monitoramento de fadiga em motoristas de caminhão no transporte rodoviário de cargas

Projeto final apresentado a Residência Tecnológica em Sistemas Embarcados do Centro de Pesquisa e Desenvolvimento Tecnológico em Informática e Eletroeletrônica de Ilhéus, como requisito parcial para obtenção do certificado de conclusão da primeira fase da residência.

Matrícula: TIC370101730

Instrutores: Prof. Wilton Lacerda
Prof. Ricardo Prates

Mentora: Profa. Aline Ramos

ILHÉUS - BAHIA
FEVEREIRO / 2025

RESUMO

O transporte rodoviário de cargas é um dos setores mais importantes para a economia brasileira, mas também um dos mais perigosos, com altos índices de acidentes relacionados à fadiga e sonolência dos motoristas. Este projeto propõe o desenvolvimento de um sistema embarcado para monitorar a fadiga em motoristas de caminhões, utilizando tecnologias de baixo custo e acessíveis. O sistema integra um microcontrolador RP2040, botões, display SSD1306, matriz de LEDs WS2812 e comunicação serial (UART) para coletar dados de anamnese e sinais vitais, como frequência cardíaca e pressão arterial. A partir desses dados, o sistema calcula uma pontuação de fadiga e emite alertas visuais (carinha feliz/triste) para indicar o nível de risco. O projeto foi validado por meio de testes na placa BitDogLab que confirmaram a eficácia na captura de respostas, processamento de dados e exibição de resultados. A implementação desse sistema visa reduzir acidentes, melhorar a segurança viária e otimizar os custos operacionais das empresas de transporte, promovendo um ambiente de trabalho mais seguro e saudável para os motoristas.

Palavras-chave: Sistema embarcado; Monitoramento de fadiga; Microcontrolador RP2040; Motorista de caminhão; BitDogLaB.

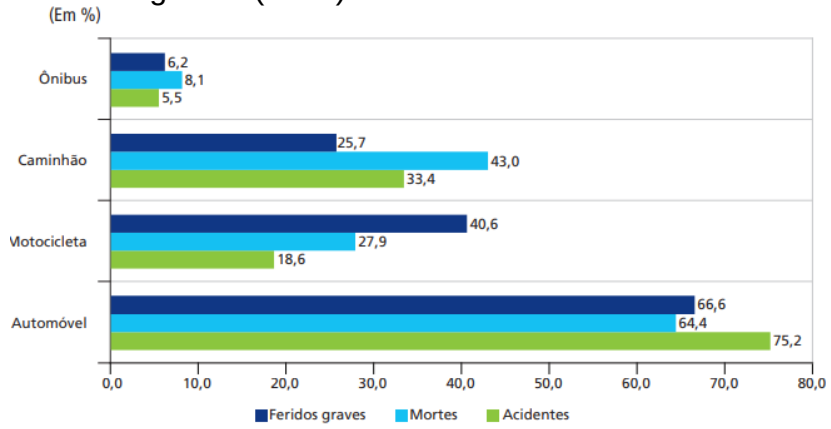
SUMÁRIO

1 INTRODUÇÃO

O transporte rodoviário desempenha um papel crucial no Brasil, sendo o principal responsável pelo deslocamento de cargas no país. Ele movimenta cerca de 60% de toda a carga transportada, utilizando uma vasta malha rodoviária de aproximadamente 1,72 milhões de quilômetros e uma frota composta por 7 milhões de veículos dedicados ao transporte de mercadorias (CNT, 2024).

De acordo com o Instituto de Pesquisa Econômica Aplicada (2015), em 2014, foram registrados 169.163 acidentes nas rodovias federais fiscalizadas pela PRF, resultando em 8.227 mortes e cerca de 100 mil pessoas feridas. Os acidentes envolvendo caminhões apresentam uma taxa de vítimas fatais proporcionalmente maior em relação à frequência desses acidentes, embora o número de lesões graves seja relativamente menor (Figura 1.1).

Figura 1.1 Envolvimento das modalidades de transporte nos acidentes com mortes e vítimas com lesões graves (2014)



Fonte: IPEA (2015).

De acordo com dados do Observatório de Segurança e Saúde no Trabalho, elaborado pelo Ministério Público do Trabalho (MPT) em parceria com a Organização Internacional do Trabalho (OIT) citado por Silva (2022), o transporte rodoviário de cargas ocupou, em 2021, a terceira posição no ranking de Comunicações de Acidentes de Trabalho (CATs) no Brasil. No entanto, o setor liderou em número de mortes, com 327 óbitos registrados em acidentes de trabalho e 12.771 notificações de acidentes ao longo do ano (Figura 1.2).

Figura 1.2 – Acidente com transporte rodoviário de cargas

Fonte: Silva (2022).

Pesquisas apontam fatores como fadiga, sonolência, cansaço físico e mental, uso de substâncias ilícitas e sistemas de pagamento por produção como principais causas de acidentes envolvendo motoristas profissionais. Apesar desses riscos, o setor de transporte rodoviário de cargas (TRC) frequentemente prioriza a produtividade, vinculando a remuneração dos motoristas ao número e à distância das viagens realizadas. Esse modelo de trabalho leva muitos motoristas a enfrentar longas jornadas com períodos insuficientes de descanso, configurando uma das principais causas de acidentes no setor (Fragoso Junior; Garcia, 2019).

Aproximadamente 23,7% dos caminhoneiros trabalham entre 26 e 31 dias por mês, com uma média diária de até 13 horas ao volante segundo a pesquisa “A realidade do caminhoneiro autônomo em 2022”, da Confederação Nacional dos Transportadores Autônomos (CNTA). Alysson Coimbra, da Associação Mineira de Medicina do Tráfego (Ammetra), destaca que motoristas de veículos pesados estão entre as profissões mais perigosas do Brasil. Ele alerta que a carga excessiva de trabalho compromete a saúde física e mental, levando ao abuso de álcool e drogas para prolongar as jornadas. Esse cenário gera impactos negativos não apenas para os motoristas, mas também para toda a sociedade, que arca com os custos da insegurança viária (Silva, 2022).

Assim, a fadiga é uma das principais causas de acidentes nas estradas, representando um perigo tanto para os motoristas quanto para os outros usuários da via. Para empresas que gerenciam grandes frotas, o impacto da fadiga vai além da segurança. Ela também afeta os custos operacionais devido a danos aos veículos, atrasos nas entregas e aumento dos prêmios de seguros.

Investir em sistemas de monitoramento da fadiga não é apenas uma questão de segurança, mas também de eficiência e responsabilidade corporativa. Com a adoção de tecnologias apropriadas e conscientização dos motoristas, é possível reduzir significativamente os riscos associados à fadiga no transporte rodoviário de cargas (Figura 1.3).

Figura 1.3 – Sistema ilustrativo de monitoramento da fadiga em transporte rodoviário de carga

Fonte: figura gerada por algoritmo de inteligência artificial.

1.1 Justificativa

O setor de transporte rodoviário de cargas (TRC) é um dos mais relevantes para a economia brasileira, mas também um dos mais perigosos no que se refere à segurança do trabalho. A fadiga e a sonolência dos motoristas estão entre os principais fatores que contribuem para esse cenário alarmante, impactando diretamente a segurança viária e os custos operacionais das empresas.

O modelo de remuneração baseado na produtividade, aliado a longas jornadas de trabalho e períodos insuficientes de descanso, leva muitos caminhoneiros a enfrentar desgaste físico e mental severo. Esse ritmo exaustivo não apenas compromete a saúde dos condutores, mas também aumenta o risco de acidentes, muitas vezes levando ao uso de substâncias estimulantes para manter-se acordado.

Diante dessa realidade, torna-se essencial o desenvolvimento de soluções tecnológicas que auxiliem no monitoramento da fadiga dos motoristas, prevenindo acidentes e promovendo melhores condições de trabalho. A implementação de um sistema embarcado para monitoramento da fadiga pode oferecer um suporte essencial na detecção precoce de sinais de cansaço, permitindo ações preventivas

para garantir a segurança nas estradas. Além da redução de acidentes, tal sistema pode contribuir para a otimização dos custos operacionais das empresas, minimizando danos aos veículos, atrasos e despesas com seguros.

Assim, o desenvolvimento de um sistema embarcado capaz de monitorar a fadiga dos motoristas de caminhão, utilizando sensores e tecnologias adequadas para fornecer alertas em tempo real justifica-se. Dessa forma, busca-se promover um transporte mais seguro e eficiente, beneficiando tanto os condutores quanto as empresas e a sociedade como um todo.

1.2 Objetivos

1.2.1 Objetivo geral

Propor um projeto de sistema embarcado para monitorar a fadiga de motoristas de caminhões.

1.2.2 Objetivos específicos

De maneira a atingir o objetivo geral deste projeto, propõem-se os seguintes objetivos específicos:

- Desenvolver um sistema embarcado baseado no microcontrolador RP2040;
- Implementar uma interface de anamnese;
- Integrar a captura e processamento de sinais vitais;
- Criar um algoritmo de avaliação de fadiga;
- Realizar testes de validação.

2 REVISÃO BIBLIOGRÁFICA

Cruz et al. (2016), no trabalho intitulado “Um sistema para monitoramento de sinais fisiológicos baseado em hardware de baixo custo com acesso via WEB”, apresentam um sistema para monitoramento de sinais fisiológicos baseado em hardware de baixo custo, com acesso remoto via web. O objetivo é permitir o acompanhamento de pacientes, especialmente cardíacos, sem a necessidade de deslocamentos constantes a hospitais. A arquitetura do sistema é composta por três camadas: coleta de dados, processamento e disponibilização online. A coleta é realizada por sensores acoplados a uma placa Arduino, que se comunica com uma placa Intel Galileo Gen2 para processar e armazenar os dados. Esses dados podem ser acessados remotamente via um serviço web, facilitando o acompanhamento por profissionais de saúde.

