



Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Bacharelado em
Engenharia de Computação

**Sistema de Telemetria de Sensores
LEGO MINDSTORMS EV3**

Antonio Carlos Albuquerque
Judenilson Araujo Silva

Campina Grande

2023

Antonio Carlos Albuquerque
Judenilson Araujo Silva

Sistema de Telemetria de Sensores LEGO MINDSTORMS EV3

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso Superior de Bacharelado em Engenharia de Computação do IFPB - Campus Campina Grande, como requisito parcial para a conclusão do curso de Bacharelado em Engenharia de Computação.

Orientador: MSc. Henrique do Nascimento Cunha

Campina Grande

2023

Antonio Carlos Albuquerque
Judenilson Araujo Silva

Sistema de Telemetria de Sensores LEGO MINDSTORMS EV3

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso Superior de Bacharelado em Engenharia de Computação do IFPB - Campus Campina Grande, como requisito parcial para a conclusão do curso de Bacharelado em Engenharia de Computação.

MSc. Henrique do Nascimento Cunha
Orientador

Avaliador 1
Membro da Banca

Avaliador 2
Membro da Banca

Campina Grande
2023

A345s Albuquerque, Antonio Carlos.

Sistema de telemetria de sensores: Lego Mindstorms EV3 / Antonio Carlos Albuquerque, Judenilson Araujo Silva. Campina Grande, 2023.

131 f. : il.

Trabalho de Conclusão de Curso (Curso Superior de Bacharelado em Engenharia de Computação) - Instituto Federal da Paraíba, 2023.

Orientador: Prof. Me. Henrique do Nascimento Cunha.

1. Telemetria de Sensores 2. ESP32 3. Robô I. Silva, Judenilson Araújo. II. Cunha, Henrique do Nascimento III. Título.

CDU 004

Eu Judenilson, dedico esta obra, fruto do meu trabalho humano, a Deus que por Ele e para Ele todas as coisas são feitas, bem como, a minha família que me atura todo dia, minha querida esposa Neriane e nossas duas filhas, Carolina e Rebeca pelas quais me esforço para tornar o dia mais produtivo.

Eu Antônio, dedico este trabalho a comunidade acadêmica, que possam se interessar em métodos de depuração de sensores e robôs, bem como os entusiastas e competidores da área de robótica, que pretendem aprender mais, e por isso buscam novos conhecimentos e ferramentas como o sistema de depuração dos sensores Lego EV3 deste trabalho.

Agradecimentos

Antonio

Agradeço a Deus, pelo dom da vida e a oportunidade de trabalhar nesse projeto e muitos outros.

Agradeço a minha família, que sempre me apoiou nas minhas decisões, bem como nos desafios dos caminhos que percorri. Aos meus pais que sempre me mostraram a importância da educação e da busca por conhecimento, e me educaram com os valores de honestidade, humildade, coragem entre muitos outros.

Aos meus irmãos, irmãs, primos e primas com que compartilhei meus desafios da universidade, e também desafios profissionais e pessoais.

Aos professores do curso, com os quais pude aprender sobre a área, e que foram referência e inspiração para mim, em especial o professor MSc. Henrique do Nascimento Cunha, que desde a primeira disciplina ministrada ampliou minha visão do curso, apresentando uma série de projetos e desafios incluindo este trabalho do qual ele foi nosso orientador.

Agradeço aos meus colegas de curso, com os quais pude compartilhar experiências e conhecimentos ao longo do curso. Em especial ao meu amigo Judenilson Araujo que tive a sorte de conhecer já no início do curso, e com quem compartilhei de vários projetos e desafios incluindo este trabalho.

Por último, mas também especiais os amigos que conheci durante o curso, Isaque Melo, Erick Pimentel, Micael Marques, Rayane Costa, e Gledson, com os quais participei de momentos de estudo e desafios, bem como momentos descontraídos e divertidos.

Judenilson

Agradeço a Deus, por ter escrito, antes da fundação do mundo, este projeto em meus caminhos.

À minha esposa Neriane que nos momentos difíceis me impulsionava, motivando a continuar e nunca desistir de concluir mais essa etapa na vida.

Às minhas filhas Carolina e Rebeca, que entendiam a minha ausência para estudar e paravam para escutar minhas teorias sobre matemática e desenvolvimento, ajudando a manter a memória fresca ao relembrar de problemas fundamentais.

À minhas irmãs Juliana e Jushlana e aos meus pais Genilson Gomes da Silva e Judith Araujo Silva, ambos *in memoriam*, por suas presenças e amor incondicional na minha vida sempre. Esta monografia é a prova de que todos os esforços deles pela minha educação não foram em vão.

Aos amigos do trabalho que, com toda boa vontade concordaram em trabalhar alguns horários sozinhos para que eu assistisse as aulas presenciais.

Obrigado ao grande amigo Mateus Maciel que emprestou sua impressora 3D com a maior boa vontade de sempre.

Externo também meus agradecimentos ao meu companheiro de jornada Antonio Albuquerque, que durante todo o período do curso foi parceiro e colega em vários dos projetos apresentados nas disciplinas e por fim aceitou colaborar também neste último trabalho desenvolvido no curso.

Ao meu orientador MSc. Henrique do Nascimento Cunha, que apesar da intensa rotina de sua vida acadêmica aceitou me orientar nesta monografia e acreditou no meu potencial em executar o projeto que foi de sua autoria.

Agradeço ao IFPB e aos seus docentes por todo conhecimento compartilhado em todos esses anos de curso, em especial ao Professor Dr. Jerônimo Silva Rocha por todo apoio e incentivo.

Agradeço aos meus colegas de IFPB, que nos momentos mais difíceis sempre foram muito solícitos.

Por último, mas não menos importante agradeço ao meu amigo Pedro Chagas que motivava todo dia que nos encontrávamos e me impulsionava a concluir mais este projeto, dizendo que eu deveria alçar voos maiores.

”Senhor, tu me sondas e me conheces. Sabes quando me sento e quando me levanto; de longe percebes os meus pensamentos. Sabes muito bem quando trabalho e quando descanso; todos os meus caminhos te são bem conhecidos. Antes mesmo que a palavra me chegue à língua, tu já a conheces inteiramente, Senhor. Tu me cercas, por trás e pela frente, e pões a tua mão sobre mim. Tal conhecimento é maravilhoso demais e está além do meu alcance, é tão elevado que não o posso atingir.”

(Salmos 139:1-6)

Resumo

Alunos de cursos de desenvolvimento de hardwares e softwares tendem a participar de torneios, concursos e apresentações na área da robótica. Diante de tamanha quantidade de pessoas envolvidas na área, bem como, após presenciarmos alguns desafios e dificuldades em nosso laboratório de robótica do IFPB-CG, enfrentando desafios relacionados a sensores, decidimos investigar este tópico e apresentar neste trabalho um hardware capaz de monitorar os dados dos sensores LEGO MINDSTORMS EV3 em tempo real, operando na máxima taxa de transferência de dados permitida pelo sensor, de forma dedicada e exclusiva, proporcionando maior segurança nas informações e conhecimento sobre erros de sensores, isso libera os recursos computacionais do robô LEGO, permitindo que ele se concentre inteiramente em suas rotinas operacionais. Todos os dados coletados pelo hardware desenvolvido na plataforma do ESP32 serão encaminhados para um aplicativo Android, também desenvolvido por nós, este aplicativo armazenará os logs e, potencialmente, também poderá gravar um vídeo para documentar o comportamento do robô durante sua operação. No decorrer desse trabalho serão demonstrados todos os passos e métodos utilizados para construção do hardware e do software, entre todo o processo estão a confecção de placa de circuito impresso, criação de case de armazenamento e acoplagem aos módulos LEGO com o protótipo completamente desenvolvido no Autodesk Fusion 360, o desenvolvimento do *firmware* em C++, que utiliza Websocket para comunicação e o aplicativo para celular Android construído em Java Script com auxílio do React Native.

Palavras-chaves: ESP32, Lego. Robô. Sensor. Cpp. Android. Websocket. Java Script. React Native. Engenharia Reversa.

Abstract

Students on hardware and software development courses tend to participate in tournaments, competitions and presentations in the field of robotics. Given such a large number of people involved in the area, as well as, after witnessing some challenges and difficulties in our IFPB-CG robotics laboratory, facing challenges related to sensors, we decided to investigate this topic and present in this work hardware capable of monitoring data of LEGO MINDSTORMS EV3 sensors in real time, operating at the maximum data transfer rate allowed by the sensor, in a dedicated and exclusive way, providing greater information security and knowledge about sensor errors, this frees up the LEGO robot's computational resources, allowing he focuses entirely on his operational routines. All data collected by the hardware developed on the ESP32 platform will be forwarded to an Android application, also developed by us, this application will store the logs and, potentially, can also record a video to document the behavior of the robot during its operation. During this work, all the steps and methods used to build the hardware and software will be demonstrated, including the production of a printed circuit board, creation of a storage case and coupling to LEGO modules with the prototype completely developed in Autodesk Fusion 360, the development of *firmware* in C++, which uses Websocket for communication and the Android cell phone application built in Java Script with the help of React Native.

Key-words: ESP32, Lego. Robot. Sensor. Cpp. Android. Websocket. JavaScript. React Native. Reverse Engineering.

Lista de ilustrações

Figura 1 – Visão geral do kit LEGO MINDSTORMS EV3	19
Figura 2 – Exemplo robô LEGO MINDSTORMS EV3	20
Figura 3 – Portas do kit LEGO MINDSTORMS EV3	20
Figura 4 – Disposição de Pinos do ESP32-WROOM	23
Figura 5 – Conector macho e fêmea da LEGO 11145	36
Figura 6 – Divisor de tensão	36
Figura 7 – Conexão para upload de firmware	37
Figura 8 – Primeira PCI, análise inicial de dados	38
Figura 9 – Lado esquerdo tonner já transferido, lado direito papel quase limpo	40
Figura 10 – Problema de curto na transferência de tonner	41
Figura 11 – Disposição dos componentes, topo e fundo respectivamente	42
Figura 12 – Esquemático completo do projeto	43
Figura 13 – Imagem do circuito gerado, visão fundo(vias) e topo(componentes)	43
Figura 14 – Imagem 3D da PCI do projeto	44
Figura 15 – Case, tampa inferior	44
Figura 16 – Case, tampa superior	45
Figura 17 – Comunicação Websocket	52
Figura 18 – Renderização do Gráfico	55
Figura 19 – Modelo Relacional	55
Figura 20 – Gravação de vídeo	58
Figura 21 – Depuração da execução	61

Lista de abreviaturas e siglas

ABS	<i>Anti-lock Braking System</i>
API	<i>Application Programming Interface</i>
APP	Aplicativo
CAD	<i>Computer Aided Design</i>
CAE	<i>Computer Aided Engineering</i>
CAM	<i>Computer Aided Manufacturing</i>
CNC	Controle Numérico Computadorizado
DFU	<i>Device Firmware Upgrade</i>
ER	Engenharia Reversa
FR	<i>Fire Retardant</i>
GIT	Sistema de Controle de Versão
GPIO	<i>General Purpose Input/Output</i>
GUI	<i>Graphic User Interface</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hipertext Transfer Protocol</i>
IEEE	Instituto de Engenheiros Elétricos e Eletrônicos
JSON	<i>JavaScript Object Notation</i>
LIB	Biblioteca
PCB	<i>Printed Circuit Board</i>
PCI	Placa de Circuito Impresso
SEK	<i>Standard Educational Kit</i>
SMD	<i>Surface Mounted Device</i>
SQL	<i>Structured Query Language</i>

TCP	<i>Transmission Control Protocol</i>
UART	<i>Universal Asynchronous Receiver/Transmitter</i>
UDP	<i>User Datagram Protocol</i>
WEP	<i>Wired Equivalent Privacy</i>
WPA	<i>Wi-Fi Protected Access</i>

Lista de Códigos

Código 4.1	Configuração PlatformIO	45
Código 4.2	Cabeçalho Firmware	46
Código 4.3	Análise de estado	47
Código 4.4	Obtendo Dados	48
Código 4.5	Inicializando Websocket server	49
Código 4.6	Callback Eventos Websocket server	50
Código 4.7	Enviando os Dados	50
Código 4.8	Pacote de Logs	52
Código 4.9	Buffer dos Logs	53
Código 4.10	Componente do Gráfico	54
Código 4.11	Configuração de execução	56
Código 4.12	Buffer dos logs para o Banco	56
Código 4.13	Armazenando vídeo	57
Código 4.14	Asset do vídeo	57
Código 4.15	Objeto json da execução	58
Código 4.16	Componente de vídeo	59
Código 4.17	Buffer de logs do vídeo	59

Sumário

1	INTRODUÇÃO	17
1.1	Considerações Preliminares	17
1.2	Objetivos	18
1.2.1	Objetivo Geral	18
1.2.2	Objetivos Específicos	18
2	FUNDAMENTAÇÃO TEÓRICA	19
2.1	LEGO MINDSTORMS EV3	19
2.2	Microcontroladores	21
2.2.1	ESP32	22
2.3	Prototipagem	23
2.4	Engenharia Reversa	24
2.5	GitHub	25
3	METODOLOGIA	27
3.1	Métodos	27
3.2	Ferramentas Utilizadas	27
3.2.1	Placa de Circuito Impresso	28
3.2.2	Osciloscópio	28
3.2.3	ESP32-WROVER-B	29
3.2.4	Arduino IDE	29
3.2.5	Visual Studio Code	30
3.2.6	PlatformIO	31
3.2.7	Websockets	31
3.2.8	React Native	32
3.2.9	Expo	32
3.2.10	Libs do React Native	32
3.2.11	Git e GitHub	33
4	RESULTADOS	34
4.1	Hardware	34
4.1.1	Análise de Dados	34
4.1.2	Circuito Eletrônico	35
4.1.3	Prototipagem	38
4.1.4	Desenho Técnico	41
4.1.5	<i>Firmware</i>	44

4.2	Aplicativo móvel	51
4.2.1	Comunicação Websocket	51
4.2.2	Logs e Buffers	52
4.2.3	Gráficos dos sensores	53
4.2.4	Banco de Dados	55
4.2.5	Gravação de vídeo	57
4.2.6	Depuração da execução	58
5	CONSIDERAÇÕES FINAIS	62
5.1	Sugestões para Trabalhos Futuros	62
	REFERÊNCIAS	63
	APÊNDICES	65
	APÊNDICE A – REPOSITÓRIO FIRMWARE	66
A.1	.gitignore do firmware	66
A.2	Arquivo configuração PlatformIO	66
A.3	Arquivo de configuração da rede	66
A.4	Arquivo código do firmware	67
	APÊNDICE B – REPOSITÓRIO APLICATIVO MÓVEL	71
B.1	Arquivo de configuração do Gradle	71
B.2	Arquivo de configuração do App	71
B.3	Arquivo de configuração do Metro	72
B.4	Arquivo index do App	73
B.5	Arquivo de configuração do Babel	73
B.6	Arquivo de estilo do App	73
B.7	Arquivo de configuração Android do App	73
B.8	.gitignore do App	74
B.9	Objeto de definição das dimensões dos canvases	75
B.10	Arquivo da classe dos gráficos	76
B.11	Arquivo da classe de listagem dos gráficos	77
B.12	Arquivo da classe de controle do Objeto Path	77
B.13	Arquivo de constantes do App	79
B.14	Arquivo de componente de navegação das telas	79
B.15	Arquivo inicialização do banco de dados	79
B.16	Arquivo das funções de interface de acesso as tabelas do banco de dados	80

B.17	Arquivo das operações da tabela de execuções	84
B.18	Componente da tela de listagem das execuções	87
B.19	Declaração da store compartilhada entre os componentes	88
B.20	Componente de player do vídeo da execução	89
B.21	Operações de banco na tabela de logs	95
B.22	Operações de banco na tabela de portas do sensor	99
B.23	Componente de visualização da execução	101
B.24	Propriedades do componente de vídeo player	103
B.25	Componente da tela de gravação	105
B.26	Componente de definição do sensor	109
B.27	Componente de estilo css	110
B.28	Componente de exibição dos sniffers cadastrados	111
B.29	Componente de compartilhamento de estados globais da aplicação	112
B.30	Componente de estilo css	117
B.31	Componente de declaração das rotas das telas do sensor	117
B.32	Componente do menu de rotas	119
B.33	Componente de estilo css	119
B.34	Componente genérico das telas	119
B.35	Componente da tela de sensores	120
B.36	Componente de listagem de sensores	121
B.37	Funções de manipulação das tabelas de sniffers	122
B.38	Componentes de estilo css	124
B.39	Componente de menu das execuções	126
B.40	Objeto de conexão cliente WebSocket	129

1 Introdução

1.1 Considerações Preliminares

Na competição anual *Latin American Robotics IEEE Standard Educational Kit* (SEK), os participantes enfrentam o desafio de construir robôs totalmente autônomos, projetados para resolver uma variedade de tarefas práticas, como o carregamento de cargas e o conserto de gasodutos, em ambientes competitivos diversificados.

O kit LEGO MINDSTORMS EV3 contém um computador/controlador programável (*brick*) com bateria, 4 portas de saída que podem ser usadas em motores e atuadores, bem como, 4 portas de entrada usadas por sensores de toque, ultrassônico e outros, sendo todas essas peças compatíveis com os blocos Lego, facilitando a montagem do robô.

As complexas arenas em que esses robôs atuam geram cenários que exploram os limites de funcionamento dos sensores LEGO. Por exemplo, um sensor ultrassônico angulado ou operando em distância muito próxima acaba realizando a leitura de valores inesperados que dão origem a bugs no comportamento do robô.

Esses bugs podem ser recorrentes e atenuados através de calibragem dos sensores, ou podem ser intermitentes, surgindo devido à leitura de valores inesperados pelos sensores, o que gera a necessidade de monitoramento contínuo desses.

Nesse contexto foi desenvolvido o sistema de telemetria do robô que consiste de um hardware *sniffer* acoplado entre o *brick* e os sensores. Este dispositivo lê os valores dos sensores e gera logs, que são transmitidos em tempo real para um aplicativo responsável pela plotagem gráfica desses valores. O aplicativo armazena os logs dos sensores juntamente com a gravação de vídeo da execução do robô, assim é possível rever o comportamento do robô e os logs no momento em que o bug ocorreu possibilitando a depuração dos erros intermitentes, a leitura dos valores em tempo real facilita a identificação de erros nos sensores auxiliando nas rotinas de calibragem.

Devido o restrito recurso computacional do *brick* a utilização de um dispositivo externo dedicado para leitura e transmissão do logs evita interferências na performance do robô.

O sistema de telemetria proporciona uma forma consistente para depurar os diversos bugs oriundos de valores inesperados nos sensores do robô, além de prover o monitoramento dos sensores sem impactar a performance do robô. Tudo isso facilita o processo de desenvolvimento dos robôs, beneficiando os competidores da competição *Latin American Robotics IEEE Standard Educational Kit*, além de também ajudar entusiastas

por robótica que utilizam o kit LEGO MINDSTORMS EV3.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um sistema integrado de hardware e software, que possa monitorar em tempo real o funcionamento de um sensor ultrassônico do Kit de Desenvolvimento LEGO MINDSTORMS EV3, que é utilizado para medir distâncias.

1.2.2 Objetivos Específicos

- Possibilitar a leitura precisa dos dados do sensor ultrassônico, visando melhorar a eficácia do monitoramento e a precisão na medição de distâncias;
- Implementar a plotagem gráfica dos dados do sensor em tempo real em dispositivos móveis, facilitando a análise visual e imediata das medições do sensor;
- Desenvolver um aplicativo que permita a gravação de vídeo simultaneamente à aquisição de dados do sensor, oferecendo uma visão abrangente do desempenho do robô;
- Assegurar que o sistema possua uma bateria com duração mínima de 20 minutos, garantindo operação contínua durante competições ou testes;
- Garantir um tempo de resposta máximo de 10 ms, crucial para a eficiência e a precisão na operação do robô em ambientes dinâmicos;

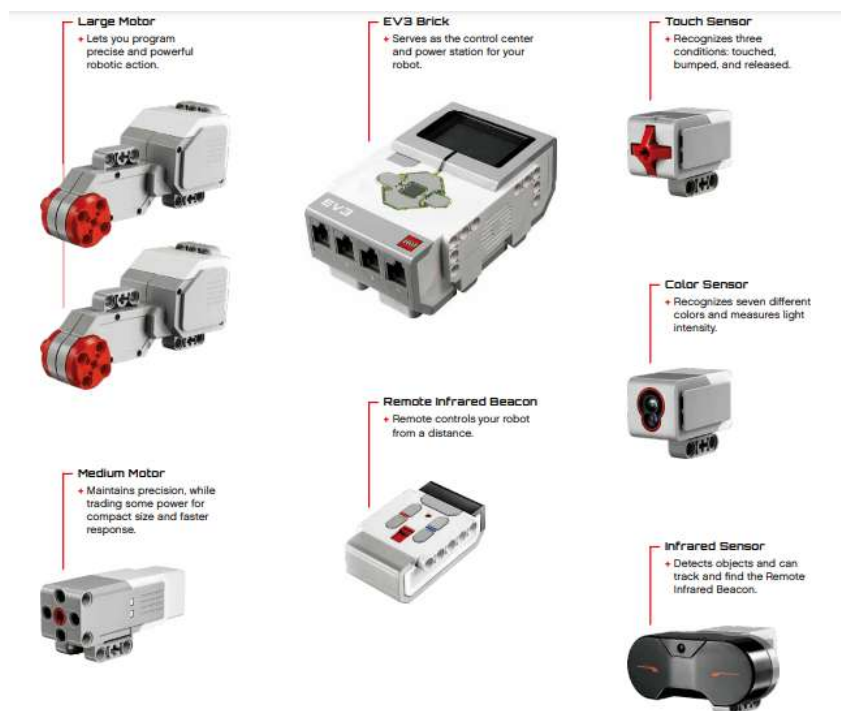
2 Fundamentação Teórica

Este capítulo apresenta o embasamento teórico crucial para o desenvolvimento do Sistema de Telemetria de Sensores LEGO MINDSTORMS EV3. O foco recai sobre a aplicação de técnicas específicas, incluindo a construção de Placas de Circuito Impresso (PCI), a prática de engenharia reversa e o desenvolvimento de software para dispositivos móveis.

2.1 LEGO MINDSTORMS EV3

O LEGO MINDSTORMS Education EV3 é um kit avançado destinado ao desenvolvimento de robôs, combinando peças LEGO com a inovadora tecnologia EV3. Este kit é amplamente reconhecido por seu valor educacional, principalmente em ensinar programação e conceitos interdisciplinares, além de ser frequentemente utilizado em competições de robótica segundo (SOUZA et al., 2018). Como descrito no manual (LEGO, 2013) o kit é composto por um computador/controlador chamado de Bloco EV3 (*Brick*), baterias, motores, sensores de ultrassônico, de toque, entre outros, acompanhados de cabos conectores e peças LEGO.

Figura 1 – Visão geral do kit LEGO MINDSTORMS EV3



Fonte: LEGO, 2023, p. 6

A estrutura física do robô é construída pela combinação de dessas peças LEGO

montadas juntamente com os sensores, motores e o Bloco EV3 como na Figura 2 retirada de (Fernando Veiga, 2015).

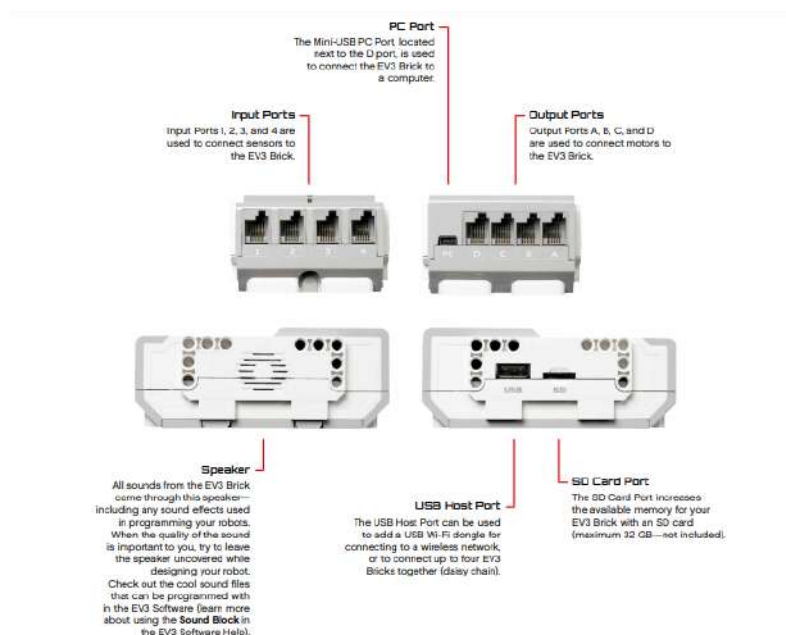
Figura 2 – Exemplo robô LEGO MINDSTORMS EV3



Fonte: (Fernando Veiga, 2015, p. 1)

A alimentação do robô pode ser feita pela bateria ou 6 pilhas AA acopladas ao *Brick*, os sensores e motores são plug and play bastando apenas serem conectados ao *Brick* através dos cabos RJ12 para funcionarem, o Bloco EV3 tem as portas RJ12 A,B,C, D, 1, 2, 3 e 4 sendo as letras para motores, e os números para os sensores, uma porta PC que é um mini usb, uma porta de cartão SD para expansão de memória não volátil, um alto-falante e uma porta USB como mostra na Figura 3, além de *bluetooth* o *Brick* também vem com seu próprio firmware instalado para que tudo funcione de forma integrada.

Figura 3 – Portas do kit LEGO MINDSTORMS EV3



Fonte: (LEGO, 2023, p. 9)

Para programar o robô é utilizado programação em blocos gráficos que pode ser feita pelo próprio Bloco EV3, esta também pode ser feita em um computador basta instalar

o software EV3 e com o *Brick* conectado através de um cabo usb pela porta PC, por *bluetooth* ou até por wi-fi com auxílio de um adaptador wifi USB, (LEGO, 2013). Assim, é possível utilizar diversos tipos de blocos como blocos de sensor, de dados para manipulação de variáveis, de operações matemáticas e lógicas, de fluxo, de ação para controle dos motores, entre outros que ao serem utilizados em conjunto formam o programa que é enviado para o robô e posteriormente executado.

Além da programação em blocos do software EV3 também é possível instalar e utilizar outros *firmwares* para programar o robô com linguagens de programação mais apropriadas para projetos complexos. Como mostra o livro *Beginning Robotics Programming in Java with LEGO MINDSTORMS* (LU, 2016) *leJOS* é um ecossistema para programar o robô em Java, instalando o *firmware* no Bloco EV3 através da porta SD e no computador juntamente com Eclipse IDE e seu plugin *leJOS*, basta escrever o programa Java no eclipse utilizando os métodos da API *leJOS* para ler os sensores e controlar os motores do *Brick*, compilar e enviar o programa para o robô. Também há alternativas com possibilidades mais amplas como o *ev3dev* (ev3dev, 2016a) que é uma distribuição linux baseada no Debian, este é instalado no cartão SD acoplado ao Bloco EV3, fornece um framework completo para controle do *Brick* para programar o robô com diversas APIs de várias linguagens como, *ev3dev-lang-python*, *ev3dev-lang-java*, *ev3dev-lang-cpp*, entre outras disponíveis em (ev3dev, 2016b).

2.2 Microcontroladores

Um microcontrolador é essencialmente um microcomputador em miniatura, incorporado em um único circuito integrado. Este dispositivo integra várias funcionalidades de processamento e controle, tornando-se um componente vital em inúmeras aplicações eletrônicas, conforme (ARAUJO; CAVALCANTE; SILVA, 2019). Dentro do microcontrolador, temos uma unidade de processamento e os componentes necessários para realizar tarefas de forma autônoma, como leitura de memória, gravação de dados e armazenamento de *firmware*, conversores de sinal analógico/digital e digital/analógico e portas programáveis. Entrada e saída para várias funções, como controlar outros dispositivos, fornecer interação com o usuário, etc.

As aplicações de microcontroladores são tão vastas que o limite é a imaginação do usuário. Esses pequenos controles estão no carro, televisão, telefone, impressora, forno de micro-ondas, ônibus espacial, brinquedo, etc. Algumas fontes estimam que uma casa média dos EUA tenha aproximadamente 250 microcontroladores.

A seguir estão alguns campos onde os microcontroladores são amplamente utilizados:

- Na indústria de imagem e som: processamento de imagem e som, controle de motores

que acionam um *blu-ray player*, gravadores, vídeo, etc.

- Em um computador: como controlador periférico. Por exemplo, controlar impressoras, plotters, câmeras, *scanners* de terminal, drives de disco, teclados, conexões (rede), mouse, etc.
- Na indústria de eletrodomésticos: controle de aquecedores, máquinas de lavar, fogões elétricos, aspiradores de pó, etc.
- Na indústria automotiva: Controle do motor, alarmes, controle do servofreio, ABS, etc.

2.2.1 ESP32

O ESP32 é um microcontrolador produzido pela Espressif Systems, uma multinacional pública de semicondutores sem fábrica, fundada em 2008, com escritórios na China, República Tcheca, Índia, Cingapura e Brasil. Conta com uma equipe de engenheiros e cientistas de todo o mundo, focados no desenvolvimento de soluções de comunicação sem fio de ponta, baixo consumo de energia e *AIoT*. Comprometidos em oferecer soluções seguras, robustas e com baixo consumo de energia. Com tecnologia e soluções de código aberto, permitindo assim, que os desenvolvedores usem as soluções da Espressif globalmente e construam seus próprios dispositivos conectados de forma inteligente, (ANN, 2022).

Como observado na Figura 4 o ESP32 com sua placa de desenvolvimento é um hardware bastante poderoso com diversas possibilidades de entrada e saída. De acordo com (KOYANAGI, 2018) na sua versão ESP32-WROOM-32UE, contamos com comunicação de 2.4 GHz Wi-Fi, *bluetooth* convencional e um módulo de baixa energia *bluetooth*, esse kit de desenvolvimento foi construído em torno da série ESP32 com um microprocessador Xtensa® dual-core de 32-bit LX6. Possui versões com 4, 8 e 16 MB de memória flash disponíveis, gerenciando 26 GPIOs (*General Purpose Input/Output*), com um riquíssimo conjunto de periféricos, bem como, antena PCI integrada ou conector de antena externa.

Segundo (MIRANDA, 2019) o ESP32 embora apresentando uma tecnologia bem mais recente que seus concorrentes, a robustez é notória demonstrando existir muitas vantagens com o uso do ESP32 em relação aos outros microprocessadores da sua categoria no mercado.

Tomando como referência as descrições acima, fica evidente a viabilidade do uso do ESP32 como microcontrolador para execução de projetos de baixo custo e alta conectividade aos componentes de um sistema de monitoramento.

a transferência a quente do desenho impresso e a posterior corrosão com ataque químico a base de Perclorato de Ferro.

Este método em que o arranjo do circuito é transferido para a superfície da placa usando papel especial e um aquecedor aquecido a cerca de 130°C é amplamente utilizado devido ao baixo custo de produção. No entanto, na maioria das vezes o processo utilizado mostra que não funciona bem, pois requer materiais de qualidade no processo e uma visão muito boa do espaço tridimensional do designer (CARSTENS; CARSTENS, 2015). Conforme descreve (PERCHÉ, 2013), a maior dificuldade técnica encontrada é a retransmissão precisa do circuito, podendo ser necessário reimprimir um novo. Outro fator importante é a corrosão, que quase sempre altera a forma real da pista de cobre, de tal forma que o circuito fica comprometido.

Diante de tais dificuldades e com o avanço da eletrônica, submeter-se ao processo de fabricação de PCI utilizando-se de uma máquina CNC tem grande importância para a prototipagem rápida.

Atualmente a tecnologia CNC é abundantemente aplicada em várias áreas da indústria, através de diversos tipos de máquinas e robôs, que aumentaram muito o processo de automatização. Um exemplo bem comum são as fresadoras CNC. Máquinas cuja ferramenta possui movimento de rotação e que permite movimentar a peça em um, dois, três ou mais eixos (lineares ou giratórios). Todas estas ações são comandadas por códigos de usinagem escritos em código G (linguagem de programação textual), padronizado pela norma ISO-1056:1975, que são interpretados através do CNC e que atuam nos elementos de movimentação. Sendo assim tem-se uma máquina elaborada para execução facilitada de peças prismáticas e superfícies complexas (CARSTENS; CARSTENS, 2015).

2.4 Engenharia Reversa

A engenharia reversa tem um impacto fundamental na indústria atual, e vai muito além de simplesmente estimular mais concorrência e inserir no mercado produtos menos caro conforme (WANG, 2011). A evolução industrial ocorre também com o uso da engenharia reversa. Antigamente o ciclo de vida dos produtos inventados geralmente durava séculos, vejamos a lâmpada elétrica em substituição das lanternas. Contudo, as invenções modernas tem um ciclo de vida médio muito mais curto. Em poucas décadas observamos o mercado de fitas VHS dar lugar as mídias digitais e logo em seguida, as produtoras de vídeo online expandiram-se praticamente por todo o planeta.

Embora o termo Engenharia Reversa seja um conceito abandonado pela linguagem técnica formal, devido a confusa comparação com pirataria, no tocante a este trabalho ela será utilizada para entendimento de características específicas de comunicação, resultando assim na elaboração de um novo produto capaz de aumentar as capacidades do produto

original, ou seja, um hardware extra usado juntamente com o hardware original, que não exclui ou deixa obsoleto seu predecessor.

Engenharia Reversa comporta uma larga diversidade de definições, em parte, devido a diferentes empregos, em parte, devido a diferentes processos adotados. Não há uma definição consensual do que seja ER. Mas as definições, variadas que sejam, comportam a observação de pelo menos duas etapas. Uma primeira se constitui na obtenção de informação que caracteriza e especifica o objeto da ação de ER, identificando seus componentes e seu padrão de inter-relacionamento. Na segunda, o objeto é representado em outra forma ou com um mais elevado nível de abstração. É uma atividade que não altera o objeto da ação. É um processo, como norma, não destrutivo, um processo de exame, não de modificação do objeto do exame (DIAS, 1997).

A engenharia reversa abrange uma quantidade considerável de atividades e está em constante evolução, ainda que esta disciplina envolva perto de todos os campos de tecnologias (aviação, química, medicina, ciência da computação, mecânica, eletrônica, etc.), este estudo convergirá apenas para a ciência de computação (análise de dados, protocolos de comunicação, etc.).

Segundo (RATHORE N. & JAIN, 2014) todo processo de engenharia reversa deve ser auxiliado por computador. A vista disso, o uso de máquinas que possam processar grandes quantidade de dados e entregar resultados otimizados é de suma importância e faz parte do processo de engenharia reversa.

Tanto a competência quanto a confiabilidade dos dados são essenciais para a engenharia reversa. (WANG, 2011) aponta que o julgamento da engenharia chamado com frequência para a discrepância entre as medições, devido a inconsistência instrumental e humana na prática da engenharia reversa. Alertando assim, para o uso de instrumentos calibrados por empresas especializadas e os cuidados na escolha dos profissionais que coletarão dados a serem analisados.

2.5 GitHub

GitHub foi criado em 2008 como um serviço Web que possibilitou a hospedagem de repositórios Git. Ele permite a colaboração mais facilmente com outras pessoas em um projeto. Ele faz isso fornecendo um local centralizado para compartilhar o repositório, uma interface baseada na web para visualizá-lo e recursos como *forking*, *pull requests*, *issues* e *wikis*, que permitem especificar, discutir e revisar as mudanças de forma mais eficaz (BELL; BEER, 2014).

O GitHub funciona através do Git que é um sistema de controle de versão, o qual funciona como um gerenciador de histórico de edições onde são guardadas várias versões do código. Linus Torvalds, criador do Linux, com o seu descontentamento com o BitKeeper

o sistema de controle de versão utilizado no desenvolvimento do *kernel* do Linux, resolveu desenvolver o Git em 2005 (AQUILES; FERREIRA, 2014).

O GitHub é muito mais que apenas um lugar para armazenar seus repositórios Git (BELL; BEER, 2014). Ele fornece uma quantidade grande de benefícios adicionais, concedendo ao usuário a capacidade de documentar bugs e especificar novos recursos que gostaria que sua equipe desenvolvesse, colaborar com diferentes fluxos do desenvolvimento, revisar os trabalhos em andamento e ver o progresso da equipe enquanto examina o histórico de *commits* observando assim como estão trabalhando.

Segundo (GITHUB, 2022) atualmente são hospedados mais de 200 milhões de repositórios no GitHub, com a maior parte deles sendo projetos de código aberto, é utilizada por mais de 83 milhões de pessoas e 90% das empresas do Fortune 100 utilizam a plataforma, bem como, possui mais de 4 milhões de organizações. Estas estatísticas mostram que o GitHub está entre os clientes de Git (*GUI - Graphic User Interface*) mais utilizados hoje em dia.

3 Metodologia

3.1 Métodos

Este capítulo detalha as ferramentas e métodos empregados no desenvolvimento deste projeto. As principais ferramentas incluem: Placa de Circuito Impresso para conexão de sensores; Osciloscópio para análise de dados; Microprocessador ESP32-WROVER-B como unidade central de processamento; IDE Visual Studio Code com o plugin PlatformIO e *Framework* Arduino para programação de *firmware*; Websocket para comunicação entre o aplicativo e o hardware; React Native para desenvolvimento de aplicativo móvel; e Git para controle de versão. Além disso, o projeto segue uma metodologia de desenvolvimento ágil, facilitando a adaptação às mudanças e a entrega contínua de valor.

3.2 Ferramentas Utilizadas

Para o desenvolvimento da aplicação de gerenciamento e monitoramento do hardware, será empregado o *framework* React Native, enquanto a construção do *firmware* se dará por meio do Arduino IDE. Ambos os desenvolvimentos serão guiados pela metodologia XP (*eXtreme Programming*), garantindo eficiência e adaptabilidade ao projeto.

A metodologia XP é valorizada por sua flexibilidade e adaptabilidade, adequando-se a desenvolvedores de diversos níveis de experiência e a equipes de variados tamanhos. Esta abordagem é particularmente eficaz para responder a mudanças rápidas nas necessidades do projeto (WILDT et al., 2015).

Algumas práticas da metodologia XP foram adotadas no desenvolvimento deste trabalho, sendo elas as seguintes atividades:

- **Planejamento:** Neste trabalho, foram estabelecidos os requisitos que serviram de base para o desenvolvimento do programa. E nesta fase, os requisitos foram priorizados, identificando quais seriam os primeiros e mais importantes para que a aplicação funcionasse.
- **Stand Up Meeting:** Reuniões semanais de curta duração para apresentar o andamento do desenvolvimento, fizeram parte de todo o processo.
- **Código Coletivo:** O código de todos os programas é disponibilizado no Sistema de Controle de Versões (Git) e todos os desenvolvedores tem acesso colaborativo, a fim de entender e contribuir com o desenvolvimento.

- Na padronização do código, a atividade de codificação do *firmware* foi realizada utilizando a IDE Visual Studio Code da Microsoft. Este editor de código fonte suporta diversas linguagens de programação e foi complementado pelo PlatformIO, um *plugin* eficaz para o desenvolvimento de sistemas embarcados. Para o desenvolvimento da aplicação móvel, optou-se pelo *Framework* React Native, escolhido pelo domínio demonstrado pelos desenvolvedores, e empregou-se Javascript para programação e WatermelonDB para consultas ao banco de dados.

3.2.1 Placa de Circuito Impresso

A Placa de Circuito Impresso (PCI) projetada para interceptar as comunicações entre os sensores e o *Brick* (o componente central de processamento) do LEGO EV3 foi fabricada usando um substrato FR4 de dupla face. No entanto, apenas uma face foi utilizada devido à simplicidade do circuito.

FR, abreviatura de *Fire Retardant* (retardante de chamas), denota que o FR4 é um tipo de material laminado de vidro, amplamente empregado na fabricação de PCIs devido à sua notável resistência mecânica e propriedades de retardância à chama.

O primeiro protótipo de circuito eletrônico é composto por dois divisores de tensão, utilizando cada um, dois resistores de $10k\Omega$, dois conectores SCKT-Nx-1-NXT Fêmea (LEGO), bem como, uma barra de pinos $1x40x17$ com seis pinos machos dispostos em uma posição adequada para conexão de pontas de prova, e por último uma barra de pinos $2x40x17$ com doze pinos machos para acoplamento de um conector 12 pinos fêmea, possibilitando assim através desse conector, deixar a PCI com opção de ter o Sistema de Telemetria sendo ligado em série ou em paralelo.

Como já dito, por se tratar de um circuito muito simples, no caso do protótipo, para a confecção das trilhas de contato do protótipo, optou-se por utilizar uma mini retífica manual. Esse método foi escolhido por sua praticidade e rapidez, adequando-se perfeitamente às necessidades de um circuito de natureza simplificada, sem a necessidade de recorrer a softwares de design, impressão em filme e processos químicos complexos.

3.2.2 Osciloscópio

O osciloscópio desempenhou um papel crucial na engenharia reversa, permitindo a interceptação e a análise das comunicações entre o *Brick* e os sensores do LEGO EV3. Essa análise viabilizou a captura do código de comunicação inicial, para o desenvolvimento e a implementação do *firmware* no ESP32.

O osciloscópio utilizado para tal finalidade pertence ao Laboratório de Eletrônica Digital do IFPB Campus Campina Grande, é um Osciloscópio de Bancada da marca

Tektronix, modelo MSO 2024B, possui 4+16CH MSO, com 200Mhz de largura de banda e 1GS/s de taxa de amostragem.

Para analisar a comunicação entre os sensores e o *Brick* do LEGO EV3, utilizamos dois canais do osciloscópio: um canal dedicado à transmissão de dados (TX) do sensor, e outro para a recepção de dados (RX) pelo sensor. Esse arranjo permitiu uma análise detalhada e bidirecional da comunicação.

O sinal, após o divisor de tensão, contém uma variação entre 0V e 1,7V, sendo 0V para o bit 0 e 1,7V para o bit 1. Diante disso, a análise focou na medição da menor duração em que o sinal se mantinha em 1,6V, com o objetivo de determinar o tamanho de um bit na frequência inicial de comunicação. Podendo dessa forma, ser possível determinar quantos zeros e quantos uns tinha na comunicação inicial. Como visto na documentação da LEGO sobre o EV3, o protocolo de comunicação usado em diversos sensores, especificamente no de cor e ultrassônico que utilizamos, é o padrão UART sendo 1 bit de Início, 8 bits de dados, sem paridade e 1 bit de parada.