O estudo de caso desenvolvido validou a proposta utilizando um sensor de batimentos cardíacos, comunicação sem fio via módulos XBee e um aplicativo para visualização dos dados. Os testes mostraram que o sistema apresenta resultados confiáveis comparáveis a dispositivos comerciais, demonstrando seu potencial para aplicações na área médica. A pesquisa destaca a importância do uso de tecnologias acessíveis para o monitoramento remoto da saúde, reduzindo custos e melhorando a qualidade de vida dos pacientes. Como próximos passos, os autores sugerem a incorporação de novos sensores para monitoramento de outros parâmetros fisiológicos, ampliando a aplicabilidade do sistema (Cruz et al., 2016).

Ribas e Oliveira (2020) desenvolveram de um sistema de monitoramento remoto de sinais vitais utilizando o microcontrolador ESP32. O objetivo é oferecer suporte à Atenção Domiciliar à Saúde, permitindo que pacientes sejam monitorados à distância por meio da medição de temperatura corporal, frequência cardíaca e saturação de oxigênio no sangue. Para a aquisição desses dados, foram utilizados os sensores Max30100 e LM35, cujas leituras são transmitidas via protocolo MQTT para a plataforma ThingSpeak, possibilitando o acesso remoto por equipes médicas.

Os autores contextualizam a importância do monitoramento remoto, especialmente em cenários como a pandemia de COVID-19, onde a necessidade de evitar internações desnecessárias e reduzir a exposição ao vírus foi um fator

determinante. Além disso, destacam a relevância da Internet das Coisas (IoT) para a área da saúde, permitindo uma comunicação eficiente entre dispositivos e profissionais.

A implementação do protótipo envolveu tanto a construção do hardware, incluindo a integração dos sensores ao ESP32, quanto o desenvolvimento do software, responsável pela coleta, processamento e transmissão dos dados. Testes foram realizados comparando os resultados obtidos com dispositivos comerciais, evidenciando limitações dos sensores utilizados. O estudo conclui que, apesar das dificuldades encontradas, o projeto representa uma contribuição relevante para o avanço da IoT na área da saúde (Ribas; Oliveira, 2020).

Costa Neto e Costa (2022) abordaram o desenvolvimento de um sistema embarcado de baixo custo para aplicação na telemedicina, utilizando o microcontrolador ESP-32. A proposta surge da necessidade de soluções acessíveis para monitoramento de sinais vitais, especialmente em tempos de crise sanitária, como a pandemia de COVID-19, que evidenciou limitações no acesso à saúde. O sistema proposto visa aferir sinais vitais, como frequência cardíaca e oxigenação sanguínea, empregando sensores de baixo custo, como o MAX-30100, e comunicação via Bluetooth para transmissão dos dados.

A pesquisa destaca a importância da tecnologia na evolução da medicina, ressaltando o papel dos sistemas embarcados e da Internet das Coisas (IoT) na telemedicina. O estudo inclui uma revisão teórica sobre microcontroladores, protocolos de comunicação (I2C), sensores biomédicos e avanços tecnológicos aplicados à saúde (Costa Neto; Costa, 2022).

O desenvolvimento do protótipo envolveu a programação do ESP-32 na IDE Arduino, garantindo a integração eficiente entre hardware e software. Os testes demonstraram a funcionalidade do sistema, com transmissão remota de dados para dispositivos móveis. O trabalho conclui que soluções acessíveis podem ampliar o acesso à saúde, reduzindo custos e viabilizando o monitoramento remoto de pacientes, com potencial para futuras melhorias e ampliações no projeto (Costa Neto; Costa, 2022).

Silva et al. (2019) no artigo intitulado “Protótipo de plataforma embarcada para medição de sinais vitais utilizando IoT” apresentaram o desenvolvimento de um protótipo de plataforma embarcada para medição de sinais vitais utilizando a Internet

das Coisas (IoT). A plataforma e-Health foi utilizada para coletar dados de sinais vitais, como pressão arterial, eletrocardiograma, respiração e temperatura corporal, com o objetivo de criar uma base de dados para treinar redes neurais artificiais que possam auxiliar no diagnóstico de doenças e anomalias em tempo real. A metodologia envolveu estudos em engenharia biomédica, sistemas embarcados e IoT, além da aplicação prática na plataforma e-Health. Foram utilizados sensores como o AMS 5915 para pressão arterial, MAX30100 para oximetria e frequência cardíaca, AD8232 para ECG e MLX90614 para temperatura. Os resultados mostraram que o protótipo é viável, com medições precisas e discrepâncias mínimas em comparação com equipamentos comerciais. O estudo conclui que a integração de tecnologias IoT com plataformas de saúde pode reduzir custos e melhorar a eficácia no monitoramento de pacientes.

A dissertação "Um Sistema Embarcado de Detecção de Fadiga e Distração de Motoristas", de Ricardo Creonte Câmara de Meira Santos, aborda o desenvolvimento de um sistema baseado em visão computacional para identificar sinais de fadiga e distração em motoristas, visando aumentar a segurança viária.

O sistema analisa imagens faciais do motorista para detectar distrações por meio da orientação da face e sinais de fadiga através da duração das piscadas e bocejos. A medição da duração das piscadas é realizada utilizando a métrica PERCLOS (Percentage of Eyelid Closure). Ao identificar situações de risco, o sistema emite alertas para prevenir acidentes.

A pesquisa inclui uma comparação entre diferentes plataformas de desenvolvimento embarcado para a execução de algoritmos de detecção e classificação de face e olhos, auxiliando na escolha do hardware adequado para o sistema. A validação do sistema foi realizada em ambiente real com motoristas profissionais, evitando viés de laboratório e demonstrando a eficácia na detecção de situações de risco.

Além disso, a dissertação analisa como os alertas sonoros emitidos pelo sistema influenciam positivamente o comportamento dos motoristas. Os testes indicaram que, com os alertas sonoros ativados, os motoristas apresentaram uma redução nas situações de risco.

Em suma, o trabalho propõe uma solução tecnológica acessível para monitoramento e prevenção de acidentes relacionados à fadiga e distração dos

motoristas, com potencial para ser integrada a sistemas de segurança veicular existentes.

Em consonância com o trabalho de Santos (2020), no mercado, existem diversos sistemas de monitoramento de fadiga para motoristas de caminhão que usa câmeras e sensores para identificar sinais de sonolência e distração, a saber: Cobli Cam Pro[1], Maxtrack[2], ACR[3], entre outros.

[1]https://www.cobli.co/recursos/camera-de-fadiga/?campaignid=21251709577&adgroupid=162599881860&adid=698045155893&utm_source=google&utm_term=c%C3%A2mera%20de%20fadiga&utm_medium=cpc&utm_campaign=br-coblicam-pro&hsa_net=adwords&hsa_kw=c%C3%A2mera%20de%20fadiga&hsa_tgt=kwd-1638694072779&hsa_acc=6887042085&hsa_grp=162599881860&hsa_mt=b&hsa_cam=21251709577&hsa_ver=3&hsa_ad=698045155893&hsa_src=g&gad_source=1&gclid=Cj0KCQiA8fW9BhC8ARIsACwHqYqfg6T1C3vy5t4EzgP-VksJiuU3BbPQO5L_E7fZKSA_OkVQkSG4we4aArrVEALw_wcB
[2]https://www.maxtrack.com.br/monitoramento-de-fadiga-e-desatencao/
[3]https://acr1.com.br/produtos_post/monitoramento-fadiga/

3 METODOLOGIA

Esta seção discorre sobre a abordagem metodológica a ser utilizada para a realização deste trabalho, incluindo a classificação e roteiro do desenvolvimento da pesquisa.

3.1 Classificação da pesquisa

A pesquisa pode ser classificada quanto a sua natureza como aplicada, uma vez que tem como objetivo principal o desenvolvimento de um sistema embarcado para monitoramento de fadiga em motoristas de caminhão, com foco na solução de um problema prático relacionado à segurança no transporte rodoviário de cargas.

No que tange a abordagem do problema, é do tipo quali-quantitativa, uma vez que trata a subjetividade do sujeito e traduz em números opiniões e informações para classificá-las e analisá-las.

Em relação aos seus objetivos, pode-se classifica-la como explicativa, visto que visa identificar os fatores que determinam ou contribuem para a ocorrência da fadiga em motoristas de caminhão.

Além disso, a pesquisa é de natureza bibliográfica e experimental, pois foi elaborada a partir de material já publicado e envolve a construção e teste de um protótipo funcional, utilizando hardware e software específicos para validar as funcionalidades propostas.

3.2 Etapas da pesquisa

A primeira etapa do projeto consistiu em uma revisão bibliográfica para embasar o desenvolvimento do sistema. Foram analisados trabalhos relacionados ao monitoramento de sinais vitais, sistemas embarcados de baixo custo e tecnologias de Internet das Coisas (IoT). A pesquisa bibliográfica forneceu *insights* sobre a escolha de componentes, protocolos de comunicação e estratégias de implementação.

Com base nas pesquisas realizadas e na capacitação do Embarcatech, foram selecionados os seguintes componentes para o desenvolvimento do sistema:

- Microcontrolador RP2040 (BitDogLab): escolhido por sua versatilidade, baixo custo e suporte a múltiplas interfaces (GPIO, I2C, PIO, UART);

- Display SSD1306 (128x64): para exibição de perguntas, respostas e sinais vitais;
- Matriz de LEDs WS2812 (5x5): para feedback visual do nível de fadiga;
- LEDs RGB: para indicar respostas da anamnese (verde para "Sim" e vermelho para "Não") e alerta visual de acordo a avaliação da fadiga;
- Botões A e B: para captura de respostas do usuário;
- Comunicação Serial (UART): para entrada de dados dos sinais vitais via PC.

As funcionalidades do software foram definidas com base nos requisitos do projeto, a saber: anamnese, sinais vitais e avaliação da fadiga. Estas foram definidas da seguinte forma:

- Anamnese:
    - Exibição de perguntas no display SSD1306;
    - Captura de respostas via botões A e B;
    - Armazenamento das respostas para avaliação de fadiga.
- Sinais Vitais:
    - Entrada de dados via comunicação serial (UART);
    - Exibição dos valores no display SSD1306;
    - Armazenamento dos valores para avaliação de fadiga.
- Avaliação de Fadiga:
    - Cálculo da pontuação com base nas respostas da anamnese e sinais vitais;
    - Exibição do resultado no display, na matriz de LEDs (carinha feliz/triste) e LEDs RGB (verde/vermelho).

Para o desenvolvimento do software, foi utilizada a IDE Visual Studio Code (VS Code***)***, configurada para suportar o microcontrolador RP2040 e o Pico SDK. A configuração inicial incluiu os seguintes passos:

- Instalação do VS Code: download e instalação do Visual Studio Code, garantindo um ambiente de desenvolvimento moderno e flexível;
- Configuração do suporte ao RP2040: instalação do Pico SDK e do CMake, permitindo a compilação e programação do microcontrolador RP2040;

- Instalação das extensões necessárias: adição das extensões do C/C++, CMake Tools e Pico-W-Go para facilitar a escrita, compilação e upload do código;
- Configuração das bibliotecas essenciais: inclusão das bibliotecas necessárias no ambiente de desenvolvimento, como:
    - pico-sdk: Biblioteca principal para o RP2040;
    - ssd1306: Controle do display OLED via I2C;
    - font: fontes para A-Z, a-z, 0-9 e sinais de pontuação cujos caracteres tem 8 x 8 pixels;
    - ws2812: Controle da matriz de LEDs WS2812 via PIO.
- Definição dos pinos GPIO e interfaces: configuração no código-fonte para definir corretamente os pinos de entrada e saída, bem como os protocolos de comunicação I2C, PIO e UART;
- Configuração do CMake: criação dos arquivos de CMakeLists.txt para garantir a correta compilação do firmware no VS Code.

A programação foi realizada em linguagem C, seguindo uma estrutura modular para facilitar a manutenção e o entendimento. As principais etapas de programação foram:

- Inicialização dos Periféricos:
    - Configuração dos pinos GPIO para botões e LEDs;
    - Inicialização do display SSD1306 via I2C;
    - Configuração da matriz de LEDs WS2812 via PIO;
    - Configuração da comunicação serial (UART).
- Implementação das Funcionalidades:
    - Desenvolvimento das funções para exibição de perguntas e captura de respostas;
    - Implementação da lógica para cálculo da pontuação de fadiga;
    - Integração dos componentes para exibição dos resultados no display e na matriz de LEDs.
- Tratamento de Interrupções e Debouncing:
    - Uso de interrupções para captura rápida e eficiente dos acionamentos dos botões;

- Implementação de debouncing via software para evitar leituras incorretas.

A depuração foi realizada em duas etapas: testes unitários e integrados. Nos testes unitários foi verificado individualmente cada funcionalidade (exibição no display, acionamento dos botões, controle dos LEDs, etc.) e feita a correção de erros de lógica e configuração. Já os integrados, foi realizada a validação do funcionamento do sistema como um todo, testes de cenários reais, como respostas à anamnese e entrada de sinais vitais e por último, ajustes finais para garantir a confiabilidade e a precisão do sistema.

4 PROJETO

4.1 Descrição do projeto

Com o emprego da ferramenta educacional BitDogLab, foram utilizadas várias interfaces como entradas/saídas e comunicação, presente no microcontrolador RP2040, para projetar um sistema a partir do proposto abaixo escrito em linguagem C, juntamente com os recursos do kit de Desenvolvimento de Software Pico SDK.

4.1.1 Componentes necessários

Neste projeto, foram utilizados os seguintes componentes conectados à placa BitDogLab:

- LED RGB, com os pinos conectados às GPIOs (Verde – 11 e Vermelho - 13);
- Botão A conectado à GPIO 5;
- Botão B conectado à GPIO 6;
- Matriz 5x5 de LEDs (endereçáveis) WS2812, conectada à GPIO 7;
- Display SSD1306 conectado via I2C (GPIO 14 e GPIO15);
- Microcontrolador Raspberry Pi Pico W.

4.1.2 Requisitos do projeto

Para o desenvolvimento, foram seguidos os seguintes requisitos:

1. Uso de interrupções: todas as funcionalidades relacionadas aos botões foram implementadas utilizando rotinas de interrupção (IRQ);
2. Debouncing: foi implementado o tratamento do bouncing dos botões via software;
3. Utilização do Display 128 x 64: a utilização de ferramentas gráficas demonstrou o entendimento do princípio de funcionamento do display, bem como, a utilização do protocolo I2C;
4. Organização do código: o código foi estruturado e comentado para facilitar o entendimento.

4.1.3 Funcionalidades do projeto

4.1.3.1 Anamnese – exibição no display SSD1306

As perguntas de anamnese são exibidas no display SSD1306 de 128 x 64 pixels após a mensagem escrita “ANAMNESE”. As perguntas são:

a) Você sente pressão para cumprir prazos apertados de entrega?
b) Você sente que o seu tempo de descanso é insuficiente antes da nova jornada?
c) Você sente dificuldades para dormir?
d) Você sente sonolência excessiva durante o dia?
e) Você costuma usar cafeína ou energéticos para se manter acordado?
f) Você sente dores musculares ou articulares relacionadas ao trabalho?
g) Você tem problemas de visão ou dor de cabeça frequente?
h) Você já sentiu sintomas como irritabilidade, estresse ou ansiedade no trabalho?
i) Você tem hipertensão ou diabetes diagnosticada?
j) Faz uso de medicação contínua?

As respostas às perguntas da anamnese são dadas pelo acionamento do Botão A ou Botão B. As respostas por meio dos acionamentos dos botões A e B são armazenadas para que possam ser utilizados depois na avaliação de fadiga. As respostas são exibidas no display SSD1306. Após exibição da resposta da anamnese sim ou não no display SSD1306, aguarda 02 segundos e exibe a próxima pergunta.

4.1.3.2 Interação com o Botão A

Ao pressionar o Botão A é exibido no Display SSD1306 a resposta Sim e o LED RGB verde é ligado. A resposta no display e acionamento do LED são executados ao mesmo tempo.

A resposta no display, assim como o LED ligado dura por 2 segundos.

O Botão A só é acionado para responder as perguntas da anamnese. Em qualquer outro acionamento, ele fica apagado.

4.1.3.3 Interação com o Botão B

Ao pressionar o Botão B é exibido no Display SSD1306 a resposta Não e o LED RGB vermelho é ligado. A resposta no display e acionamento do LED são executados ao mesmo tempo.

A resposta no display, assim como o LED ligado, dura por 2 segundos.

O Botão B só é acionado para responder as perguntas da anamnese. Em qualquer outro acionamento, ele fica apagado.

4.1.3.4 Sinais vitais – entrada via PC e exibição no PC e no display SSD1306

As perguntas de sinais vitais são exibidas no serial monitor. Antes, é exibido no display SSD1306 a mensagem “SINAIS VITAIS”.

As respostas são digitadas pelo usuário no serial monitor. Os valores são armazenados para que possam ser utilizados depois na avaliação de fadiga.

Após digitado o valor no serial monitor, é exibido no display SSD1306 um resumo do sinal vital.

Após aparecer a resposta do sinal vital no display SSD1306, aguarda 02 segundos e exibe a próxima pergunta no serial monitor.

4.1.3.5 Avaliação da fadiga

A avaliação é feita baseada nos critérios abaixo:

I. ANAMNESE

Para cada resposta "Sim", soma 1 ponto.

Pontuação da Anamnese: 0 a 10 pontos (peso 1)

II. SINAIS VITAIS

Cada parâmetro fora da faixa normal adiciona 1 ponto:

- Frequência cardíaca (FC) fora da faixa de 60-100 bpm.
- Pressão arterial fora da faixa de 100-140 mmHg (sistólica) e 60-90 mmHg (diastólica).
- Frequência respiratória (FR) fora da faixa de 12-20 mrpm.

Pontuação dos Sinais Vitais: 0 a 3 pontos (peso 2)

III. ESCALA DE FADIGA

Pontuação Total = Anamnese (0-10) + [Sinais Vitais (0-3) x 2] (Máximo: 16 pontos)

- 0 a 5 pontos – Baixa fadiga: Estado adequado para conduzir.
- 6 a 9 pontos – Moderada fadiga: Atenção necessária para evitar riscos. Recomenda-se descanso adicional.
- 10 a 13 pontos – Alta fadiga: Risco significativo de segurança. Descanso imediato e revisão da rotina.

- 14 a 18 pontos – Fadiga severa: Alto risco de acidentes. Requer repouso urgente e avaliação profissional.

Se a pontuação for de 0 a 5 pontos, aparece na matriz de LED 5 x 5 uma carinha feliz e uma mensagem no display: Adequado para conduzir. Caso contrário, aparece na matriz de LED 5 x 5 uma carinha triste e uma mensagem no display de acordo a faixa de pontuação: Atenção! Recomenda-se descanso; Atenção! Descanso imediato; Atenção! Repouso urgente.