3.2.3 ESP32-WROVER-B

Os microprocessadores centrais deste projeto são dois Xtensa® 32-bit LX6, integrados ao kit de desenvolvimento ESP32-WROVER-B. Eles têm a responsabilidade de gerenciar o protocolo de comunicação completo, capturar dados e disponibilizá-los via Wi-Fi.

A princípio, apenas o sinal de transmissão do sensor (TX) será analisado pelo microprocessador, através da porta de entrada 26 (GPIO26) após passar pelo divisor de tensão, pois o *Brick* do LEGO MINDSTORMS EV3 trabalha com a tensão de 5V e a tensão máxima de trabalho nas portas do ESP32 é de 3,3V. A conexão do pino TX do sensor está gravada na PCI de conexão com o nome de "6 TX".

3.2.4 Arduino IDE

O Arduino IDE é um ambiente de desenvolvimento de software de código aberto que facilita aos usuários a escrita e o upload de códigos em um ambiente de trabalho dinâmico e em tempo real. Esta ferramenta será crucial para programar o firmware necessário para o projeto. É um ambiente desenvolvido para que você tenha tudo (ou quase tudo) o que precisa para programar sua placa baseada nessa plataforma, escrevendo seus códigos satisfatoriamente, de forma rápida e eficiente. Como este código será posteriormente armazenado na nuvem, é frequentemente utilizado por aqueles que estão procurando por um nível extra de redundância. O sistema é totalmente compatível com qualquer placa de software do Arduino e placas da ESPRESSIF, caso do ESP32.

O Arduino IDE pode ser implementado nos sistemas operacionais Windows, Mac

e Linux. A maioria de seus componentes é escrita em JavaScript para facilitar a edição e a compilação. Embora sua intenção principal seja baseada em códigos de escrita, há vários outros recursos dignos de nota. Ele foi equipado com um meio de compartilhar facilmente quaisquer detalhes com outras partes interessadas do projeto. Os usuários podem modificar layouts internos e esquemas quando necessário. Existem guias de ajuda detalhados que serão úteis durante o processo de instalação inicial. Tutoriais também estão disponíveis para aqueles que podem não ter uma quantidade substancial de experiência com a estrutura do Arduino e do ESP32.

A linguagem de programação utilizada para escrever os códigos para Arduino e ESP32 é baseada nas tradicionais C/C++ (com modificações), possui um grau de abstração alto e uma série de bibliotecas que encapsulam a maior parte da complexidade do microcontrolador. Esse alto grau de abstração e o set de bibliotecas são os grandes responsáveis por fazer a programação mais intuitiva e rápida, pois não é necessário que o desenvolvedor conheça os registradores, os detalhes de memória e a dinâmica do processador.

3.2.5 Visual Studio Code

O Visual Studio Code, um editor de código-fonte versátil da Microsoft, será utilizado no projeto por sua compatibilidade com várias linguagens de programação. Sua capacidade de suportar C/C++, em particular, é vital para o desenvolvimento do firmware do ESP32. Os recursos incluem suporte para depuração, realce de sintaxe, conclusão de código inteligente, *snippets*, refatoração de código e Git incorporado. Os usuários podem alterar o tema, atalhos de teclado, preferências e instalar extensões que adicionam mais funcionalidades.

Na Pesquisa *Stack Overflow* de 2021, o Visual Studio Code foi classificado como a ferramenta de ambiente de desenvolvedor mais popular, com 70% de 82.000 entrevistados relatando que o usam.

O Visual Studio Code é um editor de código-fonte que pode ser usado com uma variedade de linguagens de programação, incluindo Java, JavaScript, Go, Node.js, Python, C++, C, Rust e Fortran. Ele é baseado na estrutura *Electron*, que é usada para desenvolver aplicativos da Web node.js que são executados no mecanismo de layout *Blink*. No entanto, vamos nos ater à linguagem de programação C/C++ para desenvolvimento do *firmware* do ESP32.

A capacidade do VS Code de instalar *plugins* é tamanha, que o torna a ferramenta mais poderosa de desenvolvimento do mercado. Dito isso, instalaremos algumas extensões para um melhoramento na performance de uso. São elas:

- PlatformIO

- C/C++ IntelliSense
- GitLens
- PlatformIO
- Live Share da Microsoft

3.2.6 PlatformIO

O PlatformIO, um ambiente de desenvolvimento integrado, se destaca por sua simplicidade e extensibilidade, oferecendo um conjunto de ferramentas profissionais para desenvolvimento. No projeto, será utilizado para compilar o código em diferentes plataformas de desenvolvimento, facilitando a criação e entrega do produto final.

O PlatformIO permite que o desenvolvedor compile o mesmo código com diferentes plataformas de desenvolvimento usando o *Only One Command* > *platformio run*. Isso acontece devido ao Arquivo de Configuração de Projeto (*platformio.ini*) onde você pode configurar diferentes ambientes com opções específicas (tipo de plataforma, configurações de upload de firmware, frameworks pré-construídos, build flags e muito mais).

A integração do PlatformIO acontece no Virtual Studio Code da Microsoft, ele é instalado como um *plugin* normalmente. Para utilização com o ESP32-WROVER-B é necessário algumas configurações, que são a criação de um projeto, a escolha da placa de desenvolvimento que deve ser selecionada é a opção Espressif ESP-WROVER-KIT e o framework Arduino. Após inicialização do projeto, no diretório raiz da estrutura de arquivos deve ser editado o arquivo *platformio.ini*, a fim de, configurar qual porta o ESP32 estará conectado no seu PC, para que os códigos do firmware desenvolvido possam ser upados diretamente para o microprocessador através do VS Code.

3.2.7 Websockets

Websockets é uma tecnologia avançada que estabelece um canal de comunicação bidirecional *full-duplex* sobre um *socket* TCP permanente entre um cliente e um servidor. Esta tecnologia é notavelmente mais rápida que as requisições HTTP convencionais, que incluem cabeçalhos, *cookies* e outros dados, contribuindo para maior latência na transmissão de dados essenciais.

A tecnologia de websockets permite a transmissão de dados de forma contínua e com baixa latência e por isso é amplamente utilizada para comunicações de tempo real como chats, jogos online, entre outros.

Por isso iremos realizar a transferência dos logs através de websockets, onde criaremos a conexão entre o ESP32 e o aplicativo de monitoramento, isso irá nos permitir

a transferência de informação rápida e segura sem perda de dados. Para o ESP32 no framework arduino será utilizado a biblioteca WebSockets para a criação do server, e para o aplicativo no *framework* React Native iremos usar a biblioteca nativa WebSocket para a programação do cliente.

3.2.8 React Native

React Native é um *framework* amplamente utilizado para o desenvolvimento de aplicativos móveis, conhecido por sua programação intuitiva em JavaScript. Ele se baseia em componentes reaproveitáveis com uma sintaxe similar ao HTML, facilitando o desenvolvimento e aumentando a eficiência do processo.

É um *framework* muito popular por ser *cross-platform* permitindo que o mesmo código seja reaproveitado para todas as plataformas *mobile*. O código javascript é renderizado para código nativo da plataforma, o que garante execução rápida e eficiente sobre os recursos nativos de *frontend* e *backend* da plataforma *mobile*.

Além de ter uma gama de ferramentas para facilitar o processo de desenvolvimento, como o npm que é um *package manager* para a fácil instalação de *libs* javascript para o React Native, Node e React. Por essas características esse *framework* foi escolhido para o desenvolvimento do aplicativo de monitoramento.

3.2.9 Expo

Expo é um *framework* que simplifica significativamente as configurações de ambiente para desenvolver aplicativos em React Native. Ele oferece acesso facilitado a diversas APIs nativas, como a câmera e a manipulação de arquivos de mídia, eliminando a necessidade de lidar diretamente com o código nativo da plataforma.

Isso é possível porque o Expo tem um aplicativo *mobile* que é baixado e instalado na plataforma, esse aplicativo já contém o código nativo necessário para o acesso aos recursos da plataforma, dispensando a necessidade de utilizar Android Studio para ligar com o código nativo.

Assim, utilizaremos o Expo para podermos desenvolver o aplicativo e testá-lo diretamente no *mobile* em tempo de desenvolvimento através de recursos de *hot reload*, além de facilitar os recursos nativos através de API's evitando códigos nativos da plataforma e permitindo utilizar apenas o javascript do React Native para programar o *app*.

3.2.10 Libs do React Native

O React Native conta com uma série de bibliotecas open source que são facilmente instaladas pelo expo ou npm, as bibliotecas do aplicativo serão utilizadas para armaze-

namento de dados, controle da câmera, *player* de vídeo entre outras funcionalidades do aplicativo de monitoramento.

SQLite, instalado via *expo*, é um banco de dados relacional *offline-first* para funcionar sem necessidade de instanciar um serviço online, de rápida inicialização devido sua característica *lazy loaded* que carrega os dados sobre demanda evitando o carregamento de grande quantidade de dados na inicialização do aplicativo, e rápido envio e gravação de registro de forma assíncrona e escalável. Este banco de dados será utilizado para armazenar os *logs* enviados em alta frequência do ESP32.

Para a gravação dos vídeos utilizaremos a *lib expo-camera*, e para o armazenamento e manipulação teremos a biblioteca *expo-media-library*, são duas *libs* intuitivas e eficientes em utilizar os recursos nativos da plataforma *mobile* além de serem facilmente instaladas pelo *expo*.

3.2.11 Git e GitHub

É extremamente importante que os códigos da aplicação sejam versionados para que se tenha o controle do desenvolvimento dos programas do sistema de forma segura, para isso será utilizado o Git por ser uma ferramenta intuitiva e amplamente utilizada por desenvolvedores a bastante tempo.

O Git permite que sejam criadas várias versões do código de um programa em *branchs*, isso permite que o desenvolvimento de diferentes partes do mesmo programa sejam separadas e depois integradas a *branch* principal comumente chamada de *master* que guarda a versão final do código fonte.

Para que os códigos desenvolvidos possam ser distribuídos e utilizados pelos membros da equipe é necessário que seu versionamento seja constantemente salvo de forma remota. O GitHub foi a ferramenta escolhida para a distribuição remota dos versionamentos feitos no git, essa escolha se deu, devido a fácil integração com o git funcionando como um reflexo remoto do seu versionamento.

Os repositórios com o projeto podem ser encontrados em:

- <https://github.com/judenilson/ifpb-pec-tcc>
- <https://github.com/antonio357/telemetry-mobile-app>

4 Resultados

O foco deste capítulo é detalhar o processo de desenvolvimento do projeto, abrangendo desde a criação do hardware e seu respectivo *firmware* - essenciais para a integração do 'cérebro' do robô com seus sensores - até o desenvolvimento do software mobile, responsável por exibir os dados coletados pelo hardware.

4.1 Hardware

Nesta seção é apresentada a construção de um protótipo de hardware que coletará os dados dos sensores, e transmitirá via websocket para um dispositivo móvel conectado na mesma rede.

4.1.1 Análise de Dados

No decorrer do desenvolvimento, tornou-se essencial analisar como os dados transitavam entre o 'cérebro' do robô (*Brick*) e o sensor de ultrassom. Utilizando um osciloscópio Tektronix MSO 2024B, capturamos dados em intervalos não superiores a 10 minutos para estabelecer um padrão de comunicação e identificar os dados mais relevantes para o projeto.

Para isso, com auxílio de um osciloscópio Tektronix MSO 2024B, foram feitas capturas dos dados em períodos variados, porém não maiores que 10 minutos, a fim de estabelecer um padrão sobre como a conexão entre os dispositivos se iniciava e quando o sensor estaria enviando a grandeza esperada, contudo uma quantidade irrelevante de dados foi encontrada e em seguida descartada, tendo como fundamento o princípio de que os dados relevantes para o projeto seriam apenas os dados que mediam a grandeza física esperada, ou seja, os dados importante para o sensor ultrassom são as medidas em centímetros que ele transmite para o *Brick* após o início da comunicação que dura menos de um segundo.

Como consta no manual do kit de desenvolvedor de hardware LEGO MINDSTORMS EV3 a velocidade inicial de transmissão e recepção de dados entre o *Brick* e o sensor é de 2400 BAUDS, e após o estabelecimento da comunicação inicial (*handshake*) essa velocidade sofre uma mudança dependendo do tipo de sensor utilizado, no caso do sensor ultrassom a velocidade é mantida em 57600 BAUDS.

O padrão de comunicação UART bidirecional foi escolhido pelo fabricante LEGO para os sensores e o *Brick* do MINDSTORMS EV3. Este protocolo permite comunicações assíncronas variando de 2400 bit/s a 460 Kbit/s, dependendo da porta utilizada, e inclui

configurações de 1 bit de parada, 8 bits de dados e sem paridade. Um desvio padrão de 3% é aceito, contudo um desvio maior pode resultar em problemas de conexão. Para estabelecer a comunicação UART entre o *Brick* programável EV3 e um dispositivo externo, uma sequência de comunicação específica deve ser seguida. Tal sequência de comunicação foi descartada pelo hardware desenvolvido, visto que, conforme mencionado anteriormente era irrelevante para a aquisição dos dados referentes a grandeza mensurada, e não podemos intervir no processo de comunicação entre o sensor e o *Brick*.

Uma interface de 6 fios é implementada para permitir que dispositivos externos enviem dados de volta para o EV3 Brick de várias maneiras, usaremos o padrão de cabo da LEGO que é uma criação proprietária de uma versão RJ12 6p com a trava levemente deslocada do centro do conector para a lateral, o uso do conector da LEGO é para facilitar a conexão dos sensores que já usam o devido cabo, cujo número da peça é 11145. A configuração de pinos implementada no conector das portas de entrada para MINDSTORMS EV3 Brick é a seguinte:

- Pino 1 - ADC@5V ref, 9V com resitor limitador
- Pino 2 - GPIO, com funcionalidade de identificação automática
- Pino 3 - Terra
- Pino 4 - VCC 5V
- Pino 5 - Digital E/S, SCL (I2C), UART RX
- Pino 6 - Digital E/S, SDA (I2C), ADC@5V ref, UART TX

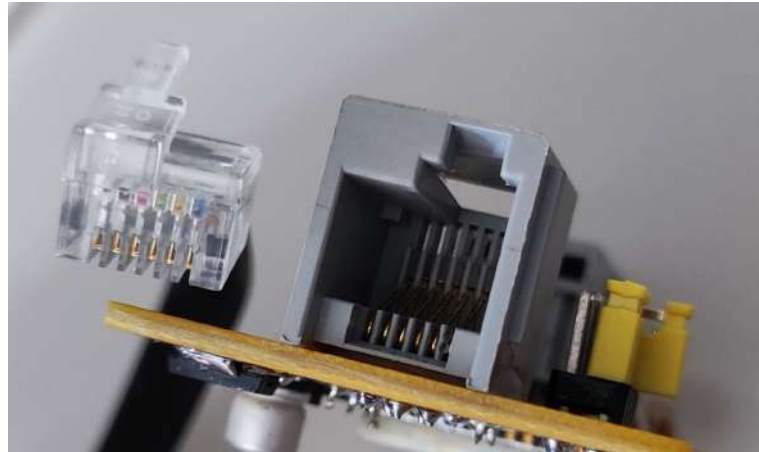
Vemos na Figura 5 a estrutura do *plug* do cabo LEGO, bem como o conector fêmea utilizado na confecção do protótipo em desenvolvimento.

4.1.2 Circuito Eletrônico

No design do circuito, utilizamos especificamente alguns pinos do conector de comunicação de dados do EV3 *Brick*: o pino 3 para aterramento do circuito, o pino 4 para alimentá-lo com 5V, e o pino 5 para a recepção de dados, seguindo o padrão da interface UART.

O ESP32-WROOVER-B possui uma tensão de operação de seus componentes que determinam, o uso nas portas de entrada, uma tensão máxima de 3,3V. Isso obriga a redução da tensão do sinal na porta de comunicação do EV3 Brick (pino 5, 5V) que segue para leitura no ESP32 na porta 14 (GPIO25). Nesse caso, foi considerado a utilização de um divisor de tensão a fim de manter a segurança e o correto funcionamento do ESP32,

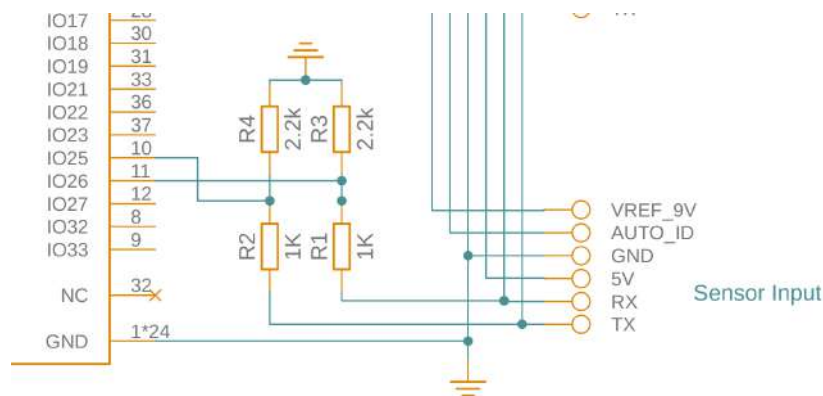
Figura 5 – Conector macho e fêmea da LEGO 11145



Fonte: Elaborado pelo autor, 2023

diante disso colocamos dois resistores de 1/8W no divisor de tensão com os valores 2,2K Ω e 1K Ω , conforme o circuito ilustrado na Figura 6.

Figura 6 – Divisor de tensão



Fonte: Elaborado pelo autor, 2023

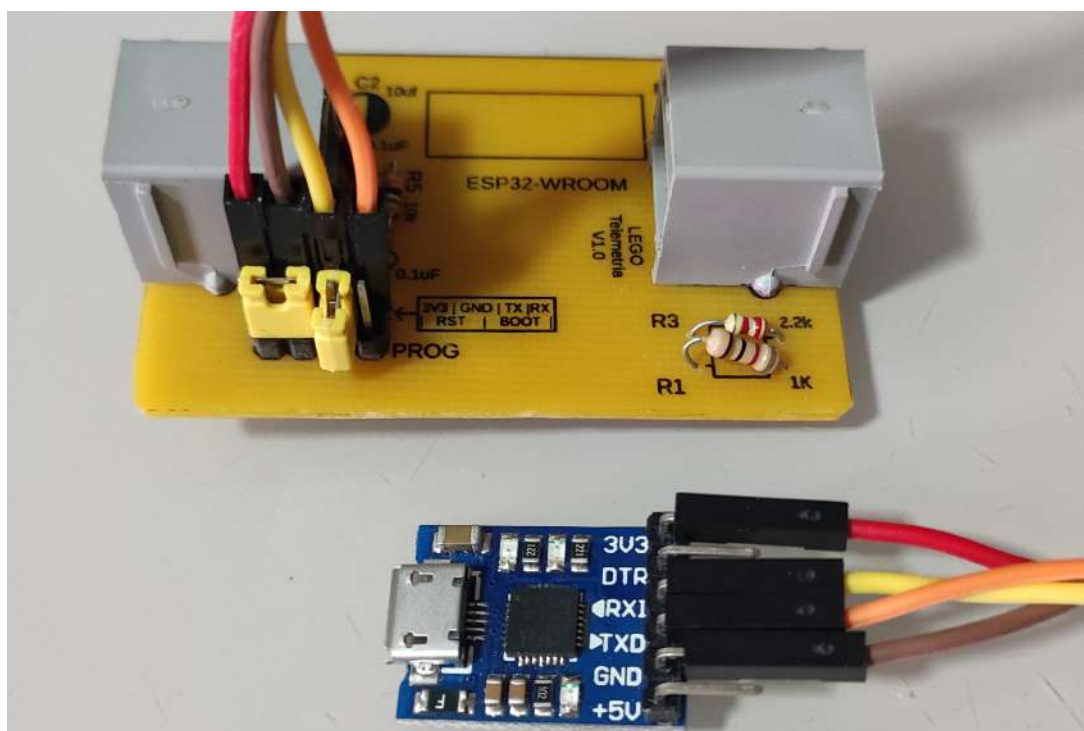
Para alimentação do ESP32, foi utilizado o pino 4 do conector do EV3 Brick o qual possui 5V e suporta uma corrente máxima de 1 A, contudo, o pino não pode alimentar diretamente o ESP32, pela limitação de trabalho do mesmo ser 3,3V, logo, foi utilizado um regulador de tensão AMS1117 T33, para reduzir a tensão da entrada de 5v para 3,3V. O regulador suporta 800mA de carga máxima, e o circuito consome no máximo 240mA, deixando o regulador com uma margem aceitável de trabalho, sem aquecimento, e sem necessidade de dissipador de calor. Juntamente com o regulador temos um capacitor eletrolítico de 10 μ F que suporta até 25V e outro capacitor de poliéster com o valor de 0.1 μ F utilizados como filtragem para melhorar a estabilização da tensão de saída do regulador e evitar ruídos indesejáveis.

A fim de manter uma linha de comunicação que pudesse proporcionar a análise de

dados na linha de comunicação TX proveniente do EV3 Brick, foi mantida uma conexão entre o pino 6 e a porta 15 (GPIO26) do ESP32, utilizando um divisor de tensão com as mesmas configurações da porta RX.