4.2 Especificação do Hardware

4.2.1 Diagrama em Bloco

O sistema embarcado é composto pelos seguintes blocos interligados:

- Microcontrolador RP2040 (BitDogLab): controla todo o sistema e gerencia a comunicação entre os dispositivos;
- Botões (A e B): entrada de usuário para respostas da anamnese;
- LEDs RGB: indicadores visuais das respostas da anamnese;
- Display SSD1306 (128x64): exibe as perguntas da anamnese e os sinais vitais;
- Matriz de LEDs WS2812 (5x5): exibe um ícone indicando a fadiga do motorista;
- Comunicação Serial (UART): entrada de dados dos sinais vitais pelo PC e exibição de resultados.

Assim, o diagrama em blocos representativo é apresentado na Figura 4.1.

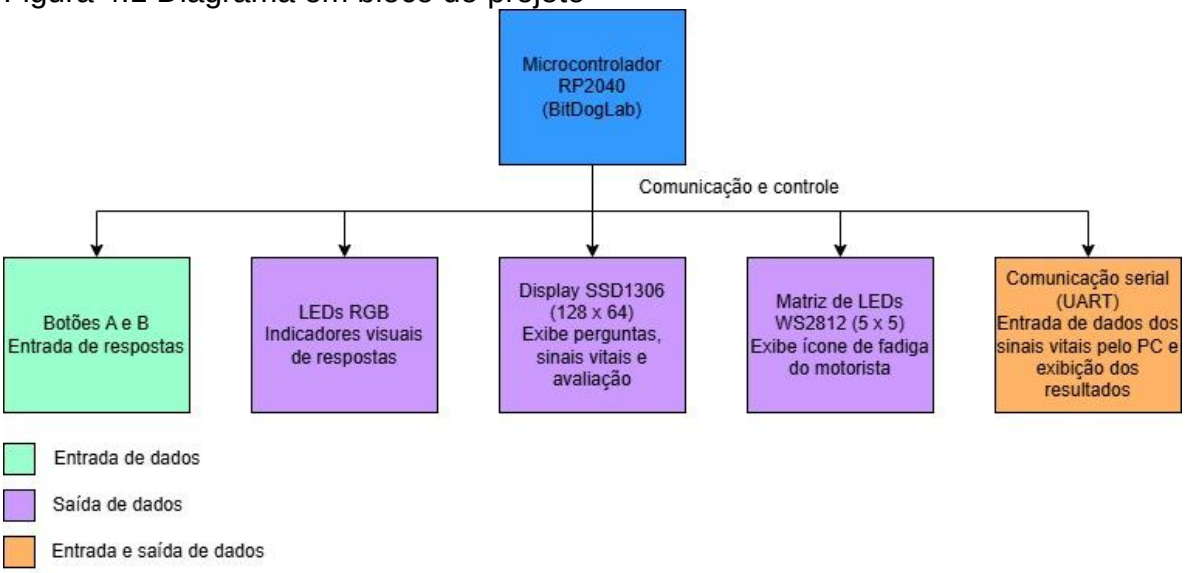
4.2.2 Função de Cada Bloco

As funções de cada bloco são:

- Microcontrolador RP2040: gerencia todo o sistema, processa as entradas dos botões, controla os LEDs RGB, a matriz de LEDs WS2812, o display SSD1306 e a comunicação serial;
- Botões (A e B): capturam as respostas do usuário durante a anamnese;
- LEDs RGB: indicam visualmente as respostas do usuário (verde para "Sim" e vermelho para "Não");
- Display SSD1306: exibe as perguntas da anamnese, as respostas e os sinais vitais;

- Matriz de LEDs WS2812: exibe um ícone (carinha feliz ou triste) indicando o nível de fadiga do motorista;
- Comunicação Serial (UART): recebe os sinais vitais digitados pelo usuário no PC e exibe os resultados no display.

Figura 4.1 Diagrama em bloco do projeto



Fonte: autoria própria.

4.2.3 Configuração de Cada Bloco

As configurações de cada componente são:

- Microcontrolador RP2040:
  - Configuração dos pinos GPIO para botões, LEDs RGB, matriz de LEDs e display;
  - Uso de interrupções para tratar os eventos dos botões;
  - Configuração do I2C para comunicação com o display SSD1306;
  - Configuração do PIO para controlar a matriz de LEDs WS2812.
- Botões (A e B):
  - Configurados como entradas com pull-up interno;
  - Debouncing implementado via software.
- LEDs RGB:
  - Configurados como saídas digitais.
- Display SSD1306:
  - Configurado via I2C com endereço 0x3C;
  - Utiliza biblioteca SSD1306 para exibição de texto e gráficos.

- Matriz de LEDs WS2812:
  - Configurada via PIO para controle dos LEDs endereçáveis.
- Comunicação Serial (UART):
  - Configurada para receber dados do PC e exibir resultados no display.

4.2.4 Comandos e Registros Utilizados

Os principais comandos utilizados foram:

- GPIO**:**
  - `gpio_init(GPIO_NUM)`: inicializa os pinos;
  - `gpio_set_dir(GPIO_NUM, GPIO_IN/OUT)`: define a direção do pino;
  - `gpio_pull_up(GPIO_NUM)`: ativa pull-up interno para botões;
  - `gpio_put(GPIO_NUM, HIGH/LOW)`: controla estados dos LEDs;
  - `gpio_set_irq_enabled_with_callback()`: habilita interrupções nos pinos dos botões.

- I2C (Display SSD1306)**:**
  - `i2c_init(i2c1, 400 * 1000)`: inicializa I2C a 400 kHz;
  - `ssd1306_init(&ssd, WIDTH, HEIGHT, false, ENDERECO, I2C_PORT)`: configura o display;
  - `gpio_set_function()`: configura os pinos SDA e SCL para I2C.

- PIO (Matriz WS2812)**:**
  - `pio_sm_put_blocking(pio0, 0, pixel_grb << 8u)`: envia dados para os LEDs endereçáveis;
  - `pio_add_program()`: adiciona o programa PIO para controlar a matriz de LEDs WS2812;
  - `ws2812_program_init()`: inicializa o controlador WS2812.

- Serial (UART)**:**
  - `scanf("%d", &valor)`: captura valores de sinais vitais;
  - `printf("Texto")`: exibe informações no Serial Monitor;
  - `stdio_init_all()`: inicializa a comunicação serial.

4.2.5 Descrição da Pinagem Usada

Os pinos utilizados são apresentados no Quadro 4.1.

Quadro 4.1 Pinagem utilizada

Pino RP2040	**Componente**	**Função**
GPIO 5	Botão A	Entrada do usuário (Sim)
GPIO 6	Botão B	Entrada do usuário (Não)
GPIO 11	LED RGB (Verde)	Indicador visual (Sim)
GPIO 13	LED RGB (Vermelho)	Indicador visual (Não)
GPIO 7	Matriz WS2812	Controle dos LEDs endereçáveis
GPIO 14	I2C SDA	Comunicação com o display
GPIO 15	I2C SCL	Comunicação com o display

Fonte: dados da pesquisa.

4.2.6 Circuito completo do hardware

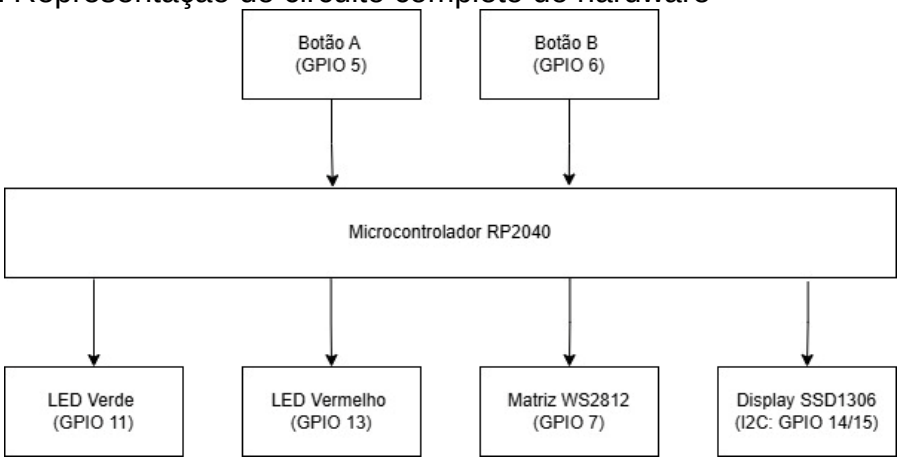
O detalhamento do circuito é apresentado abaixo e a representação esquemática na Figura 4.2.

- Botões (A e B):
  - Botão A: Conectado ao GPIO 5 do RP2040.
    - Um terminal do botão está conectado ao GPIO 5.
    - O outro terminal está conectado ao GND (terra).
    - Resistor de pull-up interno ativado no GPIO 5.
  - Botão B: Conectado ao GPIO 6 do RP2040.
    - Um terminal do botão está conectado ao GPIO 6.
    - O outro terminal está conectado ao GND.
    - Resistor de pull-up interno ativado no GPIO 6.
- LEDs RGB:
  - LED Verde: Conectado ao GPIO 11 do RP2040.
    - O ânodo do LED está conectado ao GPIO 11.
    - O cátodo do LED está conectado ao GND através de um resistor limitador de corrente (ex.: 220 Ω).
  - LED Vermelho: Conectado ao GPIO 13 do RP2040.
    - O ânodo do LED está conectado ao GPIO 13.
    - O cátodo do LED está conectado ao GND através de um resistor limitador de corrente (ex.: 220 Ω).
- Matriz de LEDs WS2812 (5x5):

- Conectada ao GPIO 7 do RP2040.
  - O pino de entrada de dados (DIN) da matriz está conectado ao GPIO 7.
  - O pino de alimentação (VCC) da matriz está conectado ao 3.3V do RP2040.
  - O pino de terra (GND) da matriz está conectado ao GND do RP2040.