O ESP32 requer um circuito específico para sua ativação, incluindo um mecanismo para acionar o modo DFU (*Device Firmware Upgrade*), essencial para o upload do *firmware*. Este processo é manual, exigindo a extensão da conexão de várias portas do ESP32 para uma barra de pinos, facilitando o acesso aos pinos necessários para operações como *reset*, *boot* e upload de *firmware*. Assim sendo, foi estendida a conexão das portas Vin, GND, 40, 41, 23 e 9 do ESP32 para as barras de pinos 2x40x17 com 8 pinos machos, onde podem ser encontrado os pinos com 3,3V, GND, TX, RX para upload do firmware, bem como reset e boot. O jumper de reset deve ficar ligado sempre que o dispositivo estiver em modo operacional pois ele faz com que a porta 9 (EN) do ESP32 fique em nível lógico alto acompanhado de um resistor de 10K Ω , já o jumper de boot só deve ser ligado no momento de upload de firmware. Os demais pinos mencionados devem ser ligados em um conversor USB serial, que nesse caso o conversor utilizado tem como componente principal um Circuito Integrado CP210x da TFDI. Para melhor compreensão a conexão com o conversor serial é mostrada na Figura 7.

Figura 7 – Conexão para upload de firmware

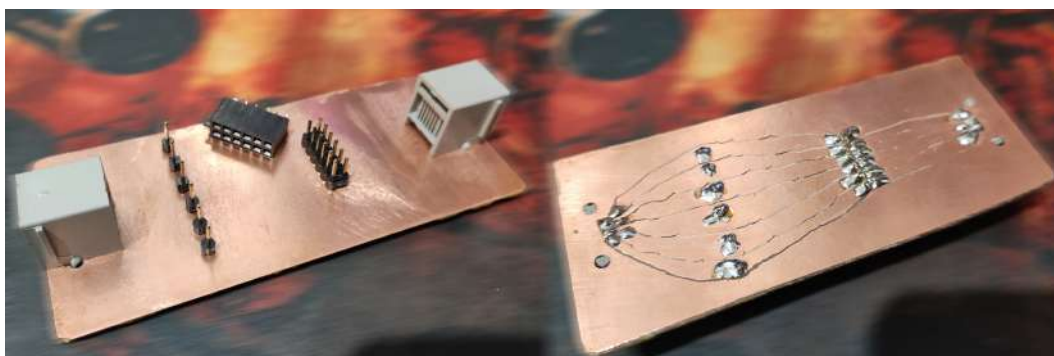


Fonte: Elaborado pelo autor, 2023

4.1.3 Prototipagem

Nos estágios iniciais de testes e análise de dados, criamos uma placa de circuito impresso manualmente, conforme visualiza-se na Figura 8. As trilhas foram desenhadas à mão e cortadas com o auxílio de uma mini retífica, trabalhando diretamente sobre o material de cobre. Este método, embora tradicional, provou ser eficaz para prototipar rapidamente um circuito simples, evitando os problemas comuns de mau contato em *protoboards*, que poderiam introduzir ruídos e interferir na interpretação dos dados.

Figura 8 – Primeira PCI, análise inicial de dados



Fonte: Elaborado pelo autor, 2023

Uma vez confirmada a correta aquisição dos dados, tornou-se necessário aprimorar a placa de circuito impresso (PCI). Este upgrade visou acomodar o circuito principal do ESP32 WROOM sem recorrer a uma placa de desenvolvimento externa. Assim, iniciamos o processo de criação de uma PCI que atendesse a requisitos específicos como tamanho compacto, uso de uma única face, componentes facilmente disponíveis no mercado e resistência a pequenos impactos.

O Fusion 360, aplicativo desenvolvido pela Autodesk que é uma plataforma de software de modelagem 3D, CAD, CAM, CAE e PCI na nuvem para projeto e manufatura de produtos, foi escolhido pela versatilidade e a capacidade de obter licenças educacionais. O uso do software tornou possível visualizar e manufaturar todo o hardware, possibilitando assim perceber de forma rápida os ajustes necessários para atender os requisitos do projeto, antes da etapa de corrosão da PCI.

Nem todos os componentes possuem suas dimensões modeladas em um arquivo 3D, logo, o desenho manual e a correta pinagem para uso na PCI foi realizada no Fusion 360 e será disponibilizada no GitHub do projeto para os devidos fins acadêmicos. A lista de componentes eletrônicos para o projeto da PCI restá relacionada abaixo:

- 2 resistores $10\text{k}\Omega$ 1/8W;
- 2 resistores $1\text{k}\Omega$ 1/8W;
- 1 capacitor eletrolítico $1\mu\text{F}$ 35v;

- 2 capacitores cerâmicos 0,1 μ F;
- 1 Circuito Integrado regulador de tensão AMS1117 T33;
- 1 módulo ESP32-WROOM;
- 1 placa de fenolite virgem, 40mm x 60mm, face única;
- 2 conectores RJ12 6p LEGO, fivela direita;

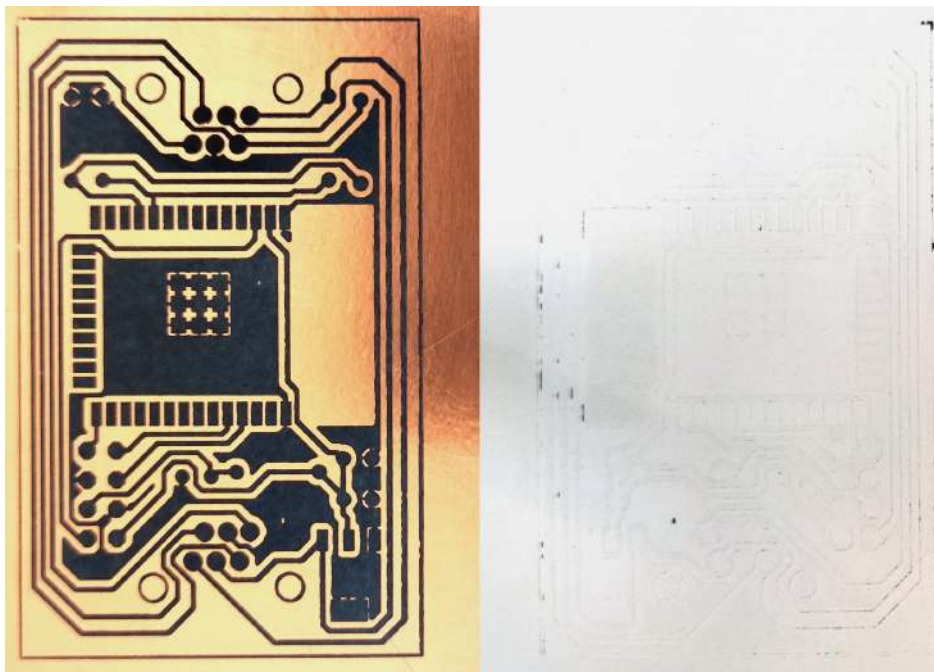
A primeira placa de circuito impresso usada como protótipo do projeto foi confeccionada de forma bem arcaica, visto que as intenções eram, inicialmente, apenas observar se os dados poderiam ser extraídos. Conforme o projeto foi avançando se fez necessário produzir uma PCI mais robusta e livre de interferências a fim de proporcionar um produto com qualidade final que pudesse ser comparado a qualidade de produtos comercializados, auxiliando assim na substituição da placa de desenvolvimento do ESP32 por uma placa com os fins esperados e apenas isso.

O processo de corrosão da placa de circuito impresso exigiu uma marcação precisa das trilhas. Optamos pelo método de transferência de toner de impressora a laser, combinando técnicas de transferência térmica e química. Isso envolveu o uso de um ferro de passar roupa e Propanona (acetona) para transferir o toner do papel fotográfico para a placa de fenolite, seguido de um processo de corrosão com perclorato de ferro para remover o cobre excedente e revelar as trilhas do circuito.

Após análise de algumas versões impressas difíceis de serem transferidas para a placa de circuito impresso, a seleção da trilha com no mínimo 16 mil de espessura e o espaçamento também com o mínimo de 16 mil foram suficientes para uma transferência da impressão laser numa qualidade com um nível de excelência acima do aceitável. Como o planejamento era fazer o circuito com um baixo custo, optamos por utilizar o método de transferência de toner de impressora laser. Esse método tem algumas maneiras de ser realizado, no entanto, duas das tais são bastante difundidas, uma é a transferência por calor a qual utiliza-se um simples ferro de passar roupa, encontrado facilmente em qualquer residência, a outra é a transferência química e mecânica, que consiste em umedecer o papel impresso com Propanona e friccionar sobre a placa de fenolite. A utilização de apenas uma das formas de transferência tem a probabilidade alta de acarretar erros durante a transferência do toner que está no papel para a placa de fenolite. Contudo, fizemos a mesclagem das duas técnicas e imprimimos o circuito numa folha de papel fotográfico, obtendo assim um resultado muito satisfatório, conforme pode-se observar na Figura 9. O teste com esse procedimento foi realizado cinco vezes e todos foram concluídos com sucesso e com alta qualidade.

Antes de tudo é preciso garantir que a placa de fenolite tenha o tamanho mínimo que caiba o circuito, o ideal é que sobre uma quantidade mínima de placa que será cortada

Figura 9 – Lado esquerdo toner já transferido, lado direito papel quase limpo



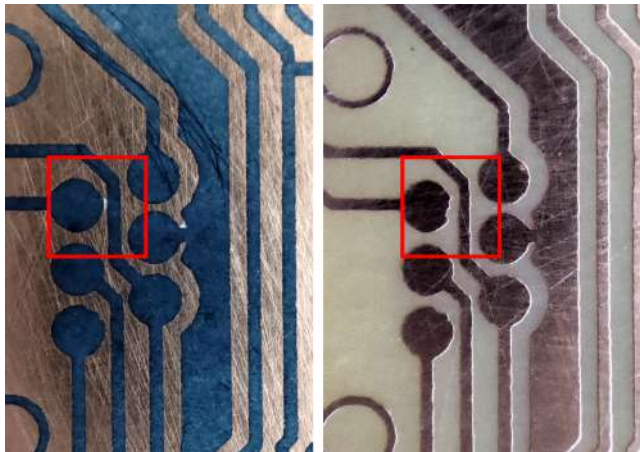
Fonte: Elaborado pelo autor, 2023

e descartada no final do processo, e que ela esteja efetivamente limpa, passando sobre a mesma uma esponja de aço sob água corrente e detergente neutro. Logo em seguida umedecer a placa com Propanona e posicionar o circuito impresso no papel com o toner em contato com a placa de fenolite. Aplicar Propanona mais uma vez sobre o papel e friccionar com o indicador com o cuidado de manter a mesma posição do papel durante todo o momento em que o dedo é passado, até q a Propanona seque completamente, nesse momento é perceptível que o toner já mantém certa aderência na placa de fenolite, após isso, aplicar mais Propanona umedecendo o papel e posicionar o ferro de passar roupa em temperatura máxima sobre o papel, fazendo movimentos circulares com uma pressão que seja exercida apenas com o peso do próprio ferro. Nessa etapa pode-se estimar um tempo de 10 segundos com o ferro de passar roupa sobre o papel a fim de que o toner tenha aderido completamente a placa de fenolite, o papel deve ser removido com cautela e uma das pontas deve estar presa para que o mesmo não solte aleatoriamente no final da remoção danificando a transferência.

Após a transferência do toner para o papel, pode ser necessária alguma correção no circuito, este deverá ser submetido a uma inspeção visual para garantir que a corrosão do cobre seja feita da forma devida, deixando apenas as trilhas que pertencem ao circuito. Se por ventura, se fizer necessário uma correção de trilha onde o toner não aderiu, este pode ser corrigido utilizando-se um marcador de tinta permanente com a ponta ultrafina. Caso a correção seja por contato entre trilhas, visto que, pode ficar celulose entre trilhas, isso é mais comum entre furos que estão muito próximos, deve-se remover o excesso utilizando um estilete para isso, a Figura 10 demonstra muito bem esse tipo de problema exibindo

um curto entre uma parte que será furada e uma trilha, e logo após a correção podemos perceber como ficou a placa depois de passar pelo processo de corrosão.

Figura 10 – Problema de curto na transferência de tonner



Fonte: Elaborado pelo autor, 2023

Com o circuito corrigido e pronto para a corrosão, utilizamos percloroeto de ferro diluído em água num recipiente plástico que caiba a PCI. Ao submergir a mesma no líquido devemos fazer leves movimentos para que o mesmo passe sobre a placa e possa remover o cobre com mais facilidade. O processo deve ser observado a olho nu e verificado se o cobre já foi completamente removido, percebendo-se tal remoção pela mudança de coloração e visualização clara da PCI que agora deve estar exposta. Nesse momento, remove-se a PCI do líquido de percloroeto de ferro e lava-a em água corrente, para que a mesma não permaneça em processo de corrosão. Após a lavagem seca-se a placa e aplica-se Propanona sobre o toner com auxílio de um algodão ou papel toalha, a fim de revelar o cobre que permaneceu na PCI com o circuito esperado. Depois disso, pode-se furar os buracos para inserção dos semicondutores e cortar a placa nos limites desejados.

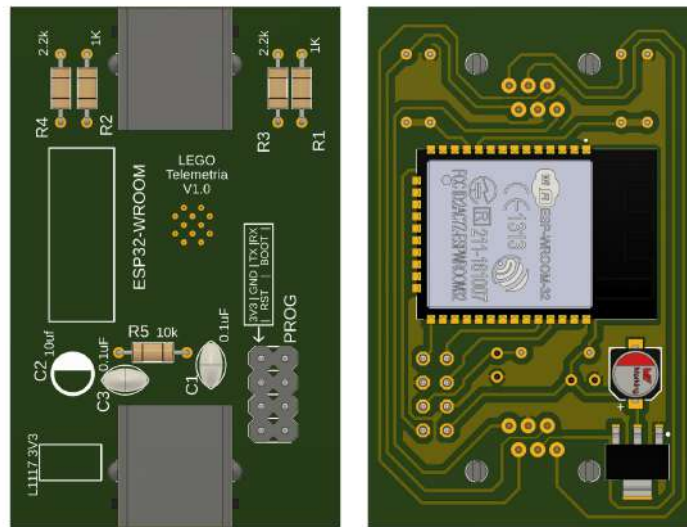
O posicionamento dos semicondutores na PCI deve ser feito de acordo com a Figura 11. Em sua maioria os componentes deverão ser posicionados na parte superior da placa, a qual não possui cobre, por se tratar de uma PCI de simples face, exceto pelo módulo do ESP32 que será colocado no lado cobreado da placa juntamente com o capacitor eletrolítico e o regulador de tensão.

4.1.4 Desenho Técnico

O uso do Fusion 360 foi crucial para o desenho técnico da Placa de Circuito Impresso (PCI) e do *case*, graças à sua capacidade de proporcionar uma visualização detalhada e tridimensional do projeto. Isso nos permitiu realizar ajustes precisos antes mesmo de iniciar a produção da PCI e do *case*, garantindo eficiência e precisão no desenvolvimento.

Durante nossas práticas no laboratório de robótica do IFPB, sabíamos que o circuito

Figura 11 – Disposição dos componentes, topo e fundo respectivamente



Fonte: Elaborado pelo autor, 2023

juntamente com seu case deveriam ser os menores possíveis, a fim de possibilitar o uso do mesmo acoplado a um robô montado para competições, diante disso, a placa foi projetada utilizando componentes pequenos e SMD, contudo embora seja um circuito que pode sofrer melhorias com relação ao tamanho, se confeccionado em uma placa de fenolite de dupla face, a possibilidade de fazê-lo numa placa de face única torna o projeto menos custoso e mais fácil de executar.

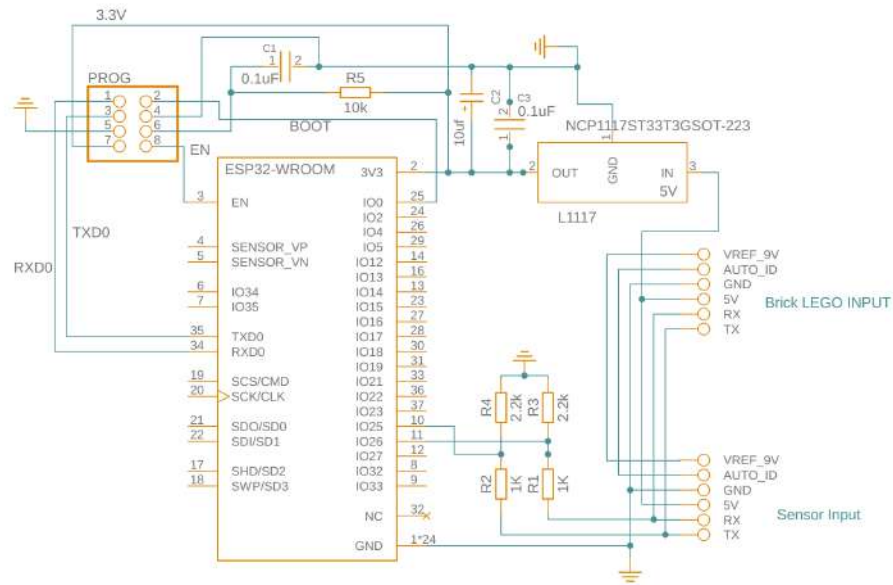
O esquemático foi completamente desenhado no Fusion 360 contemplando todas as conexões com o sensor, o *Brick* e o conversor USB/Serial. Embora a tensão seja proveniente de uma fonte de alimentação baseada em bateria, foi necessário reduzi-la para 3,3V que é a tensão de trabalho do ESP32, logo, a fim de manter o padrão de peças com pequeno tamanho foi escolhido utilizar o AMS1117 T33 e o capacitor do tipo SMD para estabilização da tensão. Na Figura 12 podemos observar o esquema do circuito por completo.

Após concluir o roteamento e a disposição dos componentes no Fusion 360, foi possível gerar o layout final da placa de fenolite. Esse layout, detalhadamente planejado, foi preparado para impressão, garantindo que todas as vias e componentes estivessem posicionados corretamente. Para a impressão no papel fotográfico, o layout com a descrição dos componentes foi espelhado de forma a assegurar a correta transferência do toner para a PCI segundo vemos na Figura 13.

Demonstrando a capacidade do Fusion 360 em possibilitar a visualização tridimensional do projeto temos em seguida a Figura 14 onde observamos a nitidez com que podemos perceber qualquer falha no projeto antes mesmo de confeccionar a PCI e soldar seus componentes.

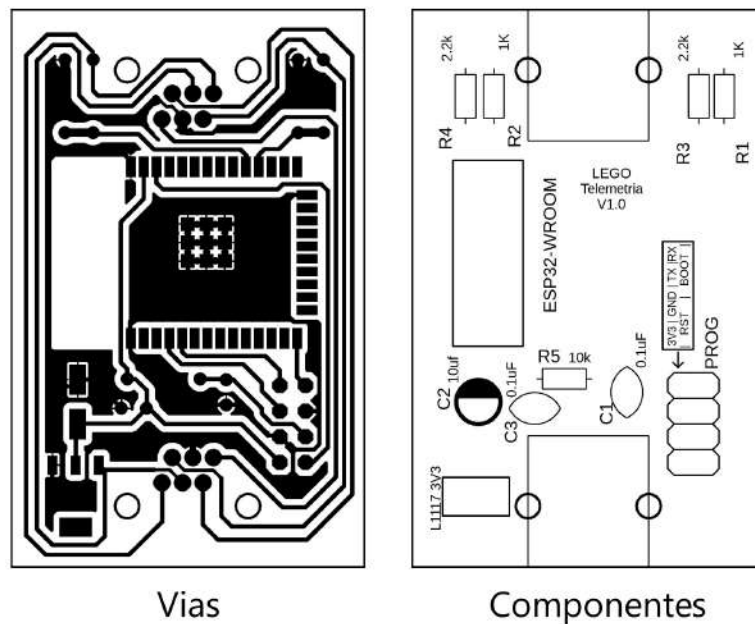
O design do case foi concebido para ser tanto simples quanto funcional, mantendo-se

Figura 12 – Esquemático completo do projeto



Fonte: Elaborado pelo autor, 2023

Figura 13 – Imagem do circuito gerado, visão fundo(vias) e topo(componentes)



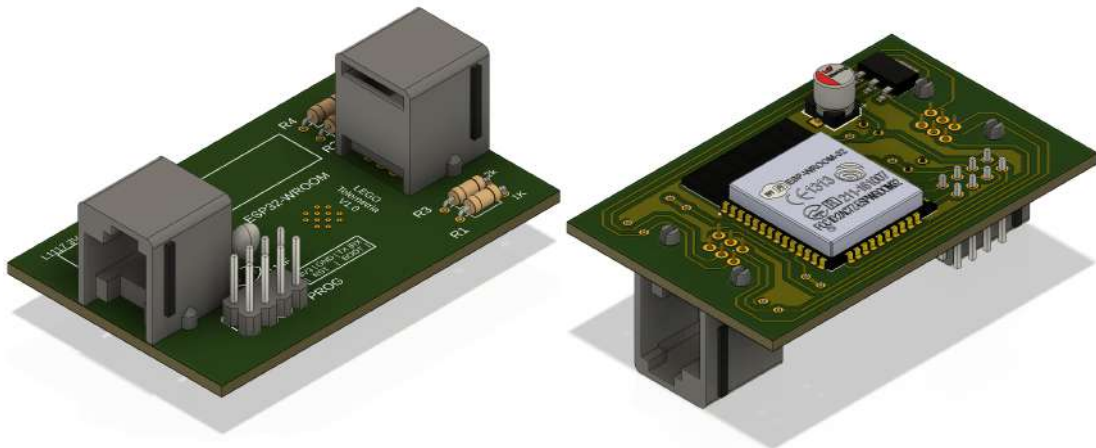
Vias

Componentes

Fonte: Elaborado pelo autor, 2023

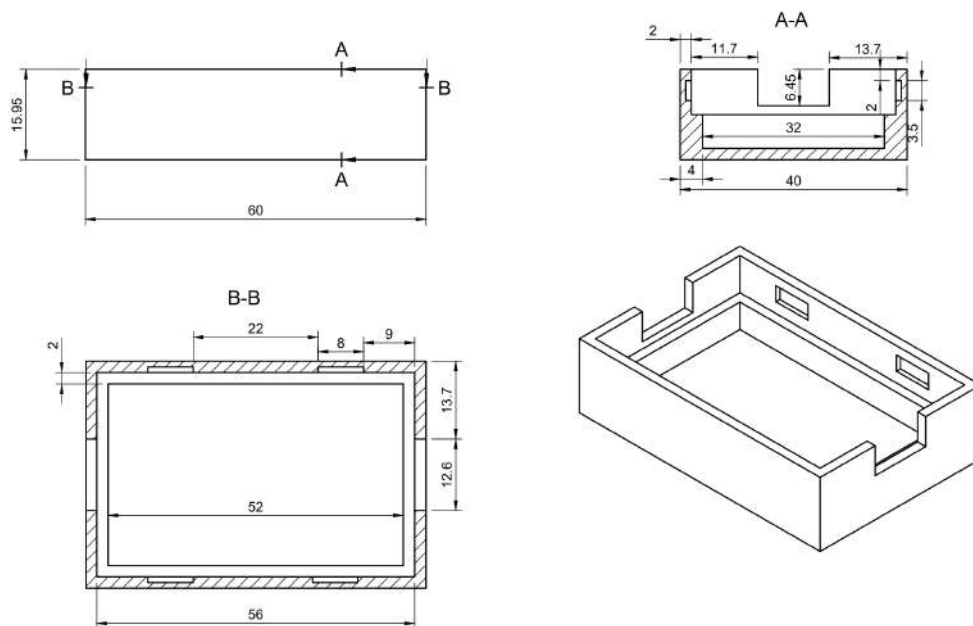
o mais compacto possível. Idealizado para impressão 3D com precisão de 0.2mm, o case foi projetado para ser auto-travante, acomodando a PCI de forma segura sem necessidade de parafusos ou ferramentas adicionais para o fechamento. O desenho técnico do case de montagem segue na Figura 15 com o esboço da tampa inferior e na Figura 16 com o layout da tampa superior, as medidas em ambos os desenhos estão em milímetros.

Figura 14 – Imagem 3D da PCI do projeto



Fonte: Elaborado pelo autor, 2023

Figura 15 – Case, tampa inferior



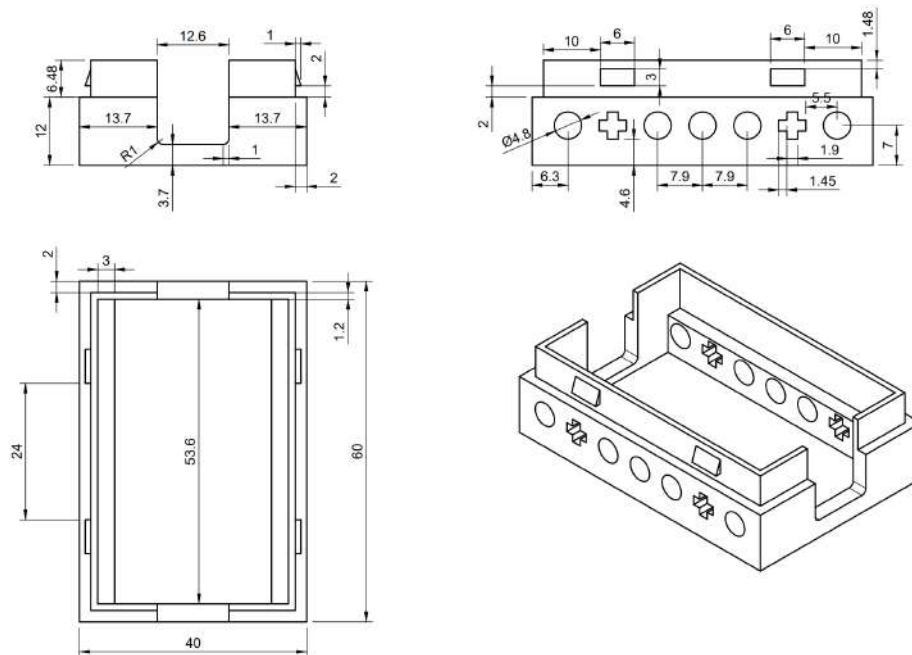
Fonte: Elaborado pelo autor, 2023

4.1.5 Firmware

O *firmware* para este projeto foi desenvolvido em C++, com adaptações específicas para a compatibilidade com o ESP32. Inspiramo-nos na plataforma Arduino para guiar nosso desenvolvimento, aproveitando a versatilidade da linguagem C++ para atender às necessidades específicas do nosso hardware. Com o auxílio da PlatformIO instalada como extensão do VSCode criamos um novo projeto cujo o código 4.1 ilustra a configuração utilizada na criação do projeto do firmware, as instruções utilizadas na configuração são:

- *platform*: Nome da plataforma de desenvolvimento;
- *board*: Nome da placa de desenvolvimento para compilação;

Figura 16 – Case, tampa superior



Fonte: Elaborado pelo autor, 2023

- *framework*: Linguagem utilizada para o desenvolvimento;
- *monitor_speed*: Velocidade de conexão do monitor serial;
- *upload_speed*: Velocidade de transferência de arquivos para o hardware em desenvolvimento;
- *upload_port*: Porta de comunicação onde está conectada a placa de desenvolvimento (em cada computador e em cada porta USB essa informação terá um nome específico);
- *monitor_port*: Porta de conexão do monitor serial;
- *lib_deps*: Caminho para opções de bibliotecas;

Código 4.1 – Configuração PlatformIO

```

1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:esp-wrover-kit]
12 platform = espressif32

```

```
13 board = esp32dev
14 framework = arduino
15 monitor_speed = 115200
16 upload_speed = 921600
17 upload_port = COM8
18 monitor_port = COM8
19 lib_deps = links2004/WebSockets@^2.3.7
```

O código 4.2 ilustra o núcleo do firmware, destacando a inclusão de bibliotecas cruciais para o projeto. Estas bibliotecas, como `Arduino.h`, `WiFi.h` e `WebSocketsServer.h`, fornecem uma gama de funcionalidades essenciais, desde a integração com o SDK do Arduino até capacidades avançadas de comunicação via Wi-Fi e Websockets.

A biblioteca `Arduino.h` é responsável pela inclusão principal do SDK do Arduino ela foi desenvolvida como software livre podendo ser modificada sob os termos do GNU *Lesser General Public License* conforme publicada pela *Free Software Foundation*. Outra biblioteca importada, com capacidade de instanciar Servidores e Clientes, enviar pacotes UDP através de Wi-Fi, conectar o hardware em redes abertas ou criptografadas (WEP, WPA), atribuir endereço IP estaticamente ou através de DHCP e gerenciar DNS, é a `WiFi.h` com suporte para o Wi-Fi do Esp32, a qual é baseada na `WiFi.h` da biblioteca Arduino Wi-Fi *Shield* modificada pelo Ivan Grokhitkov em dezembro de 2014, tudo sob os termos do GNU Lesser General Public License. Por fim, a biblioteca `WebSocketsServer.h` também distribuída sob os termos da GNU Lesser General Public License e desenvolvida pelo Markus Sattler em 2015, a referida biblioteca é capaz de realizar conexões persistentes entre um cliente e um servidor permitindo assim a comunicação bilateral entre ambas as partes utilizando uma conexão TCP. Por fim, o arquivo `wifi_router.cpp` é um arquivo contendo os dados de conexão com o roteador local, apenas para fins de teste.

Código 4.2 – Cabeçalho Firmware

```
1 #include <Arduino.h>
2 #include <WiFi.h>
3 #include <WebSocketsServer.h>
4 #include <../personal_configs/wifi_router.cpp>
5
6 #define RX_PIN 26 //Pino 26 RX DO SENSOR!
7 #define TX_PIN 25 //Pino 25 TX DO SENSOR!
```

Os *#defines* identificam os pinos que serão utilizados pelo hardware para analisar o tráfego de dados veiculado entre o sensor e o *Brick*, nesse caso, definidos como `RX_PIN 26` o pino que será conectado ao pino de recepção de dados do Sensor, e o `TX_PIN 25` como sendo o ponto de conexão com o pino de transmissão de dado do sensor. Embora tenha sido colocado no escopo da programação e esteja ligado no circuito eletrônico do projeto, o pino 26 não será levado em consideração na programação atual do firmware,

visto que a solução para o problema proposto foi encontrada utilizando-se apenas o pino 25, o qual recebe os dados provenientes do sensor, bastando para analisar a comunicação. Contudo, mesmo com um custo adicional para elaborar uma conexão com a porta RX do sensor, tal custo foi irrisório diante da possibilidade de algum benefício num update de *firmware* ser realizado, como por exemplo a capacidade de implementação de outros sensores e funcionalidades.

No próximo trecho observado no código 4.3 podemos analisar uma função que fará a verificação do estado de transmissão de dados do sensor. No caso estudado o sensor envia um sinal alto durante o repouso ou durante o intervalo entre um pacote de bits e outro. A fim de garantir que transpássemos qualquer ruído indesejado, bem como, garantir também que o sinal analisado não vai ser entre o último bit do pacote e o estado de repouso, foi adicionado um delay de 5µs que encontramos na linha 2. Em seguida, iniciaremos uma variável que funcionará como um contador, o qual será incrementado em uma unidade sempre que o loop verificar a porta TX_PIN e esta estiver no estado alto, após 10 unidades contadas no tempo de 14µs correspondentes a 10 bits (pacote de repouso) retornamos a função com o estado de verdadeiro, resultando na informação de que o sensor poderá a partir deste instante enviar o bit de início de transmissão do pacote de dados, e assim começar o processo de leitura dos bits contendo os dados desejados.

Código 4.3 – Análise de estado

```
1 int estadoAltoRepouso(){
2   ets_delay_us(5);
3   int count = 0;
4   while(1){
5     if (digitalRead(TX_PIN)){
6       count ++;
7     } else {
8       count = 0;
9     }
10    if (count >= 10){
11      return 1;
12    }
13    ets_delay_us(14);
14  }
15  return 0;
16 }
```

A leitura dos dados do sensor que serão utilizados na aplicação serão obtidos segundo o código 4.4 onde, na linha 2 é verificado o estado do sensor conforme explicado no parágrafo anterior, com o valor verdadeiro nesse parâmetro entramos no estado onde o dispositivo deve verificar o instante em que o sensor muda o sinal anteriormente alto para baixo, dando início a um atraso de 5µs a fim de esperar o tempo de transição da informação

evitando ruídos, em seguida serão iniciadas algumas variáveis, sendo i um controle do número de bits que serão lidos, no caso 40 bits de informação, a variável *bufferS* já foi iniciada no escopo do código como uma variável global do tipo *String* que será o *buffer* de armazenamento do sinal, e a variável *numero* que também está iniciada no escopo e é do tipo Inteiro de 16 bits que armazenará o número a ser transmitido para o aplicativo.

A partir daí, na linha 10 do código 4.4, temos um *loop* controlado pela variável i o qual inicia uma variável j esta por sua vez controla-rá a taxa de amostragem, que no caso deste código serão 10 amostras para cada bit, evitando ruídos. Ao entrar no loop de amostragem a entrada de sinal é lida e caso o estado seja alto ela incrementa a variável bit caso o sinal seja baixo ela decremenata, ao final do *loop* temos a condição que verifica se a variável bit for maior que zero, implicando assim afirmar que o sinal obtido é referente ao bit 1, bem como se o valor da variável bit for menor que zero, o sinal é referente a um bit 0, armazenando os valores no *bufferS*. A cada final de *loop* a leitura é pausada a fim de esperar o próximo bit, que de acordo com a velocidade de transmissão do sensor o tempo de espera dura 14µs. Após o *bufferS* ter recebido o pacote completo (40 bits) temos que filtrar os dados retirando apenas o número correspondente ao valor medido pelo sensor, número este que encontra-se entre os bits na posição 10 e 19 e entre 20 e 29 do pacote, entretanto, os dois pacotes de dados de 8 bits cada, que resultam em 2 bytes devem ter sua ordem invertida para assim gerar um número composto por 16 bits a fim de transmitirmos para o aplicativo.

Diante dessa inversão de pacote, a partir da linha 31 do código 4.4, temos dois *loops* que invertem os mesmos modificando os bits da variável *numero* de acordo com o *bufferS* aplicando um operador *or* com um bit deslocado para a esquerda de acordo com a posição dada pela variável *incremento* e o estado conforme o *bufferS*. Por fim, desprezando valores lidos provenientes de ruídos e outras características que possam gerar um dado incoerente, armazenamos o valor do número obtido na variável global *valorDoSensor* apenas se o mesmo for maior ou igual a zero, pois o sensor não lê valores negativos, visto que se trata de uma grandeza física impossível de ter essa característica.

Código 4.4 – Obtendo Dados

```
1 void lendoDados(){
2   if (estadoAltoRepouso()){
3     while(1){
4       if(!digitalRead(TX_PIN)){
5         ets_delay_us(5);
6         int i = 0;
7         bufferS = "";
8         numero = 0;
9
10        while(i < 40){
11          int j = 0;
```

```
12     int bit = 0;
13     while(j <= 9){
14         if (digitalRead(TX_PIN) == 1) {
15             bit++;
16         } else {
17             bit--;
18         };
19         j++;
20     }
21     if (bit > 0){
22         bufferS += '1';
23     }
24     else {
25         bufferS += '0';
26     }
27     i++;
28     ets_delay_us(14);
29 }
30
31 int incremento = 0;
32 for (int i = 11; i < 19; i++) {
33     if (bufferS[i] == '1'){
34         numero |= 1 << incremento;
35     }
36     incremento++;
37 }
38 for (int i = 21; i < 29; i++) {
39     if (bufferS[i] == '1'){
40         numero |= 1 << incremento;
41     }
42     incremento++;
43 }
44 if(numero >= 0){
45     valorDoSensor = numero;
46 }
47 break;
48 }
49 }
50 }
51 return;
52 }
```

No *sniffer* o serviço de *websocket* é implementado pelo objeto "WebSocketsServer" que é instanciado na inicialização do dispositivo. Para que esse *websocket* possa receber as mensagens, precisamos definir o código da função de *callback* "receiveMsg" dos eventos dessas mensagens como mostra no código 4.5.

Código 4.5 – Inicializando Websocket server

```
1 WebSocketsServer webSocket = WebSocketsServer(81);
2 void setup() {
3     webSocket.begin();
4     webSocket.onEvent(receiveMsg);
5 }
```

Dessa forma o *sniffer* consegue interpretar comandos recebidos como mensagens de texto, como mostra no código 4.6 esses comandos podem ser “star logs”, “stop logs” e “ports” que servem respectivamente para sinalizar que deve enviar os dados lidos dos sensores, sinalizar que deve parar de enviar os *logs* e enviar a informação das portas com sensores conectados.

Código 4.6 – Callback Eventos Websocket server

```
1 void receiveMsg(uint8_t num, WStype_t type, uint8_t *payload, size_t length) {
2     switch (type) {
3         case WStype_TEXT:
4             char payloadString[strlen((char *) (payload))];
5             strcpy(payloadString, (char *) (payload));
6             if (strcmp(payloadString, "start logs") == 0) {
7                 send_log = true;
8                 time_initial_offset = millis();
9             }
10            else if (strcmp(payloadString, "stop logs") == 0)
11                send_log = false;
12            else if (strcmp(payloadString, "ports") == 0) {
13                webSocket.sendTXT(num, "{\"connectedPorts\":{\"port1\\\", \"port2\\\"}}");
14            }
15        };
16    }
```

Ao receber o comando "start logs", o *sniffer* imediatamente ajusta a variável "time-initial-offset" para marcar o *timestamp* do momento em milissegundos. Este procedimento, realizado através da função "millis()" do *framework*, serve como ponto de referência para o cálculo do tempo nos *logs* transmitidos. Esse mecanismo é ilustrado no código 4.7, onde o pacote de *logs*, um objeto json, é preparado e enviado ao cliente *websocket* no aplicativo móvel.

Esse código de envio de dados 4.7 cria o pacote de *logs* que é um objeto *json* no formato 4.8, que é transformado em *string* e enviado para o *websocket* cliente do aplicativo móvel pelo método “broadcastTXT”.

Código 4.7 – Enviando os Dados

```
1 void enviandoDados(){
2     webSocket.loop();
```

```
3 int bufferDelay = 20;
4 if (send_log && websocket.connectedClients())
5 {
6     String logs = "{\logs\":{\port1\":[";
7     for (int i = 0; i < bufferDelay; i++) {
8         if (i == bufferDelay - 1) logs.concat("{\value\":" + String(valorDoSensor) + ",\time\":" +
9             String(logTimerSimulator) + "}");
10        else logs.concat("{\value\":" + String(random(10)) + ",\time\":" + String(logTimerSimulator) +
11            "},");
12        logTimerSimulator++;
13    }
14    logTimerSimulator -= bufferDelay;
15    logs.concat(",\port2\":[");
16    for (int i = 0; i < bufferDelay; i++) {
17        if (i == bufferDelay - 1) logs.concat("{\value\":" + String(valorDoSensor) + ",\time\":" +
18            String(logTimerSimulator) + "}");
19        else logs.concat("{\value\":" + String(random(10)) + ",\time\":" + String(logTimerSimulator) +
20            "},");
21        logTimerSimulator++;
22    }
23    logs.concat("]}}");
24    websocket.broadcastTXT(logs);
25    counter++;
26    delay(bufferDelay);
27 } else {
28     logTimerSimulator = 0;
29 }
```

4.2 Aplicativo móvel

Esta seção é dedicada ao desenvolvimento do aplicativo móvel, cuja principal função é estabelecer uma conexão com o *sniffer* e receber registros em tempo real dos sensores. O aplicativo foi projetado para realizar a plotagem gráfica dos dados dos sensores em tempo real e registrar a execução do robô, armazenando esses dados para análises e depurações posteriores.

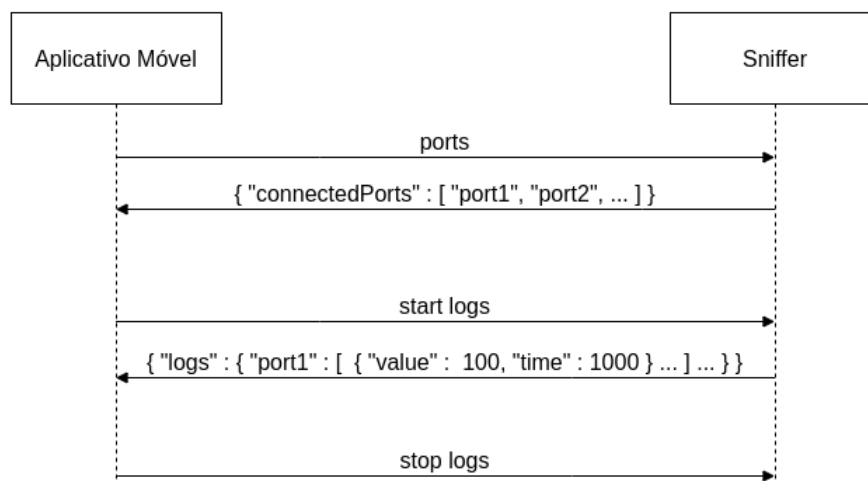
4.2.1 Comunicação WebSocket

A comunicação entre o *sniffer* e o aplicativo móvel, ambos conectados ao mesmo roteador, é realizada através de um cliente *WebSocket* integrado ao aplicativo. O cliente se conecta ao servidor *WebSocket* do *sniffer*, utilizando a biblioteca *WebSocketsServer.h* para facilitar a troca de mensagens de texto. No lado do aplicativo, a comunicação é gerenciada

pela biblioteca WebSocket nativa do React Native, que oferece uma interface simplificada para conexão e troca de mensagens.

Com isso a comunicação entre o *sniffer* e o aplicativo se dá pela troca de mensagens de texto em que o aplicativo manda *strings* de comandos para o *sniffer* e este devolve dados de objetos *json* como mostra na ilustração da Figura 17:

Figura 17 – Comunicação Websocket



Fonte: Elaborado pelo autor, 2023

Assim que a conexão entre o aplicativo e o *sniffer* é estabelecida, o aplicativo envia o comando "ports" ao *sniffer*. Este, por sua vez, responde com um objeto json contendo uma lista das portas onde os sensores estão conectados. Essa informação é crucial para o aplicativo associar os *logs* recebidos aos respectivos sensores. Além disso, o aplicativo envia o comando "start logs" para iniciar a transmissão contínua dos dados pelo *sniffer*, que persistirá até que receba o comando "stop logs", encerrando a transmissão.

4.2.2 Logs e Buffers

A transmissão de *logs* é realizada de maneira instantânea e contínua pelo *sniffer*. Após a leitura dos dados pelos sensores, o *sniffer* envia um pacote de *logs* na forma de um objeto JSON, como descrito no código 4.8. Este objeto JSON correlaciona os identificadores das portas dos sensores com os valores lidos e os respectivos *timestamps*, medidos em milissegundos desde o início das transmissões por meio do comando "start logs". Por exemplo, como mostrado no código, "port1" registra dois *logs* com valores e *timestamps* específicos.