- Display SSD1306 (128x64):
  - Conectado via I2C:
    - SDA: Conectado ao GPIO 14 do RP2040.
    - SCL: Conectado ao GPIO 15 do RP2040.
    - O pino de alimentação (VCC) do display está conectado ao 3.3V do RP2040.
    - O pino de terra (GND) do display está conectado ao GND do RP2040.
- Comunicação Serial (UART):
  - A comunicação serial é feita via USB, utilizando os pinos padrão do RP2040 para UART (TX e RX).
  - Não são necessárias conexões externas, pois o RP2040 já possui suporte nativo para comunicação serial via USB.

Figura 4.2 Representação do circuito completo do hardware



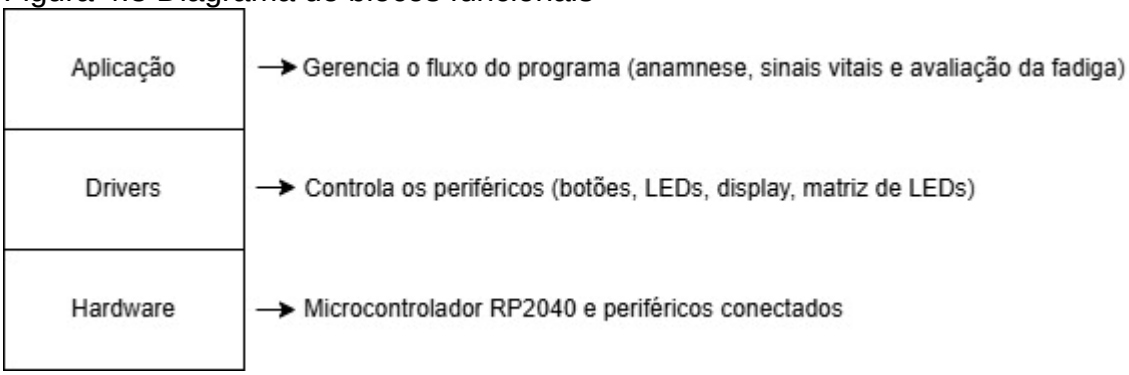
Fonte: autoria própria.

4.3 Especificação do firmware

4.3.1 Blocos funcionais

O software pode ser dividido em camadas funcionais conforme o diagrama abaixo (Figura 4.3):

Figura 4.3 Diagrama de blocos funcionais



Fonte: autoria própria.

A descrição das funcionalidades dos blocos segue:

- Aplicação:
  - Gerencia o fluxo principal do programa;
  - Realiza a anamnese, captura sinais vitais e avalia a fadiga;
  - Exibe resultados no display e na matriz de LEDs.
- Drivers:
  - Botões: implementa interrupções e debouncing para capturar respostas do usuário;
  - LEDs RGB: controla os LEDs para indicar respostas (verde para "Sim", vermelho para "Não");
  - Display SSD1306: gerencia a exibição de perguntas, respostas e sinais vitais;
  - Matriz de LEDs WS2812: controla a exibição de ícones (carinha feliz/triste) com base na avaliação de fadiga;
  - Comunicação Serial: recebe dados dos sinais vitais via UART.
- Hardware:
  - Microcontrolador RP2040 e periféricos conectados (botões, LEDs, display, matriz de LEDs).

4.3.2 Descrição das funcionalidades

As funcionalidades previstas no firmware são:

- Anamnese:

    - Exibe perguntas no display SSD1306;
    - Captura respostas via botões A e B;
    - Armazena respostas para avaliação de fadiga.
- Sinais Vitais:
    - Recebe dados via comunicação serial (UART);
    - Exibe valores no display SSD1306;
    - Armazena valores para avaliação de fadiga.
- Avaliação de Fadiga:
    - Calcula a pontuação com base nas respostas da anamnese e sinais vitais;
    - Exibe o resultado no display, na matriz de LEDs (carinha feliz/triste) e no LED RGB (verde/vermelho).

4.3.3 Definição das variáveis

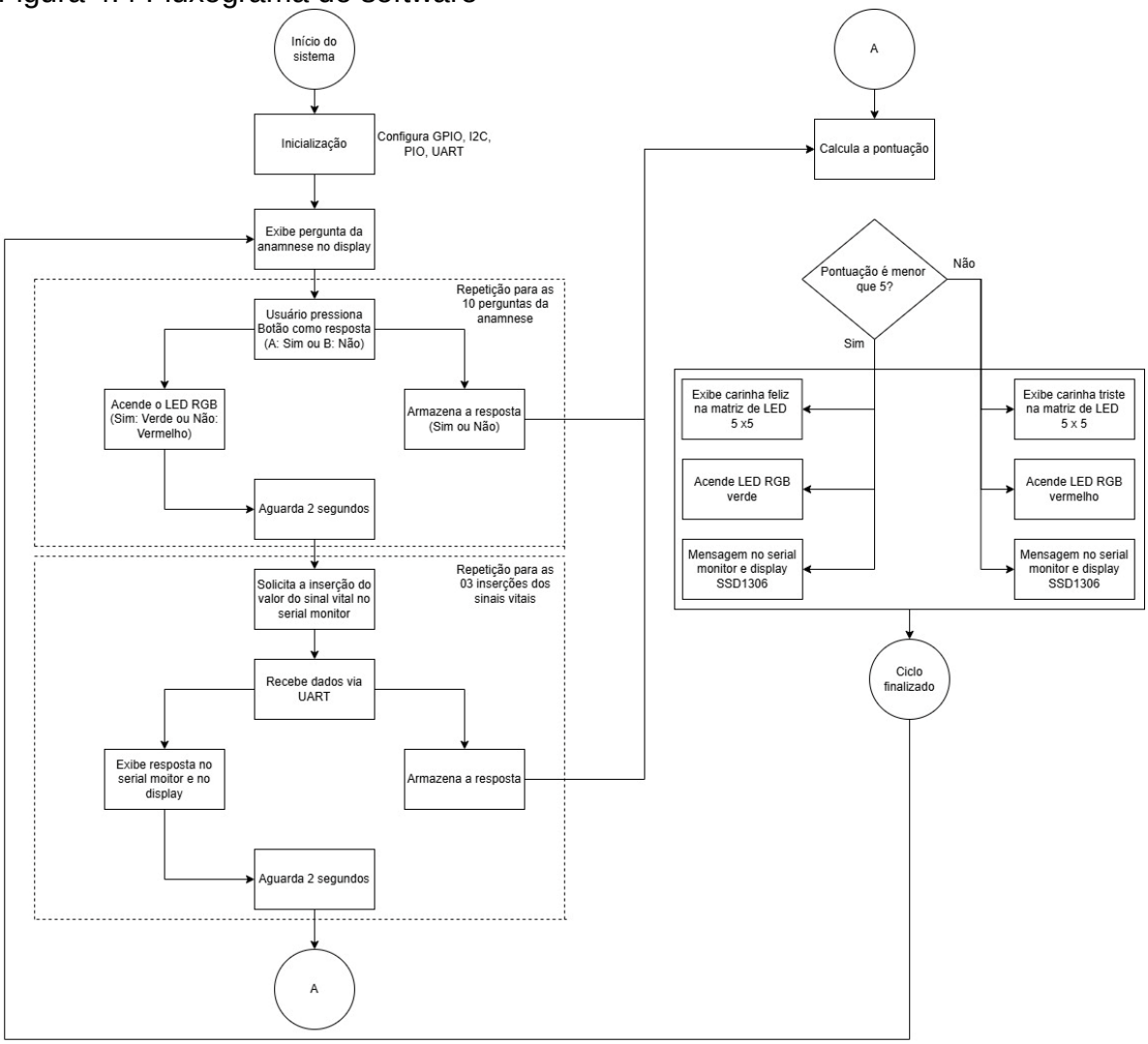
As variáveis são:

- Variáveis Globais:
    - `volatile int resposta_atual`: armazena a resposta atual do usuário (1 para "Sim", 0 para "Não");
    - `ssd1306_t ssd`: estrutura para controle do display SSD1306;
    - `const char *perguntas[][5]`: armazena as perguntas da anamnese;
    - `int respostas[10]`: armazena as respostas da anamnese;
    - `int pontuacao_anamnese, pontuacao_sinais_vitais, pontuacao_total`: armazenam as pontuações para avaliação de fadiga.
- Variáveis Locais:
    - `char input[16]`: buffer para entrada de dados via serial;
    - `int frequencia_cardiaca, pressao_arterial_sistolica, pressao_arterial_diastolica, frequencia_respiratoria`: armazenam os sinais vitais.

4.3.4 Fluxograma

O fluxograma representativo do software é apresentado na Figura 4.4.

Figura 4.4 Fluxograma do software



Fonte: autoria própria.

4.3.5 Inicialização

A inicialização do software pode ser dividida conforme segue:

- Configuração do Hardware:
  - Inicializa GPIO para botões e LEDs;
  - Configura I2C para o display SSD1306;
  - Configura PIO para a matriz de LEDs WS2812;
  - Inicializa UART para comunicação serial.
- Inicialização dos Periféricos:
  - Configura interrupções para os botões;
  - Inicializa o display SSD1306;
  - Configura a matriz de LEDs WS2812.

- Loop Principal:
    - Executa a anamnese, captura sinais vitais e avalia a fadiga;
    - Repete o processo indefinidamente.

4.3.6 Configurações dos registros

As funções são:

- GPIO:
    - `gpio_init()`: inicializa os pinos GPIO;
    - `gpio_set_dir()`: define a direção do pino (entrada/saída);
    - `gpio_pull_up()`: ativa o resistor de pull-up interno;
    - `gpio_set_irq_enabled_with_callback()`: habilita interrupções nos pinos dos botões.
- I2C:
    - `i2c_init()`: inicializa o barramento I2C.
    - `gpio_set_function()`: configura os pinos SDA e SCL para I2C.
- PIO:
    - `pio_add_program()`: adiciona o programa PIO para controlar a matriz de LEDs WS2812.
    - `ws2812_program_init()`: inicializa o controlador WS2812.
- UART:
    - `stdio_init_all()`: inicializa a comunicação serial.

4.3.7 Estrutura e formato dos dados

Os dados usados no software seguem abaixo:

- Perguntas da Anamnese:
    - Armazenadas em um array de strings (`const char *perguntas[][5]`);
    - Cada pergunta pode ocupar até 5 linhas no display.
- Respostas da Anamnese:
    - Armazenadas em um array de inteiros (`int respostas[10]`);
    - 1 para "Sim", 0 para "Não".
- Sinais Vitais:
    - Recebidos via UART e armazenados em variáveis inteiras;
    - Exibidos no display SSD1306.

- Avaliação de Fadiga:
    - Pontuação calculada com base nas respostas da anamnese e sinais vitais;
    - Resultado exibido no display e na matriz de LEDs.

4.3.8 Protocolo de comunicação

O protocolo de comunicação usado segue:

- UART:
    - Utilizado para receber dados dos sinais vitais;
    - Configuração padrão: 9600 baud rate, 8 bits de dados, sem paridade, 1 bit de stop.

4.3.9 Formato do pacote de dados

O formato é como segue:

- Entrada de Sinais Vitais:
    - Dados são enviados via UART no formato de texto;
    - Exemplo: 60 (frequência cardíaca), 120/80 (pressão arterial), 16 (frequência respiratória).

4.3.10 Código em linguagem C

```
#include <stdio.h>
#include <stdlib.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include "hardware/gpio.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "hardware/timer.h"
#include "inc/ssd1306.h"
#include "inc/font.h"
#include "ws2812.pio.h"

//Definição das constantes
#define I2C_PORT i2c1
#define I2C_SDA 14
#define I2C_SCL 15
#define ENDERECO 0x3C

#define BOTAO_A 5
```

```c
#define BOTAO_B 6
#define LED_VERDE 11
#define LED_VERMELHO 13
#define DEBOUNCE_DELAY 200 // Tempo de debounce em ms
#define WS2812_PIN 7
#define NUM_PIXELS 25
#define IS_RGBW false

// Estrutura do display
ssd1306_t ssd;

volatile int resposta_atual = -1;

// Declaração da função de callback
void botao_callback(uint gpio, uint32_t eventos);

// Função para enviar um pixel para a matriz de LEDs
static inline void put_pixel(uint32_t pixel_grb) {
    pio_sm_put_blocking(pio0, 0, pixel_grb << 8u);
}

// Função para converter RGB para formato de 32 bits
static inline uint32_t urgb_u32(uint8_t r, uint8_t g, uint8_t b) {
    return ((uint32_t)(g) << 16) | ((uint32_t)(r) << 8) | (uint32_t)(b);
}

// Função para desligar todos os LEDs da matriz
void clear_led_matrix() {
    for (int i = 0; i < NUM_PIXELS; i++) {
        put_pixel(0); // Define todos os LEDs como apagados (R=0, G=0, B=0)
    }
}

// Inicializa o display SSD1306
void inicializar_display() {
    i2c_init(I2C_PORT, 400 * 1000);
    gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
    gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
    gpio_pull_up(I2C_SDA);
    gpio_pull_up(I2C_SCL);

    ssd1306_init(&ssd, WIDTH, HEIGHT, false, ENDERECO, I2C_PORT);
    ssd1306_config(&ssd);
    ssd1306_send_data(&ssd);
    ssd1306_fill(&ssd, false);
    ssd1306_send_data(&ssd);
}
```

```
// Inicializa botões e LEDs
void inicializar_gpio() {
    gpio_init(BOTAO_A);
    gpio_set_dir(BOTAO_A, GPIO_IN);
    gpio_pull_up(BOTAO_A);
    gpio_set_irq_enabled_with_callback(BOTAO_A, GPIO_IRQ_EDGE_FALL, true,
&botao_callback);

    gpio_init(BOTAO_B);
    gpio_set_dir(BOTAO_B, GPIO_IN);
    gpio_pull_up(BOTAO_B);
    gpio_set_irq_enabled_with_callback(BOTAO_B, GPIO_IRQ_EDGE_FALL, true,
&botao_callback);

    gpio_init(LED_VERDE);
    gpio_set_dir(LED_VERDE, GPIO_OUT);

    gpio_init(LED_VERMELHO);
    gpio_set_dir(LED_VERMELHO, GPIO_OUT);


}

// Inicializa o PIO e o controlador WS2812
void inicializar_ws2812() {
    PIO pio = pio0;
    int sm = 0;
    uint offset = pio_add_program(pio, &ws2812_program);
    ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);
}

// Callback para interrupção dos botões
void botao_callback(uint gpio, uint32_t eventos) {
    static uint32_t ultimo_tempo = 0;
    uint32_t tempo_atual = to_ms_since_boot(get_absolute_time());

    if (tempo_atual - ultimo_tempo < DEBOUNCE_DELAY) return;
    ultimo_tempo = tempo_atual;

    if (gpio == BOTAO_A) {
        resposta_atual = 1;
        gpio_put(LED_VERDE, 1);
        gpio_put(LED_VERMELHO, 0);
    } else if (gpio == BOTAO_B) {
        resposta_atual = 0;
        gpio_put(LED_VERDE, 0);
        gpio_put(LED_VERMELHO, 1);
    }
```

```
}

// Função para exibir perguntas no display
void exibir_pergunta(const char *linha1, const char *linha2, const char
*linha3, const char *linha4, const char *linha5) {
    ssd1306_fill(&ssd, false);
    ssd1306_rect(&ssd, 3, 3, 122, 58, true, false);

    if (linha1) ssd1306_draw_string(&ssd, linha1, 8, 10);
    if (linha2) ssd1306_draw_string(&ssd, linha2, 8, 20);
    if (linha3) ssd1306_draw_string(&ssd, linha3, 8, 30);
    if (linha4) ssd1306_draw_string(&ssd, linha4, 8, 40);
    if (linha5) ssd1306_draw_string(&ssd, linha5, 8, 50);

    ssd1306_send_data(&ssd);
}

// Exibir resposta no display
void exibir_resposta(const char *resposta) {
    ssd1306_fill(&ssd, false);
    ssd1306_rect(&ssd, 3, 3, 122, 58, true, false);
    ssd1306_draw_string(&ssd, "Resposta:", 8, 20);
    ssd1306_draw_string(&ssd, resposta, 8, 40);
    ssd1306_send_data(&ssd);
    sleep_ms(2000);
    gpio_put(LED_VERDE, 0);
    gpio_put(LED_VERMELHO, 0);
}

// Perguntas da anamnese e captura de resposta
int perguntas_anamnese() {
    const char *perguntas[][5] = {
        {"Voce sente", "pressao para", "cumprir prazos", "apertados de",
"entrega?"},
        {"Seu tempo de", "descanso eh", "insuficiente", "antes da nova",
"jornada?"},
        {"Voce sente", "dificuldades", "para dormir?", NULL, NULL},
        {"Voce sente", "sonolencia", "excessiva", "durante", "o dia?"},
        {"Costuma usar", "cafeina ou", "energeticos", "para se manter",
"acordado?"},
        {"Sente dores", "musculares ou", "articulares", "relacionadas", "ao
trabalho?"},
        {"Voce tem", "problemas de", "visao ou", "dor de cabeca",
"frequente?"},
        {"Ja sentiu", "irritabilidade", "ou estresse ou", "ansiedade", "no
trabalho?"},
        {"Voce tem", "hipertensao", "ou diabetes", "diagnosticada?", NULL},
        {"Faz uso de", "medicacao", "continua?", NULL, NULL}
```

```
    };
    int respostas[10];
    int pontuacao_anamnese = 0;

    for (int i = 0; i < 10; i++) {
        resposta_atual = -1;
        exibir_pergunta(perguntas[i][0], perguntas[i][1], perguntas[i][2],
perguntas[i][3], perguntas[i][4]);
        while (resposta_atual == -1) tight_loop_contents();
        respostas[i] = resposta_atual;
        exibir_resposta(resposta_atual ? "Sim" : "Nao");

        // Calcula a pontuação da anamnese
        if (i == 1) {
            if (respostas[i] == 0) pontuacao_anamnese++; // Se a resposta for
"Não", soma 1 ponto
        } else {
            if (respostas[i] == 1) pontuacao_anamnese++; // Se a resposta for
"Sim", soma 1 ponto
        }
    }

    return pontuacao_anamnese;
}

// Função para capturar sinais vitais via serial e exibir no display
int sinais_vitais() {
    char input[16];
    int frequencia_cardiaca, pressao_arterial_sistolica,
pressao_arterial_diastolica, frequencia_respiratoria;
    int pontuacao_sinais_vitais = 0;

    // Solicitar a frequência cardíaca e exibir no serial monitor
    printf("Digite a frequencia cardiaca (bpm): ");
    scanf("%d", &frequencia_cardiaca);
    printf("Frequencia cardiaca: %d bpm\n", frequencia_cardiaca);
    sleep_ms(2000);

    // Verifica se a frequência cardíaca está fora da faixa normal
    if (frequencia_cardiaca < 60 || frequencia_cardiaca > 100)
pontuacao_sinais_vitais++;

    // Exibir no display
    ssd1306_fill(&ssd, false);
    ssd1306_draw_string(&ssd, "Frequencia", 8, 10);
    ssd1306_draw_string(&ssd, "cardiaca:", 8, 20);
    char str[16];  // Vetor maior para armazenar o valor convertido para
string
```

```
    snprintf(str, sizeof(str), "%d bpm", frequencia_cardiaca);  // Converter o
inteiro para string
    ssd1306_draw_string(&ssd, str, 8, 30);  // Exibe o valor no display
    ssd1306_send_data(&ssd);
    sleep_ms(2000);

    // Solicitar e exibir a pressão arterial
    printf("Digite a pressao arterial sistolica (mmHg): ");
    scanf("%d", &pressao_arterial_sistolica);
    printf("Digite a pressao arterial diastolica (mmHg): ");
    scanf("%d", &pressao_arterial_diastolica);
    printf("Pressao arterial: %d x %d mmHg\n", pressao_arterial_sistolica,
pressao_arterial_diastolica);
    sleep_ms(2000);

    // Verifica se a pressão arterial está fora da faixa normal
    if (pressao_arterial_sistolica < 100 || pressao_arterial_sistolica > 140
||
        pressao_arterial_diastolica < 60 || pressao_arterial_diastolica > 90)
pontuacao_sinais_vitais++;

    // Exibir no display
    ssd1306_fill(&ssd, false);
    ssd1306_draw_string(&ssd, "Pressao", 8, 10);
    ssd1306_draw_string(&ssd, "arterial:", 8, 20);
    snprintf(str, sizeof(str), "%d x %d mmHg", pressao_arterial_sistolica,
pressao_arterial_diastolica);  // Converter pressão arterial para string
    ssd1306_draw_string(&ssd, str, 8, 30);  // Exibe o valor no display
    ssd1306_send_data(&ssd);
    sleep_ms(2000);

    // Solicitar e exibir a frequência respiratória
    printf("Digite a frequencia respiratoria (mrpm): ");
    scanf("%d", &frequencia_respiratoria);
    printf("Frequencia respiratoria: %d mrpm\n", frequencia_respiratoria);
    sleep_ms(2000);

    // Verifica se a frequência respiratória está fora da faixa normal
    if (frequencia_respiratoria < 12 || frequencia_respiratoria > 20)
pontuacao_sinais_vitais++;

    // Exibir no display
    ssd1306_fill(&ssd, false);
    ssd1306_draw_string(&ssd, "Frequencia", 8, 10);
    ssd1306_draw_string(&ssd, "respiratoria:", 8, 20);
    snprintf(str, sizeof(str), "%d mrpm", frequencia_respiratoria);  //
Converter frequência respiratória para string
    ssd1306_draw_string(&ssd, str, 8, 30);  // Exibe o valor no display
```

```
    ssd1306_send_data(&ssd);
    sleep_ms(2000);

    return pontuacao_sinais_vitais;
}

//Função da carinha feliz
void exibir_carinha_feliz() {
    // Padrão da carinha feliz na matriz de LED 5x5
    const bool carinha_feliz[NUM_PIXELS] = {
        0, 1, 1, 1, 0,  // Linha 1
        1, 0, 0, 0, 1,  // Linha 2
        0, 0, 0, 0, 0,  // Linha 3
        0, 1, 0, 1, 0,  // Linha 4
        0, 1, 0, 1, 0   // Linha 5
    };

    // Exibe o padrão na matriz de LED
    for (int i = 0; i < NUM_PIXELS; i++) {
        put_pixel(carinha_feliz[i] ? urgb_u32(0, 100, 0) : 0); // Verde para a
carinha feliz
    }

    printf("Carinha feliz exibida na matriz de LED.\n");
}

// Função da carinha triste
void exibir_carinha_triste() {
    // Padrão da carinha triste na matriz de LED 5x5
    const bool carinha_triste[NUM_PIXELS] = {
        1, 0, 0, 0, 1,  // Linha 1
        0, 1, 1, 1, 0,  // Linha 2
        0, 0, 0, 0, 0,  // Linha 3
        0, 1, 0, 1, 0,  // Linha 4
        0, 1, 0, 1, 0   // Linha 5
    };

    // Exibe o padrão na matriz de LED
    for (int i = 0; i < NUM_PIXELS; i++) {
        put_pixel(carinha_triste[i] ? urgb_u32(100, 0, 0) : 0); // Vermelho
para a carinha triste
    }

    printf("Carinha triste exibida na matriz de LED.\n");
}

// Função para exibir o resultado da avaliação de fadiga
void exibir_resultado_fadiga(int pontuacao_total) {
```

```
    ssd1306_fill(&ssd, false);
    ssd1306_rect(&ssd, 3, 3, 122, 58, true, false);

    if (pontuacao_total >= 0 && pontuacao_total <= 5) {
        ssd1306_draw_string(&ssd, "Adequado", 8, 20);
        ssd1306_draw_string(&ssd, "para conduzir", 8, 40);
        gpio_put(LED_VERDE, 1);
        gpio_put(LED_VERMELHO, 0);
        exibir_carinha_feliz(); // Exibe carinha feliz na matriz de LED
    } else if (pontuacao_total >= 6 && pontuacao_total <= 9) {
        ssd1306_draw_string(&ssd, "Atencao!", 8, 20);
        ssd1306_draw_string(&ssd, "Recomenda-se", 8, 40);
        ssd1306_draw_string(&ssd, "descanso", 8, 50);
        gpio_put(LED_VERDE, 0);
        gpio_put(LED_VERMELHO, 1);
        exibir_carinha_triste(); // Exibe carinha triste na matriz de LED
    } else if (pontuacao_total >= 10 && pontuacao_total <= 13) {
        ssd1306_draw_string(&ssd, "Alerta!", 8, 20);
        ssd1306_draw_string(&ssd, "Descanso", 8, 40);
        ssd1306_draw_string(&ssd, "imediato", 8, 50);
        gpio_put(LED_VERDE, 0);
        gpio_put(LED_VERMELHO, 1);
        exibir_carinha_triste(); // Exibe carinha triste na matriz de LED
    } else if (pontuacao_total >= 14 && pontuacao_total <= 16) {
        ssd1306_draw_string(&ssd, "Perigo!", 8, 20);
        ssd1306_draw_string(&ssd, "Repouso", 8, 40);
        ssd1306_draw_string(&ssd, "urgente", 8, 50);
        gpio_put(LED_VERDE, 0);
        gpio_put(LED_VERMELHO, 1);
        exibir_carinha_triste(); // Exibe carinha triste na matriz de LED
    }

    ssd1306_send_data(&ssd);
    sleep_ms(2000);
    gpio_put(LED_VERDE, 0);
    gpio_put(LED_VERMELHO, 0);
    clear_led_matrix(); // Desliga a matriz de LED
}

int main() {
    stdio_init_all(); // Inicializa a comunicação serial
    inicializar_display();
    inicializar_gpio();
    inicializar_ws2812(); // Inicializa o PIO e o controlador WS2812

    while (true) {
        // Anamnese no display
        ssd1306_fill(&ssd, false);
```

```
        ssd1306_draw_string(&ssd, "ANAMNESE", 30, 30);
        ssd1306_send_data(&ssd);
        sleep_ms(2000);

        int pontuacao_anamnese = perguntas_anamnese();
        sleep_ms(2000); // Pausa antes de reiniciar o ciclo

        // Sinais vitais no display
        ssd1306_fill(&ssd, false);
        ssd1306_draw_string(&ssd, "SINAIS VITAIS", 10, 30);
        ssd1306_send_data(&ssd);
        sleep_ms(2000);

        int pontuacao_sinais_vitais = sinais_vitais();

        // Calcula a pontuação total
        int pontuacao_total = pontuacao_anamnese + (pontuacao_sinais_vitais *
2);

        // Exibe o resultado da avaliação de fadiga
        exibir_resultado_fadiga(pontuacao_total);
        sleep_ms(3000);
    }
}
```