Código 4.8 – Pacote de Logs

```

1 { "logs" : { "port1" : [ { "value" : 100, "time" : 1000 },
2                       { "value" : 200, "time" : 2000 } ],
3   "port2" : [ { "value" : 300, "time" : 1000 },
4               { "value" : 400, "time" : 2000 } ] } }
  
```

Quando os *logs* são recebidos pelo aplicativo, eles são armazenados em dois *buffers* distintos. Um dos *buffers* é destinado à geração dos gráficos dos sensores, enquanto o outro é utilizado para a gravação persistente dos *logs* no banco de dados. Esses *buffers* operam com a lógica FIFO ("first in, first out"), garantindo que os dados sejam processados na ordem em que foram recebidos dos sensores.

4.2.3 Gráficos dos sensores

A geração dos gráficos dos sensores ocorrem em tempo real de forma contínua conforme os pacotes de *logs* são recebidos e acumulados no *buffer* dos gráficos que tem a estrutura como descrita no código 4.9:

Código 4.9 – Buffer dos Logs

```
1 { "port1" : [ { "value" : 100, "time" : 1000 },
2           { "value" : 200, "time" : 2000 } ],
3   "port2" : [ { "value" : 300, "time" : 1000 },
4           { "value" : 400, "time" : 2000 } ] ] }
```

Este *buffer* é um objeto do mesmo formato do pacote de *logs*, em que temos as listas de *logs* relacionadas aos identificadores das portas dos sensores. Dessa forma a *thread* de renderização dos gráficos que executa continuamente vai consumindo e transferindo esses *logs* para os gráficos respectivos de cada sensor.

Como a maioria das bibliotecas de geração de gráficos para aplicativo android são projetadas para gráficos estáticos, não encontramos bibliotecas disponíveis para geração de gráficos em tempo real, inicialmente esse foi um grande desafio de performance, porque os gráficos das bibliotecas baseados em componentes *react* precisavam de atualização de estados de componente em tempo real para nossa aplicação, e essas atualizações ocorrem constantemente fazendo o aplicativo apresentar lentidão e travamentos.

Para resolver isso buscamos por bibliotecas de geração de gráficos em alta performance, a maioria eram baseadas em renderizar os gráficos em *canvas*, as opções eram escassas e algumas apresentaram problemas de compatibilidade com outras bibliotecas do projeto relacionadas a renderização da interface gráfica.

Nessa busca encontramos a biblioteca Skia que disponibiliza *APIs* de alta performance para desenhos em objetos *canvas*, ela é de código aberto sendo utilizada como motor de geração dos gráficos para várias plataformas e aplicações como Google Chrome, ChromeOs, Android entre outras, além de ser compatível com o ecossistema expo.

Com a alta performance e baixo consumo de memória da biblioteca Skia, conseguimos renderizar até 10000 *logs* por gráfico de sensor com desempenho satisfatório. Para isso

o desenho do gráfico é descrito em um componente “Path” incorporado como componente filho do componente “Canvas” como mostra no código 4.10:

Código 4.10 – Componente do Gráfico

```

1 chartPath = Skia.Path.Make();
2 < Canvas mode = 'continuous' >
3     < Path path = { chartPath } style = "stroke" />
4 </ Canvas >

```

Observe que o componente “Canvas” precisa ter a propriedade “mode” configurada como “continuous” para que todas as alterações de desenho no objeto “chartPath” sejam renderizadas pelo Canvas continuamente.

Para criar o gráfico dos sensores, utilizamos o método "chartPath.lineTo(x, y)" no componente "Canvas". Este método desenha linhas até as coordenadas (x, y) especificadas, baseando-se nos *logs* do *buffer*. As coordenadas x e y são calculadas transformando os valores de "time" e "value" dos *logs* em coordenadas relativas ao tamanho do canvas. A fórmula utilizada leva em conta a altura e o comprimento dos eixos do canvas, bem como o valor máximo lido pelo sensor e o intervalo de tempo dos *logs*.

Para converter o “value” do *log* para uma coordenada y relativa do gráfico utilizamos a fórmula $y = vl(n) \times \frac{ayc}{vmax}$, em que $vl(n)$ é o “value” do *log* atual, *ayc* é a altura do eixo y do *canvas* em *pixels*, e *vmax* é o valor máximo retornado pelo sensor.

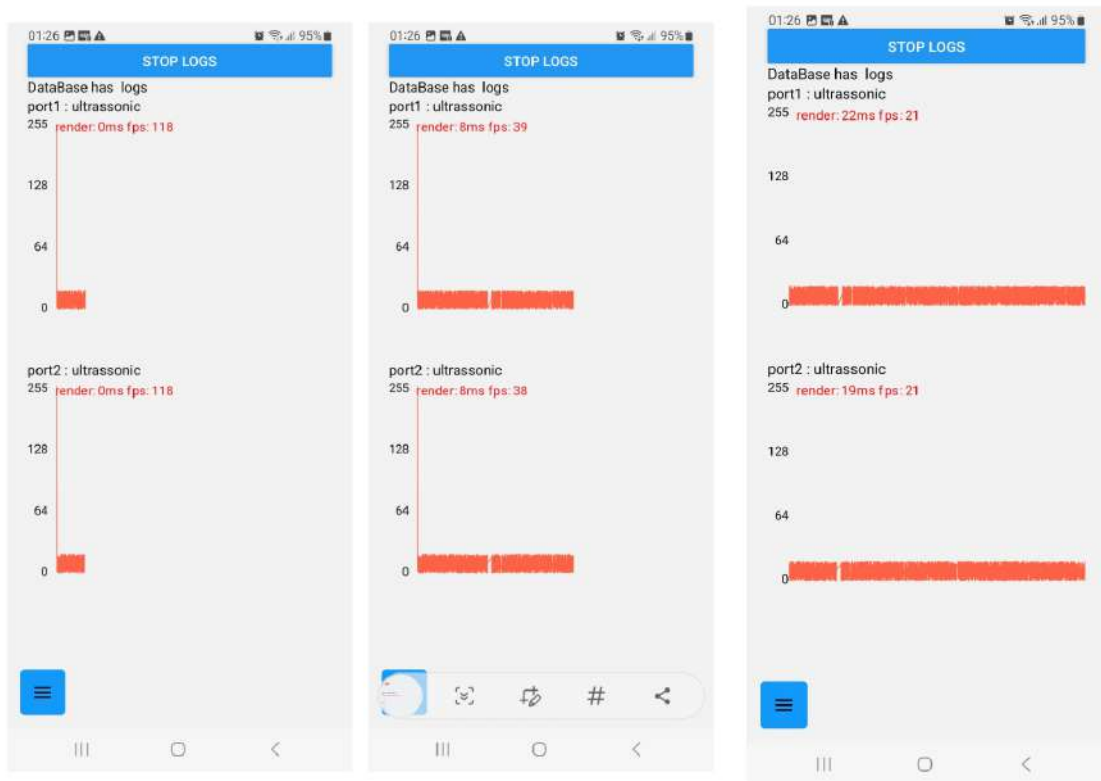
Para converter o “time” do *log* para uma coordenada x relativa do gráfico utilizamos a fórmula $x = (tl(n) - tl(n - 1)) \times \frac{cxc}{10000}$, em que $tl(n)$ é o “time” do *log* atual, $tl(n - 1)$ é o “time” do *log* anterior, *cxc* é o comprimento do eixo x do *canvas* em *pixels*, e 10000 é referentes a *timeline* de até 10.000 *logs* renderizados referentes aos últimos 10 segundos de leitura do sensor.

O gráfico funciona renderizando os *logs* da esquerda para direita como mostra na Figura 18.

Onde a extremidade direita vai representar os *logs* capturados no momento atual, e a extremidade esquerda os *logs* lidos até 10 segundos passados, dessa forma é possível visualizar a leitura do sensor em tempo real e também até os últimos 10 segundos.

Perceba que para exibir a *timeline* dos últimos 10 segundos continuamente, assim que o gráfico preenche o eixo x, realizamos uma rolagem contínua, em que constantemente o desenho é deslocado para esquerda, onde os *logs* mais antigos da esquerda são descartados do gráfico e os *logs* mais recentes são incluídos no novo espaço disponível na extremidade direita do *canvas*.

Figura 18 – Renderização do Gráfico



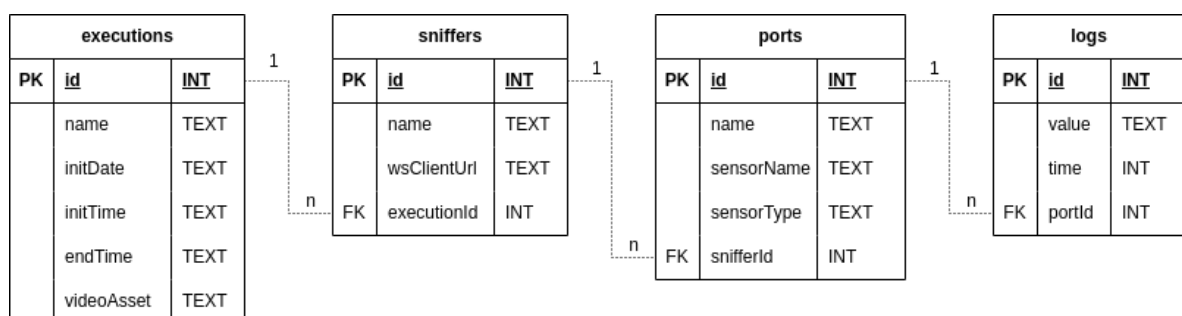
Fonte: Elaborado pelo autor, 2023

4.2.4 Banco de Dados

Os *logs* dos sensores são armazenados de forma permanente para posteriores processos de depuração. Optamos pela biblioteca SQLite, compatível com o ecossistema Expo, para o armazenamento dos *logs* dos sensores. Esta biblioteca facilita a criação de um banco de dados relacional embutido no aplicativo, oferecendo recursos como operações assíncronas, inicialização rápida e suporte a transações atômicas, além de operações SQL padrão.

O banco de dados segue o modelo relacional da Figura 19. Onde todos os ids são

Figura 19 – Modelo Relacional



Fonte: Elaborado pelo autor, 2023

do tipo “INT” para armazenar números inteiros, sendo todos esses identificadores chaves


```
4  "port2" : { "id": 3,  
5           "logs" : [ { "value" : 300, "time" : 1000 },  
6                   { "value" : 400, "time" : 2000 } ] } }
```

O *buffer* é consumido periodicamente a cada 10 segundos de forma que todos os *logs* são inseridos no banco através de uma operação de *insert* em lote e assíncrona, esse período além de ser referente a uma *timeline* completa do gráfico, também foi testado para diminuir ao máximo a quantidade de operações no banco, ainda garantindo que o *buffer* em memória não cresça demais e que a operação em lote não se torne demorada.

4.2.5 Gravação de vídeo

Durante a operação do robô, a gravação de vídeo é uma funcionalidade crucial, implementada com a ajuda da biblioteca expo-camera. Esta biblioteca fornece acesso à câmera do dispositivo móvel, permitindo não só a captura de vídeo, mas também a visualização em tempo real da gravação como mostra na imagem 20.

A gravação é disparada pela função assíncrona “recordAsync” que retorna um objeto “Promise”, quando a gravação termina o método “then” desse objeto é chamado recebendo o objeto *asset* do vídeo gravado temporariamente no *cache* do aplicativo, desse *asset* extraímos o atributo “uri” que é a referência do vídeo em *cache*.

Como o *asset* está apenas em *cache*, utilizamos a biblioteca expo-media-library para armazenar o vídeo de forma permanente na memória do celular. Com o “uri” do vídeo em *cache* podemos criar um novo *asset* permanente com o método “createAssetAsync” como mostra no código 4.13,

Código 4.13 – Armazenando vídeo

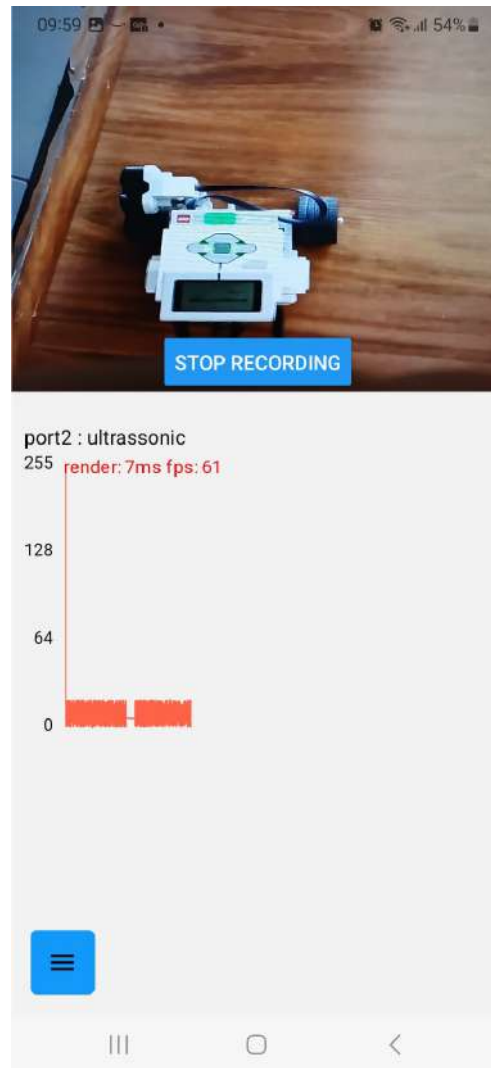
```
1 import * as MediaLibrary from "expo-media-library";  
2 const asset = await MediaLibrary.createAssetAsync(uri);
```

em que o “asset” gerado é um objeto *json* de formato descrito no código 4.14,

Código 4.14 – Asset do vídeo

```
1 { "mediaType": "video",  
2   "modificationTime": 1686517909000,  
3   "uri": "file:///storage/emulated/0/DCIM/1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4",  
4   "filename": "1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4",  
5   "width": 1080,  
6   "id": "1000010523",  
7   "creationTime": 1686517904000,  
8   "albumId": "-2075821635",  
9   "height": 1920,  
10  "duration": 7.783 }
```

Figura 20 – Gravação de vídeo



Fonte: Elaborado pelo autor, 2023

este *asset* é salvo no atributo “videoAsset” do registro de execução 4.11, para que posteriormente possamos recuperar o vídeo salvo através desse objeto referência.

4.2.6 Depuração da execução

As execuções armazenadas no banco de dados são recuperadas com as referências aos *sniffers* e portas dos sensores em objetos *json* como no código 4.15, onde de acordo com as referências de chave estrangeira do modelo relacional 19, a partir do id da execução carregamos todos os *sniffers*, e para cada id dos *sniffers* buscamos as portas dos sensores.

Código 4.15 – Objeto json da execução

```
1 { "id": 1,  
2   "name": "Queda do robo",  
3   "initDate": "2023-04-10",  
4   "initTime": "17:59:33:793",  
5   "endTime": "18:0:6:127",
```

```

6  "videoAsset": { "mediaType": "video",
7                  "modificationTime": 1686517909000,
8                  "uri": "file:///storage/emulated/0/DCIM/1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.
    mp4",
9                  "filename": "1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4",
10                 "width": 1080,
11                 "id": "1000010523",
12                 "creationTime": 1686517904000,
13                 "albumId": "-2075821635",
14                 "height": 1920,
15                 "duration": 7.783 },
16  "sniffers": [ { "id": 1,
17                "name": "sensores de distancia",
18                "wsClientUrl": "ws://192.168.1.199:81",
19                "ports": [ { "id": 1,
20                            "name": "port1",
21                            "sensorName": "sensor de distancia",
22                            "sensorType": "ultrasonic" } ] } ] }

```

Para reproduzir as execuções, foi utilizado a biblioteca expo-av que disponibiliza um *player* de vídeo através do componente “Video” como demonstrado no código 4.16, onde configuramos a propriedade “videoProps.source.uri” com o atributo “videoAsset.uri” da execução 4.15, o componente precisa dessa propriedade para poder carregar e reproduzir o vídeo.

Código 4.16 – Componente de vídeo

```

1  < Video
2    videoProps = { source: { uri: "file:///storage/emulated/0/DCIM/1e37dd68-3a55-462e-9a66-7
    d2c7dcc77d2.mp4" } }
3    onPlaybackStatusUpdate = { updatePlaybackCallback } />

```

Nesse componente temos a propriedade “onPlaybackStatusUpdate” que é configurada como a função de *callback* “updatePlaybackCallback” disparada todas as vezes que o status do *player* é alterado, isso acontece nos cenários em que o vídeo carrega, o usuário clica no vídeo, avança ou retrocede no vídeo e também é disparada continuamente enquanto o vídeo é reproduzido.

Nessa função recebemos um objeto com o atributo “positionMillis” que indica o momento atual em que o vídeo está sendo reproduzido em milissegundos, dessa forma é realizada a busca dos *logs* em uma *timeline* que começa 7 segundos antes e 7 segundos depois do momento “positionMillis”, esses *logs* são injetados no *buffer* de *logs* do vídeo que tem a estrutura demonstrada no código 4.17. Periodicamente esse *buffer* vai sendo preenchido conforme o “positionMillis” vai se aproximando do valor “time” do último *log* do *buffer*.

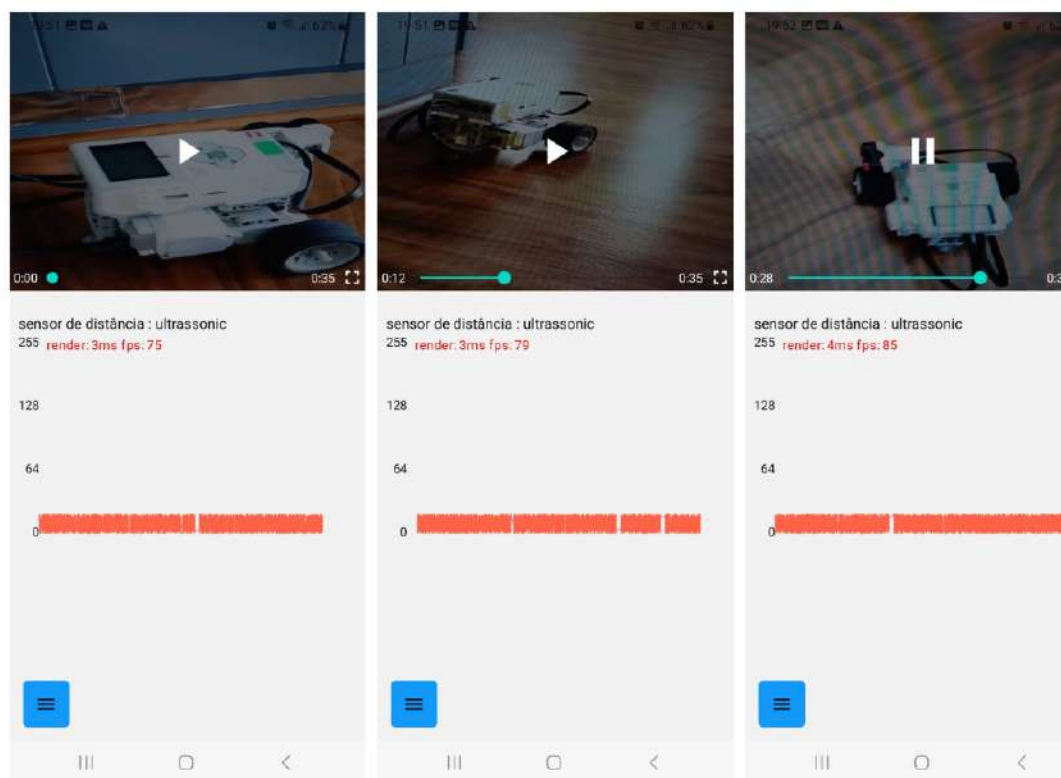
Código 4.17 – Buffer de logs do vídeo

```
1 { "port1" : { "id": 2,  
2     "logs" : [ { "value" : 100, "time" : 1000 },  
3     { "value" : 200, "time" : 2000 } ] },  
4 "port2" : { "id": 3,  
5     "logs" : [ { "value" : 300, "time" : 1000 },  
6     { "value" : 400, "time" : 2000 } ] } }
```

Durante a reprodução do vídeo a função “updatePlaybackCallback” vai sendo disparada e utilizamos a diferença entre o “positionMillis” atual com seu valor da chamada anterior, com isso obtemos quantos milissegundos se passaram desde a última renderização dos gráficos dos sensores, assim o algoritmo consome essa diferença de *timeline* dos *logs* do *buffer* 4.17 e realiza a renderização contínua dos gráficos dos sensores em forma de rolagem contínua como descrita anteriormente para o *buffer* de *logs* dos gráficos 4.9.

Assim o usuário pode reproduzir novamente a qualquer momento as execuções do robô e depurar o *bug* filmado durante a execução, podendo até selecionar um momento específico da gravação pela barra de reprodução do vídeo como mostra na Figura 21.

Figura 21 – Depuração da execução



Fonte: Elaborado pelo autor, 2023

5 Considerações Finais

Neste trabalho, apresentamos o desenvolvimento integrado de software e hardware projetados especificamente para monitorar o sensor de ultrassom do Kit de Desenvolvimento de Robótica LEGO MINDSTORMS EV3. O projeto abarcou desde a concepção do design eletrônico até a construção da placa de circuito impresso, seu *case*, e o desenvolvimento tanto do *firmware* quanto do aplicativo para dispositivos móveis Android. O objetivo central foi criar um sistema eficiente que pudesse realizar o *debug* do sensor sem sobrecarregar o processador do *BRICK* da LEGO, capturando dados do sensor e imagens de vídeo em tempo real.