4.4 Testes de validação

Os testes de validação realizados para garantir o funcionamento correto do projeto, com base na descrição e no código fornecido foram:

4.4.1 Testes de Inicialização

- Objetivo**:**
  - Verificar se todos os componentes são inicializados corretamente.
- Procedimento**:**
  - Ligar o sistema e observar se o display SSD1306 exibe a mensagem inicial;
  - Verificar se os LEDs RGB e a matriz de LEDs WS2812 estão apagados;
  - Confirmar se os botões estão configurados corretamente (sem acionamentos acidentais).
- Resultado Esperado**:**
  - Display exibe "ANAMNESE" após a inicialização;
  - LEDs RGB e matriz de LEDs permanecem apagados;

- Botões não acionam respostas sem interação do usuário.

4.4.2 Testes dos Botões

- Objetivo**:**
  - Validar o funcionamento dos botões A e B, incluindo debouncing e interrupções.
- Procedimento**:**
  - Acionar o Botão A e verificar se o LED verde acende e se a resposta "Sim" é exibida no display;
  - Acionar o Botão B e verificar se o LED vermelho acende e se a resposta "Não" é exibida no display;
  - Testar o debouncing pressionando os botões rapidamente várias vezes.
- Resultado Esperado**:**
  - Botão A acende o LED verde e exibe "Sim" no display;
  - Botão B acende o LED vermelho e exibe "Não" no display;
  - Debouncing funciona corretamente, evitando múltiplas leituras de um único acionamento.

4.4.3 Testes da Anamnese

- Objetivo**:**
  - Validar a exibição das perguntas e o armazenamento das respostas.
- Procedimento**:**
  - Iniciar a anamnese e verificar se as perguntas são exibidas corretamente no display;
  - Responder às perguntas alternando entre os botões A e B;
  - Verificar se as respostas são armazenadas e exibidas corretamente.
- Resultado Esperado**:**
  - Todas as 10 perguntas são exibidas sequencialmente;
  - As respostas são armazenadas e exibidas corretamente no display;
  - A pontuação da anamnese é calculada corretamente.

4.4.4 Testes dos Sinais Vitais

- Objetivo**:**

    - Validar a entrada e exibição dos sinais vitais.
- Procedimento**:**
    - Iniciar a fase de sinais vitais e verificar se a mensagem "SINAIS VITAIS" é exibida no display;
    - Inserir valores para frequência cardíaca, pressão arterial e frequência respiratória via serial;
    - Verificar se os valores são exibidos corretamente no display.
- Resultado Esperado**:**
    - Os valores inseridos são exibidos corretamente no display;
    - A pontuação dos sinais vitais é calculada corretamente com base nos valores inseridos.