O propósito deste projeto foi desenvolver um componente físico que, quando integrado ao sistema LEGO, oferecesse uma solução eficaz para debugar o hardware do sensor. Ao separar as medições e o processamento de vídeo do *BRICK* da LEGO, nosso sistema permite que desenvolvedores de robótica distingam claramente entre falhas no código do robô e erros decorrentes de leituras imprecisas do sensor.

Devido a capacidade do sistema atuar diretamente entre a conexão do sensor com o cérebro do robô e enviar os dados via rede local para um outro hardware, nesse caso um dispositivo móvel com sistema operacional Android, a capacidade de acompanhamento e *debug* foi então estabelecida diante da possibilidade de gravação dos dados do sensor, juntamente com a imagem da câmera do dispositivo móvel, que podem ser resgatadas posteriormente para análise de dados e comportamento do robô.

Apesar de a LEGO ter anunciado em 2022 a descontinuidade da linha MINDSTORMS, conforme reportado por (Cais Cris, 2018), é importante notar que muitas instituições educacionais ainda possuem estes kits. Devido ao investimento significativo realizado na aquisição desses kits, eles continuarão sendo uma ferramenta valiosa para o ensino e desenvolvimento de robótica, assegurando a relevância contínua do nosso projeto.

5.1 Sugestões para Trabalhos Futuros

Como sugestões de trabalhos futuros:

- Implementar a leitura de vários sensores simultaneamente;
- Expandir o sistema de monitoramento para outras plataformas, IOS, Windows, Linux e Mac OS.
- Inserir um sistema de baterias recarregáveis, a fim de não utilizar a eletricidade proveniente do *Brick*.

Referências

- ANN, T. S. **About Espressif**. 2022. Disponível em: <https://www.espressif.com/en/company/about-us/ceo-letter>. Acesso em: 12 Ago 2022. Citado na página 22.
- AQUILES, A.; FERREIRA, R. **Controlando versões com Git e Github**. [S.l.]: Casa do Código, 2014. ISBN 9788566250534. Citado na página 26.
- ARAUJO, W. M.; CAVALCANTE, M. M.; SILVA, R. O. da. Visão geral sobre microcontroladores e prototipagem com arduino. **Revista Tecnologias em Projeção**, v. 10, n. 1, p. 36–46, 2019. Citado na página 21.
- BELL, P.; BEER, B. **Introducing GitHub: A Non-Technical Guide**. [S.l.]: O'Reilly Media, 2014. ISBN 9781491949832. Citado 2 vezes nas páginas 25 e 26.
- Cais Cris. **Brick Fanatics**. 2018. Disponível em: <https://www.brickfanatics.com/lego-discontinuing-mindstorms-end-of-2022/>. Acesso em: 30 Nov 2022. Citado na página 62.
- CAO, J. **What Is a Prototype: A Guide to Functional UX**. 2022. Disponível em: <https://www.uxpin.com/studio/blog/what-is-a-prototype-a-guide-to-functional-ux/>. Acesso em: 10 Ago 2022. Citado na página 23.
- CARSTENS, S. F.; CARSTENS, T. A. **Projeto e fabricação de uma fresadora CNC para prototipagem de placas de circuito impresso**. 2015. Trabalho de Conclusão de Curso - Instituto Federal de Santa Catarina. Citado 2 vezes nas páginas 23 e 24.
- DIAS, A. B. **ENGENHARIA REVERSA: uma porta ainda aberta**. 1997. Disponível em: https://www.researchgate.net/publication/335456099_ENGENHARIA_REVERSA_uma_porta_ainda_aberta. Acesso em: 08 Ago 2022. Citado na página 25.
- ev3dev. **ev3dev Home**. 2016. Disponível em: <https://www.ev3dev.org/>. Acesso em: 4 Ago 2022. Citado na página 21.
- ev3dev. **ev3dev Programming Languages**. 2016. Disponível em: <https://www.ev3dev.org/docs/programming-languages/>. Acesso em: 4 Ago 2022. Citado na página 21.
- Fernando Veiga. **Robótica – Movimento Mindstorms**. 2015. Disponível em: <https://imasters.com.br/tecnologia/robotica-movimento-mindstorms>. Acesso em: 2 Ago 2022. Citado na página 20.
- GITHUB. **Where the world builds software**. 2022. Disponível em: <https://github.com/about>. Acesso em: 10 Ago 2022. Citado na página 26.
- KOYANAGI, F. **Introdução ao ESP32**. 2018. Disponível em: <https://www.fernandok.com/2017/11/introducao-ao-esp32.html>. Acesso em: 12 Ago 2022. Citado na página 22.
- LEGO. **LEGO MINDSTORMS EV3 User Guide**. 2013. Disponível em: <https://education.lego.com/en-gb/product-resources/mindstorms-ev3/downloads/user-guide>. Acesso em: 2 Ago 2022. Citado 2 vezes nas páginas 19 e 21.

- LU, W. **Beginning Robotics Programming in Java with LEGO Mindstorms**. [S.l.]: Springer, 2016. Citado na página 21.
- MIRANDA, L. A. V. **Monitoramento de parâmetros ambientais de um leito hospitalar utilizando ESP32**. 2019. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas. Citado na página 22.
- OLIVEIRA, M. F. de. **Aplicações da Prototipagem Rápida em Projetos de Pesquisa**. 2008. Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica. Citado na página 23.
- PERCHÉ, C. F. de P. **Desenvolvimento de um sistema de prototipagem para placas de circuitos impressos**. 2013. Trabalho de Conclusão de Curso - Universidade Federal de Itajubá - Campus Itabira. Citado na página 24.
- RATHORE N. & JAIN, P. K. **DAAAM International Scientific Book 2014, pag.567-576, cap.45**. [S.l.]: DAAAM International, 2014. ISBN 9783901509988. Citado na página 25.
- SOUZA, I. M. et al. A systematic review on the use of lego® robotics in education. In: IEEE. **2018 IEEE Frontiers in Education Conference (FIE)**. [S.l.], 2018. p. 1–9. Citado na página 19.
- WANG, W. **Reverse Engineering. Technology of Reinvention**. [S.l.]: Taylor and Francis Group, LLC, 2011. ISBN 9781439806319. Citado 2 vezes nas páginas 24 e 25.
- WILDT, D. et al. **eXtreme Programming: Práticas para o dia a dia no desenvolvimento ágil de software**. [S.l.]: Casa do Código, 2015. ISBN 9788555191077. Citado na página 27.

Apêndices

APÊNDICE A – Repositório Firmware

A.1 .gitignore do firmware

```
1 .pio
2 .vscode/.browse.c_cpp.db*
3 .vscode/c_cpp_properties.json
4 .vscode/launch.json
5 .vscode/ipch
6 personal_configs
```

A.2 Arquivo configuração PlatformIO

```
1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:esp-wrover-kit]
12 platform = espressif32
13 board = esp-wrover-kit
14 framework = arduino
15 monitor_speed = 115200
16 upload_speed = 921600
17 upload_port = /dev/ttyUSB0
18 monitor_port = /dev/ttyUSB0
19 lib_deps = links2004/WebSockets@~2.3.7
```

A.3 Arquivo de configuração da rede

```
1 #define ROUTER_NAME "NOME DA REDE WIFI"
2 #define ROUTER_PASS "SENHA DA REDE"
```

A.4 Arquivo código do firmware

```
1  #include <Arduino.h>
2  #include <WiFi.h>
3  #include <WebSocketsServer.h>
4  #include <../personal_configs/wifi_router.cpp>
5
6  #define RX_PIN 26 //Pino 26 RX DO SENSOR!
7  #define TX_PIN 25 //Pino 25 TX DO SENSOR!
8
9  String bufferS = "";
10 int16_t numero = 0;
11 unsigned long timer, timerNow = 0;
12 int valorDoSensor = 0;
13
14 boolean send_log = false;
15 WebSocketsServer websocket = WebSocketsServer(81);
16 int counter = 0;
17
18 void configSerialMonitor(int num = 9600)
19 {
20   Serial.begin(num);
21   delay(1000);
22   Serial.println("Monitor working");
23 }
24
25 void connectToRouter(const char *ssid, const char *password)
26 {
27   Serial.print("Connecting to router");
28
29   IPAddress local_IP(192, 168, 0, 199);
30   IPAddress gateway(192, 168, 0, 1);
31   IPAddress subnet(255, 255, 0, 0);
32
33   WiFi.begin(ssid, password);
34   while (WiFi.status() != WL_CONNECTED)
35   {
36     delay(500);
37     Serial.print(" .");
38   }
39   Serial.println("\nConected");
40   Serial.print("IP: ");
41   Serial.println(WiFi.localIP());
42 }
43
44 void receiveMsg(uint8_t num, WStype_t type, uint8_t *payload, size_t length)
45 {
46   switch (type)
47   {
48     case WStype_PING:
49       Serial.println("Client " + String(num) + " pinged");
50       break;
51     case WStype_PONG:
52       Serial.println("Client " + String(num) + " ponged");
53       break;
54     case WStype_DISCONNECTED: // if a client is disconnected, then type == WStype_DISCONNECTED
55       Serial.println("Client " + String(num) + " disconnected");
56       break;
57     case WStype_CONNECTED: // if a client is connected, then type == WStype_CONNECTED
```

```

58     Serial.println("Client " + String(num) + " connected");
59     Serial.println("Server has " + String(webSocket.connectedClients()) + " clients connected");
60     Serial.println("Server has " + webSocket.remoteIP(num).toString() + " clients connected");
61
62     break;
63 case WStype_TEXT: // if a client has sent data, then type == WStype_TEXT
64     char payloadString[strlen((char *) (payload))];
65
66     strcpy(payloadString, (char *) (payload));
67     if (strcmp(payloadString, "start logs") == 0)
68         send_log = true;
69     else if (strcmp(payloadString, "stop logs") == 0)
70         send_log = false;
71     // connection confirmation
72     else if (strcmp(payloadString, "ping") == 0)
73         webSocket.sendTXT(num, "pong");
74     // get connected ports
75     else if (strcmp(payloadString, "ports") == 0)
76     {
77         // manda as portas que tem sensor conectado
78         webSocket.sendTXT(num, "{\"connectedPorts\": [\"port1\", \"port2\"]}"); // caso de um sensor de toque
79         ↵ e outro de ultrasônico
80     }
81     else
82     {
83     }
84
85     Serial.printf("received: payload [%u]: %s\n", num, payloadString);
86
87     break;
88 };
89 }
90
91 void setup() {
92     configSerialMonitor(115200);
93     connectToRouter(ROUTER_NAME, ROUTER_PASS);
94     delay(1000);
95
96     pinMode(RX_PIN, INPUT);
97     pinMode(TX_PIN, INPUT);
98     delay(1000);
99
100    webSocket.begin();
101    webSocket.onEvent(receiveMsg);
102    Serial.print("Configurado");
103 }
104
105 int estadoAltoRepouso(){
106     /*
107     Leitura do estado de repouso, antes do envio de dados.
108     Nesse estado o sinal está em 1 constantemente até cair para 0 quando se dá o início de envio.
109     Caso leia uma sequência de 10 bits 1, significa que quando cair para 0, de fato é dados e não ruído.
110     */
111     ets_delay_us(5);
112     int count = 0;
113     while(1){
114         if (digitalRead(TX_PIN)){
115             count ++;
116         } else {
117             count = 0;

```

```
118     }
119     if (count >= 10){
120         return 1;
121     }
122     ets_delay_us(14);
123 }
124 return 0;
125 }
126
127 void lendoDados(){
128     if (estadoAltoRepouso()){
129         while(1){
130             if(!digitalRead(TX_PIN)){
131                 ets_delay_us(5);
132                 int i = 0;
133                 int k = 0;
134                 bufferS = "";
135                 numero = 0;
136
137                 while(i < 40){
138                     int j = 0;
139                     int bit = 0;
140                     while(j <= 9){
141                         if (digitalRead(TX_PIN) == 1) {
142                             bit++;
143                         } else {
144                             bit--;
145                         };
146                         j++;
147                     }
148                     if (bit > 0){
149                         bufferS += '1';
150                     }
151                     else {
152                         bufferS += '0';
153                     }
154                     i++;
155                     ets_delay_us(14);
156                 }
157
158                 int incremento = 0;
159                 for (int i = 11; i < 19; i++) {
160                     if (bufferS[i] == '1'){
161                         numero |= 1 << incremento;
162                     }
163                     incremento++;
164                 }
165                 for (int i = 21; i < 29; i++) {
166                     if (bufferS[i] == '1'){
167                         numero |= 1 << incremento;
168                     }
169                     incremento++;
170                 }
171                 if(numero >= 0){
172                     valorDoSensor = numero;
173                 }
174                 break;
175             }
176         }
177     }
178     return;
```

```
179 }
180
181 int logTimerSimulator = 0;
182 void enviandoDados(){
183     websocket.loop();
184     int bufferDelay = 20;
185     if (send_log && websocket.connectedClients()
186     {
187         String logs = "{\"logs\":{\"port1\":[\";
188         for (int i = 0; i < bufferDelay; i++) {
189             if (i == bufferDelay - 1) logs.concat("{\"value\":" + String(random(10)) + "\",\"time\":" +
190             ↵ String(logTimerSimulator) + "}");
191             else logs.concat("{\"value\":" + String(random(10)) + "\",\"time\":" + String(logTimerSimulator) +
192             ↵ "},");
193             logTimerSimulator++;
194         }
195         logTimerSimulator -= bufferDelay;
196         logs.concat("\",\"port2\":[\";
197         for (int i = 0; i < bufferDelay; i++) {
198             if (i == bufferDelay - 1) logs.concat("{\"value\":" + String(random(10)) + "\",\"time\":" +
199             ↵ String(logTimerSimulator) + "}");
200             else logs.concat("{\"value\":" + String(random(10)) + "\",\"time\":" + String(logTimerSimulator) +
201             ↵ "},");
202             logTimerSimulator++;
203         }
204         logs.concat("]}}");
205         websocket.broadcastTXT(logs);
206         counter++;
207         delay(bufferDelay);
208     } else {
209         logTimerSimulator = 0;
210     }
211 }
212
213 void loop() {
214     lendoDados();
215     enviandoDados();
216     Serial.println(valorDoSensor);
217 }
```

APÊNDICE B – Repositório Aplicativo Móvel

B.1 Arquivo de configuração do Gradle

```

1 rootProject.name = 'telemetrymobileapp'
2
3 apply from: new File(["node", "--print", "require.resolve('expo/package.json')"].execute(null,
  ↪ rootDir).text.trim(), "../scripts/autolinking.gradle");
4 useExpoModules()
5
6 apply from: new File(["node", "--print",
  ↪ "require.resolve('@react-native-community/cli-platform-android/package.json')"].execute(null,
  ↪ rootDir).text.trim(), "../native_modules.gradle");
7 applyNativeModulesSettingsGradle(settings)
8
9 include ':app'
10 includeBuild(new File(["node", "--print",
  ↪ "require.resolve('react-native-gradle-plugin/package.json')"].execute(null,
  ↪ rootDir).text.trim()).getParentFile())
11
12 if (settings.hasProperty("newArchEnabled") && settings.newArchEnabled == "true") {
13   include(":ReactAndroid")
14   project(":ReactAndroid").projectDir = new File(["node", "--print",
  ↪ "require.resolve('react-native/package.json')"].execute(null, rootDir).text.trim(),
  ↪ "../ReactAndroid");
15   include(":ReactAndroid:hermes-engine")
16   project(":ReactAndroid:hermes-engine").projectDir = new File(["node", "--print",
  ↪ "require.resolve('react-native/package.json')"].execute(null, rootDir).text.trim(),
  ↪ "../ReactAndroid/hermes-engine");
17 }

```

B.2 Arquivo de configuração do App

```

1 {
2   "name": "telemetry-mobile-app",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "expo-deploy": "expo build:android",
7     "emulator-server": "npx react-native start --reset-cache",
8     "emulator": "npx react-native run-android",
9     "dev": "expo start",
10    "clean-dev": "expo start --clear",
11    "prod": "npx expo start --no-dev --minify",

```



```
12   "start": "expo start --dev-client",
13   "android": "expo run:android",
14   "ios": "expo run:ios",
15   "web": "expo start --web"
16 },
17 "dependencies": {
18   "@react-native-community/slider": "^4.4.2",
19   "@react-navigation/drawer": "^6.5.0",
20   "@react-navigation/native": "^6.0.13",
21   "@shopify/react-native-skia": "0.1.172",
22   "@types/react": "~18.0.27",
23   "expo": "^48.0.15",
24   "expo-av": "^13.2.1",
25   "expo-camera": "^13.2.1",
26   "expo-media-library": "^15.2.3",
27   "expo-splash-screen": "~0.18.2",
28   "expo-sqlite": "~11.1.1",
29   "expo-status-bar": "~1.4.4",
30   "expo-updates": "^0.16.4",
31   "expo-video-thumbnails": "~7.2.1",
32   "mobx": "^6.6.2",
33   "mobx-react": "^7.5.3",
34   "react": "18.2.0",
35   "react-dom": "18.2.0",
36   "react-native": "0.71.8",
37   "react-native-dropdown-picker": "^5.4.2",
38   "react-native-gesture-handler": "~2.9.0",
39   "react-native-reanimated": "~2.14.4",
40   "react-native-safe-area-context": "4.5.0",
41   "react-native-screens": "~3.20.0",
42   "react-native-web": "~0.18.10",
43   "typescript": "^4.9.4"
44 },
45 "devDependencies": {
46   "@babel/core": "^7.12.9"
47 },
48 "private": true
49 }
```

B.3 Arquivo de configuração do Metro

```
1 // Learn more https://docs.expo.io/guides/customizing-metro
2 const { getDefaultConfig } = require('expo/metro-config');
3
4 module.exports = getDefaultConfig(__dirname);
```

B.4 Arquivo index do App

```
1 import { registerRootComponent } from 'expo';
2
3 import App from './App';
4
5 // registerRootComponent calls AppRegistry.registerComponent('main', () => App);
6 // It also ensures that whether you load the app in Expo Go or in a native build,
7 // the environment is set up appropriately
8 registerRootComponent(App);
```

B.5 Arquivo de configuração do Babel

```
1 module.exports = function(api) {
2   api.cache(true);
3   return {
4     presets: ['babel-preset-expo'],
5     plugins: [
6       'react-native-reanimated/plugin', // swipe animation and skia dependency
7     ]
8   };
9 };
```

B.6 Arquivo de estilo do App

```
1 import { StyleSheet } from "react-native";
2
3 export const styles = StyleSheet.create({
4   view: {
5     flex: 1,
6     backgroundColor: '#fff',
7     alignItems: 'center',
8     justifyContent: 'center',
9   },
10 });
```

B.7 Arquivo de configuração Android do App

```
1 {
2   "expo": {
```

```
3     "name": "telemetry-mobile-app",
4     "slug": "telemetry-mobile-app",
5     "version": "1.0.0",
6     "assetBundlePatterns": [
7         "**/*"
8     ],
9     "android": {
10        "package": "com.antonio357.telemetrymobileapp"
11    }
12 }
13 }
```

B.8 .gitignore do App

```
1 # OSX
2 #
3 .DS_Store
4
5 # Xcode
6 #
7 build/
8 *.pbxuser
9 !default.pbxuser
10 *.mode1v3
11 !default.mode1v3
12 *.mode2v3
13 !default.mode2v3
14 *.perspectivev3
15 !default.perspectivev3
16 xcuserdata
17 *.xccheckout
18 *.moved-aside
19 DerivedData
20 *.hmap
21 *.ipa
22 *.xcuserstate
23 project.xcworkspace
24
25 # Android/IntelliJ
26 #
27 build/
28 .idea
29 .gradle
30 local.properties
31 *.iml
32 *.hprof
33
34 # node.js
35 #
36 node_modules/
37 npm-debug.log
38 yarn-error.log
39
40 # BUCK
41 buck-out/
```

```
42 \.buckd/
43 *.keystore
44 !debug.keystore
45
46 # Bundle artifacts
47 *.jsbundle
48
49 # CocoaPods
50 /ios/Pods/
51
52 # Expo
53 .expo/
54 web-build/
55 dist/
```

B.9 Objeto de definição das dimensões dos canvas

```
1 import {Dimensions} from 'react-native';
2
3
4 export class CanvasDimensions {
5   // card para o canvas
6   screen = { height: Dimensions.get('window').height, width: Dimensions.get('window').width };
7   cardHorizontalMargin = 48;
8   card = { height: 250, width: this.screen.width - this.cardHorizontalMargin };
9   cardText = { height: 24, width: this.card.width - 20, fontSize: 12 };
10  canvasHorizontalMargin = 20;
11
12  // canvas
13  canvasDimensions = { height: this.card.height - this.cardText.height, width: this.card.width -
14  ↪ this.canvasHorizontalMargin };
15  xyOpointDimensions = { height: 10, width: 10 };
16  yAxisDimensions = { height: this.canvasDimensions.height - this.xyOpointDimensions.height, width:
17  ↪ this.xyOpointDimensions.width };
18  xAxisDimensions = { height: this.xyOpointDimensions.height, width: this.canvasDimensions.width };
19  lineDrawDimensios = { height: this.canvasDimensions - this.xyOpointDimensions.height, width:
20  ↪ this.canvasDimensions - this.xyOpointDimensions.width };
21
22  // drawpoints limits
23  lineDrawPoints = {
24    leftBottom: { x: this.xyOpointDimensions.width, y: this.xyOpointDimensions.height },
25    leftTop: { x: this.xyOpointDimensions.width, y: this.canvasDimensions.height },
26    rightTop: { x: this.canvasDimensions.width, y: this.canvasDimensions.height },
27    rightBottom: { x: this.canvasDimensions.width, y: this.xyOpointDimensions.height }
28  }
29  yAxisDrawPoints = {
30    leftBottom: { x: 0, y: this.xyOpointDimensions.height },
31    leftTop: { x: 0, y: this.canvasDimensions.height },
32    rightTop: { x: this.xyOpointDimensions.width, y: this.canvasDimensions.height },
33    rightBottom: { x: this.xyOpointDimensions.width, y: this.xyOpointDimensions.height }
34  }
35  xAxisDrawPoints = {
36    leftBottom: { x: this.xyOpointDimensions.width, y: 0 },
37    leftTop: { x: this.xyOpointDimensions.width, y: this.xyOpointDimensions.height },
38    rightTop: { x: this.canvasDimensions.width, y: this.xyOpointDimensions.height },
```

```

36     rightBottom: { x: this.canvasDimensions.width, y: 0 }
37   }
38 }