4.4.5 Testes da Avaliação de Fadiga

- Objetivo**:**
    - Validar o cálculo da pontuação e a exibição do resultado.
- Procedimento**:**
    - Responder à anamnese e inserir sinais vitais para gerar diferentes pontuações;
    - Verificar se a pontuação total é calculada corretamente;
    - Verificar se o resultado é exibido corretamente no display, na matriz de LEDs e LED RGB.
- Resultado Esperado**:**
    - A pontuação total é calculada corretamente;
    - O resultado é exibido no display e na matriz de LEDs conforme a pontuação:
        - 0 a 5 pontos: Carinha feliz e mensagem "Adequado para conduzir";
        - 6 a 9 pontos: Carinha triste e mensagem "Atenção! Recomenda-se descanso";
        - 10 a 13 pontos: Carinha triste e mensagem "Alerta! Descanso imediato";
        - 14 a 16 pontos: Carinha triste e mensagem "Perigo! Repouso urgente".

4.4.6 Testes da Matriz de LEDs WS2812

- Objetivo**:**
    - Validar a exibição dos ícones na matriz de LEDs.
- Procedimento**:**
    - Gerar diferentes pontuações para testar a exibição da carinha feliz e triste;
    - Verificar se os LEDs acendem conforme o padrão esperado.
- Resultado Esperado**:**
    - Carinha feliz é exibida corretamente para pontuações de 0 a 5.
    - Carinha triste é exibida corretamente para pontuações acima de 5.

4.4.7 Testes de Comunicação Serial

- Objetivo**:**
    - Validar a entrada de dados via UART.
- Procedimento**:**
    - Enviar valores para frequência cardíaca, pressão arterial e frequência respiratória via serial;
    - Verificar se os valores são recebidos e processados corretamente.
- Resultado Esperado**:**
    - Os valores são recebidos e exibidos corretamente no display.

4.5 Discussão

O projeto foi desenvolvido com base em requisitos claros e bem definidos, utilizando o microcontrolador RP2040 e componentes como botões, LEDs RGB, matriz de LEDs WS2812, display SSD1306 e comunicação serial.

No que tange ao funcionamento geral, o sistema foi capaz de executar todas as funcionalidades propostas, a saber:

- Anamnese**:** as perguntas foram exibidas corretamente no display SSD1306 e as respostas dos botões A e B foram capturadas e armazenadas com sucesso;
- Sinais Vitais: a entrada de dados via comunicação serial funcionou conforme esperado, os valores foram exibidos no display e armazenados com sucesso;

- Avaliação de Fadiga**:** a pontuação foi calculada corretamente e o resultado foi exibido no display e na matriz de LEDs.

Em relação a confiabilidade dos componentes pode-se inferir que:

- O uso de interrupções e debouncing garantiu uma resposta rápida e precisa aos acionamentos dos botões;
- Não foram observados falsos acionamentos ou múltiplas leituras de um único pressionamento dos botões;
- Os LEDs RGB responderam corretamente às respostas da anamnese, acendendo em verde para "Sim" e vermelho para "Não";
- A matriz de LEDs exibiu os ícones de carinha feliz e triste conforme a pontuação de fadiga, sem falhas ou inconsistências;
- O display exibiu todas as perguntas, respostas e sinais vitais de forma clara e legível;
- A comunicação I2C funcionou sem erros, garantindo a integridade dos dados exibidos;
- A entrada de dados via UART foi estável e os valores foram processados e exibidos corretamente no display.

Pode-se evidenciar como pontos fortes do projeto: a organização do código, o uso de interrupções, o debouncing e o retorno visual. O código foi estruturado de forma modular, com funções bem definidas para cada funcionalidade, o que facilitou a manutenção e o entendimento. A implementação de interrupções para os botões garantiu uma resposta rápida e eficiente, sem bloquear o fluxo principal do programa. O tratamento de bouncing via software foi eficaz, evitando leituras incorretas dos botões. A combinação de LEDs RGB, matriz de LEDs e display SSD1306 proporcionou um feedback visual claro e intuitivo para o usuário.

Contudo, alguns pontos de melhorias foram constatados:

- Robustez da Comunicação Serial**:**
    - A entrada de dados via UART pode ser aprimorada com a implementação de verificações de erro e tratamento de entradas inválidas.
- Interface do Usuário:

    - A inclusão de um menu interativo no display pode melhorar a experiência do usuário, permitindo a navegação entre as funcionalidades.
- Expansão das Funcionalidades:
    - O sistema pode ser expandido para incluir mais perguntas na anamnese ou mais parâmetros de sinais vitais, como oximetria para medição de saturação de oxigênio no sangue, aumentando sua utilidade;
    - O sistema pode ser expandido para incluir monitoramento via câmeras conforme trabalho de Santos (2020), para análise de padrões de piscar e fechamento dos olhos, sinais comuns de fadiga, além de anamnese e parâmetros de sinais vitais;
    - O sistema pode ser expandido para incluir sensor de posição da cabeça para detectar movimentos bruscos ou inclinação excessiva da cabeça;
    - O sistema pode ser expandido para incluir sensor de ambiente (para medir temperatura e umidade dentro da cabine, que podem influenciar o conforto do motorista;
    - O processamento dos dados é realizado localmente pelo microcontrolador, contudo, podem ser transmitidos via Wi-Fi para um servidor na nuvem. Os dados no servidor podem ser integrados a um painel de monitoramento acessível aos gestores da frota, permitindo a visualização em tempo real e o histórico de alertas;
    - Em casos extremos, o sistema pode interagir com o veículo limitando a velocidade ou sugerindo parada por meio de integração com o sistema eletrônico do veículo.

5 CONCLUSÃO

O projeto demonstrou confiabilidade em todas as suas funcionalidades principais. Os testes de validação confirmaram que: o sistema responde corretamente às interações do usuário, os dados são processados e exibidos de forma precisa e a avaliação de fadiga é calculada e exibida conforme os critérios estabelecidos.

A organização do código, o uso de interrupções e a implementação de debouncing contribuíram para a robustez do sistema. Embora existam oportunidades de melhoria, o projeto atende aos requisitos propostos e pode ser considerado confiável para o propósito educacional e de avaliação de fadiga. Os objetivos propostos foram alcançados.

6 LINK DO REPOSITÓRIO DO GIT HUB

https://github.com/priscilasuzart/Projeto_final.git

7 LINK DO VÍDEO

https://drive.google.com/file/d/1Fh2N2aZ9t5mJutaN696Oe42eiu6yJ8jK/view?usp=sharing

Referências

CNT. Confederação Nacional dos Transportes. **Boletins Técnicos CNT - dezembro/2024**. Disponível em: file:///C:/Users/ppscarvalho/Downloads/Boletim%20Unificado%20-%20Dezembro%202024.pdf. Acesso em: jan. 2025.

COSTA NETO, J.B.; COSTA, P.H.S. Sistema embarcado de baixo custo utilizando o ESP-32 na telemedicina. Trabalho de conclusão de curso em Engenharia Elétrica – Anima Educação. 2022. Disponível em: https://repositorio-api.animaeducacao.com.br/server/api/core/bitstreams/2ae210c2-3aa5-4b7c-848d-57858c5e7e75/content. Acesso em: fev. 2025.

CRUZ, L.F.; BERNARDON, C.; SANTOS, E.V.M.; SIMÕES, W.C.S.S; SILVA, V.J. Um sistema para monitoramento de sinais fisiológicos baseado em hardware de baixo custo com acesso via WEB. In: XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. **Anais...** Salvador: SBRC, 2016. Disponível em: https://sbrc2016.ufba.br/downloads/WoCCES/154096.pdf. Acesso em: fev. 2025.

FRAGOSO JUNIOR, A.; GARCIA, E. G.. Transporte rodoviário de carga: acidentes de trabalho fatais e fiscalização trabalhista. **Revista Brasileira de Saúde Ocupacional**, v. 44, p. e3, 2019.

IPEA. Instituto de Pesquisa Econômica Aplicada. Acidentes de trânsito nas rodovias federais brasileiras: caracterização, tendências e custos para a sociedade. **Relatório de Pesquisa**. Brasília, 2015.

RIBAS, A.R.; OLIVEIRA, A.A.M. **Sistema para monitoramento remoto de sinais vitais utilizando ESP32**. Trabalho de Conclusão de Curso Ciência Da Computação - Universidade Franciscana (UFN). Santa Maria: UFN, 2020. Disponível em: https://www.tfgonline.lapinf.ufn.edu.br/media/midias/AndersonRibeiroRibas.pdf . Acesso em: fev. 2025.

SANTOS, Ricardo Creonte Câmara de Meira. Um sistema embarcado de detecção de fadiga e distração de motoristas. 2020. 97 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto, 2020.

SILVA, E.S.; SILVA, J.P.; SILVA, R.F. SOUZA, M.H.S. Protótipo de plataforma embarcada para medição de sinais vitais utilizando IoT. VI Congresso Nacional de Educação. **Anais...** Campina Grande: CONEDU, 2019. Disponível em: https://www.editorarealize.com.br/editora/anais/conedu/2019/TRABALHO_EV127_MD1_SA20_ID13796_26092019192528.pdf. Acesso em: fev. 2025.

SILVA, M.A. Transporte de cargas lidera ranking de mortes no trabalho. **Jornal Opção**, jul. 2022 Disponível em: https://www.jornalopcao.com.br/transito/transporte-de-cargas-lidera-ranking-de-mortes-no-trabalho-418434/. Acesso em: jan. 2025.