```

B.10 Arquivo da classe dos gráficos

```

1  import { StyleSheet, View, Text } from "react-native";
2  import { Canvas, Path } from "@shopify/react-native-skia";
3  import { CanvasDimensions } from './CanvasDimensions';
4
5
6  const dimensions = new CanvasDimensions();
7
8  const styles = StyleSheet.create({
9    canvas: { height: dimensions.canvasDimensions.height, width: dimensions.canvasDimensions.width },
10   yAxisDrawPath: {},
11   drawPath: {},
12   xAxisDrawPath: {},
13   yAxisCanvas: { flexDirection: 'row' },
14   yAxisText: { flex: 1, flexDirection: 'column', height: 68, fontSize: 12, textAlign: "right" }
15 });
16
17 function YAxisLabel() {
18   return (
19     <View style={styles.yAxis}>
20       <Text key={1} style={styles.yAxisText}>255</Text>
21       <Text key={2} style={styles.yAxisText}>128</Text>
22       <Text key={3} style={styles.yAxisText}>64</Text>
23       <Text key={4} style={styles.yAxisText}>0</Text>
24     </View>
25   )
26 }
27
28 export function ChartCard({ sensorConfig }) {
29   // const sensorConfig = {
30   //   sensorName: "nome do sensor",
31   //   sensorType: "ultrassonic",
32   //   timeFrame: 10, // faixa de tempo do eixo x em segundos
33   //   logsRate: 1000, // quantidade dos logs por segundo
34   //   drawPath: Skia.Path.Make()
35   // }
36
37   return (
38     <View>
39       <Text style={styles.chartName} >{sensorConfig.sensorName} : {sensorConfig.sensorType}</Text>
40       <View style={styles.yAxisCanvas}>
41         <YAxisLabel />
42         <Canvas style={styles.canvas} mode='continuous' debug={true}>
43           /* path pro eixo y */
44           <Path path={sensorConfig.drawPath} style="stroke" color="tomato" strokeWidth={1} />
45           /* path pro eixo x */
46         </Canvas>
47       </View>
48     </View>

```

```
49 );  
50 }
```

B.11 Arquivo da classe de listagem dos gráficos

```
1 import { ChartCard } from './ChartCard';  
2 import { StyleSheet } from "react-native";  
3  
4  
5 const styles = StyleSheet.create({  
6   // scrollView: {flex: 1},  
7 });  
8  
9  
10 export function ChartCardsList({ sensorConfigsArray }) {  
11   // const array = [  
12     // {  
13     //   sensorName: "nome do sensor",  
14     //   sensorType: "ultrassonic",  
15     //   timeFrame: 10, // faixa de tempo do eixo x em segundos  
16     //   logsRate: 1000, // quantidade dos logs por segundo  
17     //   drawPath: Skia.Path.Make()  
18     // },  
19     // ...  
20   // ];  
21  
22   return (  
23     <>  
24     {sensorConfigsArray.map(sensorConfig => <ChartCard key={Math.random()} sensorConfig={sensorConfig}  
25     => />)}  
26     </>  
27   );  
}
```

B.12 Arquivo da classe de controle do Objeto Path

```
1 import { Skia } from "@shopify/react-native-skia";  
2 import { CanvasDimensions } from './CanvasDimensions';  
3  
4  
5 export class ChartDrawPath {  
6   constructor(sensorType, timeFrame = 10, logsRate = 1000) {  
7     this.path = Skia.Path.Make();  
8     this.path.setIsVolatile(false);  
9  
10    this.xBounds = { min: 0, max: timeFrame * logsRate };  
11    this.yBounds = { min: 0 };  
12    if (sensorType == 'ultrassonic') {
```

```

13     this.yBounds['max'] = 2550;
14   }
15   // this.canvasDimensions = new CanvasDimensions();
16   this.lineDrawPoints = new CanvasDimensions().lineDrawPoints;
17   this.axisLength = { x: this.lineDrawPoints.rightBottom.x - this.lineDrawPoints.leftBottom.x, y:
18     ↪ this.lineDrawPoints.rightTop.y - this.lineDrawPoints.rightBottom.y };
19   this.dimensionsUnits = { x: this.axisLength.x / this.xBounds.max, y: this.axisLength.y /
20     ↪ this.yBounds.max };
21   this.lastPointTime = 0;
22   console.log(`${this.axisLength.x} == ${this.dimensionsUnits.x} * ${this.xBounds.max} ==
23     ↪ ${this.dimensionsUnits.x * this.xBounds.max}`);
24   this.firstPoint = true;
25 }
26
27 pushData = data => {
28   // console.log(`pushData data = ${JSON.stringify(data)}`);
29   // console.log(`na tela de recording tem = ${this.path.countPoints()}`);
30   // const data = {
31   //   value: '', // string com o valor do sensor
32   //   time: 15000 // inteiro com o valor de tempo em ms que se passou após o início da execução
33   // };
34   const y = ((this.yBounds['max'] - parseInt(data.value)) * this.dimensionsUnits.y);
35   const timeDiff = (data.time - this.lastPointTime) * this.dimensionsUnits.x;
36   const x = (this.path.getLastPt().x + timeDiff);
37   this.path.lineTo(x, y);
38   if (this.path.getLastPt().x > this.lineDrawPoints.rightBottom.x) {
39     const pathLen = Math.abs(this.path.getPoint(0).x - this.path.getLastPt().x);
40     const trim = (pathLen - this.axisLength.x) / pathLen;
41     if (trim >= 0.25) {
42       this.path.trim(trim, 1, false);
43     }
44     const offSet = this.path.getLastPt().x - this.lineDrawPoints.rightBottom.x;
45     this.path.offset(- offSet, 0);
46   }
47   this.lastPointTime = data.time;
48 }
49
50 loadDataVector = vector => {
51   if (this.firstPoint && vector.length > 0) {
52     this.firstPoint = false;
53     this.path.moveTo(this.lineDrawPoints.leftBottom.x, parseInt(vector[0].value) *
54       ↪ this.dimensionsUnits.y);
55   }
56   for (let i = 0; i < vector.length; i++) {
57     this.pushData(vector[i])
58   }
59 }
60
61 getPath = () => {
62   return this.path;
63 }
64
65 resetDraw = () => {
66   this.firstPoint = true;
67   this.path.rewind();
68   this.lastPointTime = 0;
69 }
70 }

```

B.13 Arquivo de constantes do App

```
1 export enum ControlStates {
2   Visible = 'Visible',
3   Hidden = 'Hidden',
4 }
5
6 export enum PlaybackStates {
7   Loading = 'Loading',
8   Playing = 'Playing',
9   Paused = 'Paused',
10  Buffering = 'Buffering',
11  Error = 'Error',
12  Ended = 'Ended',
13 }
14
15 export enum ErrorSeverity {
16   Fatal = 'Fatal',
17   NonFatal = 'NonFatal',
18 }
19
20 export type ErrorType = {
21   type: ErrorSeverity
22   message: string
23   obj: Record<string, unknown>
24 }
```

B.14 Arquivo de componente de navegação das telas

```
1 import { NavigationContainer } from "@react-navigation/native";
2 import { AppRoutes } from './Routes'
3
4
5 export function Routes() {
6   return (
7     <NavigationContainer>
8       <AppRoutes />
9     </NavigationContainer>
10  )
11 }
```

B.15 Arquivo inicialização do banco de dados

```
1 import * as SQLite from "expo-sqlite";
2
3
```



```
4 const db = SQLite.openDatabase("logs.db");
5
6 export default db;
```

B.16 Arquivo das funções de interface de acesso as tabelas do banco de dados

```
1 import db from './DataBase';
2 import Executions from './tables/Executions';
3 import Sniffers from './tables/Sniffers';
4 import Ports from './tables/Ports';
5 import Logs from './tables/Logs';
6
7
8 const tables = [Executions, Sniffers, Ports, Logs];
9
10 const clearTables = async () => {
11   for (let i = 0; i < tables.length; i++) {
12     const tableName = tables[i].tableName;
13     await new Promise((resolve, reject) => {
14       db.transaction((tx) => {
15         tx.executeSql(
16           `DROP TABLE IF EXISTS ${tableName};`,
17           [],
18           (_, { rowsAffected, insertId }) => resolve(console.log(`dropped ${tableName} table,
19             ↳ rowsAffected = ${rowsAffected}, insertId = ${insertId}`)),
20           (_, error) => {
21             console.log(`could not drop ${tableName} table`);
22             reject(error) // erro interno em tx.executeSql
23           }
24         );
25       });
26     });
27     console.log(`all data base tables dropped`);
28   }
29
30   const initTables = async (reseting = false) => {
31     if (reseting) await clearTables();
32
33     for (let i = 0; i < tables.length; i++) {
34       await tables[i].init();
35     }
36     console.log(`all data base tables initialized`);
37   }
38
39   const countRecords = async () => {
40     const recordsCounting = {};
41     for (let i = 0; i < tables.length; i++) {
42       const table = tables[i];
43       recordsCounting[table.tableName] = await table.countRecords();
44     }
45     console.log(`recordsCounting = ${JSON.stringify(recordsCounting)}`);
```

```
46   return recordsCounting;
47 }
48
49 const createExecution = async (execution, sniffers, ports) => {
50   const executionInfo = {};
51   const executionId = await Executions.create(execution);
52   executionInfo['executionId'] = executionId;
53   executionInfo['sniffers'] = [];
54   for (let i = 0; i < sniffers.length; i++) {
55     const sniffer = {
56       wsClientUrl: sniffers[i].getUrl(),
57       name: sniffers[i].getUrl(),
58     };
59     const snifferId = await Sniffers.appendSnifferOnExecution(sniffer, executionId);
60     executionInfo['sniffers'].push({
61       wsClientUrl: sniffer.wsClientUrl,
62       id: snifferId,
63       portIds: []
64     });
65     for (let j = 0; j < ports.length; j++) {
66       const port = {
67         name: ports[j].port,
68         sensorName: 'sensor de distância',
69         sensorType: 'ultrassonic',
70       };
71       const portId = await Ports.appendPortOnSniffer(port, snifferId);
72       executionInfo['sniffers'][i]['portIds'].push({
73         id: portId,
74         portName: port.name
75       });
76     }
77   }
78   return executionInfo;
79 }
80
81 const updateExecution = async (id, execution) => {
82   await Executions.update(id, execution);
83 }
84
85 const findExecution = async (id) => {
86   return await Executions.findExecution(id);
87 }
88
89 const removeFromTable = async (tableName, id) => {
90   return await new Promise((resolve, reject) => {
91     db.transaction((tx) => {
92       //comando SQL modificável
93       tx.executeSql(
94         `DELETE FROM ${tableName} WHERE id=?`,
95         [id],
96         //-----
97         (_, { rowsAffected }) => {
98           resolve(rowsAffected);
99         },
100         (_, error) => reject(error) // erro interno em tx.executeSql
101       );
102     });
103   });
104 }
105
106 const removeExecution = async (id) => {
```

```
107     const execution = await Executions.findExecution(id);
108     const sniffers = await Sniffers.findSniffers(execution.id);
109     for (let i = 0; i < sniffers.length; i++) {
110         const sniffer = sniffers[i];
111         const ports = await Ports.findPorts(sniffer.id);
112         for (let j = 0; j < ports.length; j++) {
113             const port = ports[j];
114             await Logs.remove(port.id);
115             await removeFromTable(Ports.tableName, port.id);
116         }
117         await removeFromTable(Sniffers.tableName, sniffer.id);
118     }
119     await removeFromTable(Executions.tableName, execution.id);
120 }
121
122 const removeAllTempExecutions = async () => {
123     const executions = await Executions.findAllTempExecutions();
124     for (let e = 0; e < executions.length; e++) {
125         const executionId = executions[e].id;
126         const sniffers = await Sniffers.findSniffers(executionId);
127         for (let i = 0; i < sniffers.length; i++) {
128             const sniffer = sniffers[i];
129             const ports = await Ports.findPorts(sniffer.id);
130             for (let j = 0; j < ports.length; j++) {
131                 const port = ports[j];
132                 await Logs.remove(port.id);
133                 await removeFromTable(Ports.tableName, port.id);
134             }
135             await removeFromTable(Sniffers.tableName, sniffer.id);
136         }
137         await removeFromTable(Executions.tableName, executionId);
138     }
139     console.log(`removeu todas as execuções temporárias`);
140 }
141
142 const findLastExecutionInfo = async () => {
143     return await Executions.findLastExecution();
144 }
145
146 const findExecutionInfo = async (executionId, logsTime = null) => {
147     const executionInfo = {};
148     const execution = await Executions.findExecution(executionId);
149     executionInfo['id'] = execution.id;
150     executionInfo['name'] = execution.name;
151     executionInfo['initDate'] = execution.initDate;
152     executionInfo['initTime'] = execution.initTime;
153     executionInfo['endTime'] = execution.endTime;
154     executionInfo['videoAsset'] = execution.videoAsset;
155     executionInfo['sniffers'] = [];
156     const sniffers = await Sniffers.findSniffers(executionId);
157     for (let i = 0; i < sniffers.length; i++) {
158         const sniffer = sniffers[i];
159         executionInfo['sniffers'].push({
160             id: sniffer.id,
161             name: sniffer.name,
162             wsClientUrl: sniffer.wsClientUrl,
163             ports: []
164         });
165         const ports = await Ports.findPorts(sniffer.id);
166         for (let j = 0; j < ports.length; j++) {
167             const port = ports[j];
```

```
168     executionInfo['sniffers'][i]['ports'].push({
169       id: port.id,
170       name: port.name,
171       sensorName: port.sensorName,
172       sensorType: port.sensorType,
173     });
174     if (logsTime) {
175       let begin = logsTime - 5000;
176       let end = logsTime + 5000;
177       if (begin < 0) {
178         end += -1 * begin;
179         begin = 0;
180       }
181       executionInfo['sniffers'][i]['ports'][j]['logs'] = await Logs.findLogs(port.id, { begin, end });
182     }
183   }
184 }
185 return executionInfo;
186 }
187
188 const findLogsByPort = async (portId, logsTime, bufferLimit = 30000) => {
189   if (logsTime >= 0) {
190     let begin = logsTime - 15000;
191     let end = logsTime + 15000;
192     if (begin < 0) {
193       end += -1 * begin;
194       begin = 0;
195     }
196     return await Logs.findLogsBuffer(portId, { begin, end }, bufferLimit);
197   }
198 }
199
200 const findLogsByPortAndInterval = async (portId, interval, bufferLimit = 15000) => {
201   const { begin, end } = interval;
202   return await Logs.findLogsBuffer(portId, { begin, end }, bufferLimit);
203 }
204
205 const getAllExecutions = async () => {
206   return await Executions.getAllRecords();
207 }
208
209
210 const appendLogsOnPort = (logs, portId) => {
211   Logs.appendLogsOnPort(logs, portId);
212 }
213
214 const findAllLogsByPortId = async portId => {
215   return await Logs.findAllLogs(portId);
216 }
217
218 export default {
219   createExecution,
220   countRecords,
221   initTables,
222   findExecutionInfo,
223   updateExecution,
224   findExecution,
225   removeExecution,
226   removeAllTempExecutions,
227   appendLogsOnPort,
228   findLastExecutionInfo,
```

```

229   findAllLogsByPortId,
230   findLogsByPort,
231   findLogsByPortAndInterval,
232   getAllExecutions,
233 };

```

B.17 Arquivo das operações da tabela de execuções

```

1  import db from "../DataBase";
2
3
4  const tableName = "executions";
5
6  /**
7   * INICIALIZAÇÃO DA TABELA
8   * - Executa sempre, mas só cria a tabela caso não exista (primeira execução)
9   */
10 const init = async () => {
11   await new Promise((resolve, reject) => {
12     db.transaction((tx) => {
13       tx.executeSql(
14         `CREATE TABLE IF NOT EXISTS ${tableName} (
15           id INTEGER PRIMARY KEY AUTOINCREMENT,
16           name TEXT,
17           initDate TEXT,
18           initTime TEXT,
19           endTime TEXT,
20           videoAsset TEXT);`,
21         [],
22         (_, { rowsAffected, insertId }) => resolve(console.log(`created ${tableName} table, rowsAffected
23         ↵ = ${rowsAffected}, insertId = ${insertId}`)),
24         (_, error) => {
25           console.log(`could not create ${tableName} table`);
26           reject(error) // erro interno em tx.executeSql
27         }
28       );
29     });
30   });
31 }
32
33 const create = async (execution) => {
34   return new Promise((resolve, reject) => {
35     db.transaction((tx) => {
36       //comando SQL modificável
37       tx.executeSql(
38         `INSERT INTO ${tableName} (name, initDate, initTime, endTime) values (?, ?, ?, ?)`,
39         [execution.name, execution.initDate, execution.initTime, ''],
40         (_, { rowsAffected, insertId }) => {
41           if (rowsAffected > 0) {
42             console.log(`created execution with id = ${insertId}`);
43             resolve(insertId);
44           }
45           else reject(`Error inserting execution: ${JSON.stringify(execution)}`); // insert falhou
46         },

```

```

47     (_, error) => reject(error) // erro interno em tx.executeSql
48   );
49   });
50 });
51 };
52
53 /**
54  * BUSCA TODOS OS REGISTROS DE UMA DETERMINADA TABELA
55  * - Não recebe parâmetros;
56  * - Retorna uma Promise:
57  * - O resultado da Promise é uma lista (Array) de objetos;
58  * - Pode retornar erro (reject) caso o ID não exista ou então caso ocorra erro no SQL;
59  * - Pode retornar um array vazio caso não existam registros.
60  */
61 const getAllRecords = () => {
62   return new Promise((resolve, reject) => {
63     db.transaction((tx) => {
64       //comando SQL modificável
65       tx.executeSql(
66         `SELECT * FROM ${tableName};`,
67         [],
68         //-----
69         (_, { rows }) => {
70           const appExecutions = rows._array.map(execution => {
71             execution.videoAsset = JSON.parse(execution.videoAsset);
72             return execution;
73           });
74           resolve(appExecutions);
75         },
76         (_, error) => reject(error) // erro interno em tx.executeSql
77       );
78     });
79   });
80 };
81
82 const deleteAllRecords = () => {
83   return new Promise((resolve, reject) => {
84     db.transaction((tx) => {
85       //comando SQL modificável
86       tx.executeSql(
87         `DELETE FROM ${tableName};`,
88         [],
89         //-----
90         (_, { rowsAffected }) => {
91           resolve(rowsAffected);
92         },
93         (_, error) => reject(error) // erro interno em tx.executeSql
94       );
95     });
96   });
97 };
98
99 const countRecords = async () => {
100   return await new Promise((resolve, reject) => {
101     db.transaction((tx) => {
102       //comando SQL modificável
103       tx.executeSql(
104         `SELECT COUNT(*) FROM ${tableName};`,
105         [],
106         //-----
107         (_, { rows }) => resolve(rows._array[0]["COUNT(*)"]),

```

```
108     (_, error) => reject(error) // erro interno em tx.executeSql
109   );
110   });
111   });
112   };
113
114   const findExecution = async (executionId) => {
115     return await new Promise((resolve, reject) => {
116       db.transaction((tx) => {
117         //comando SQL modificável
118         tx.executeSql(
119           `SELECT * FROM ${tableName} WHERE id = ${executionId};`,
120           [],
121           //-----
122           (_, { rows }) => {
123             console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
124             const appExecution = rows._array[0];
125             appExecution.videoAsset = JSON.parse(appExecution.videoAsset);
126             resolve(appExecution);
127           },
128           (_, error) => reject(error) // erro interno em tx.executeSql
129         );
130       });
131     });
132   }
133
134   const findLastExecution = async (executionId) => {
135     return await new Promise((resolve, reject) => {
136       db.transaction((tx) => {
137         //comando SQL modificável
138         tx.executeSql(
139           `SELECT * FROM ${tableName} ORDER BY id DESC LIMIT 1;`,
140           [],
141           //-----
142           (_, { rows }) => {
143             console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
144             const appExecution = rows._array[0];
145             appExecution.videoAsset = JSON.parse(appExecution.videoAsset);
146             resolve(appExecution);
147           },
148           (_, error) => reject(error) // erro interno em tx.executeSql
149         );
150       });
151     });
152   }
153
154   const findAllTempExecutions = async () => {
155     return await new Promise((resolve, reject) => {
156       db.transaction((tx) => {
157         //comando SQL modificável
158         tx.executeSql(
159           `SELECT * FROM ${tableName} WHERE videoAsset IS NULL;`,
160           [],
161           //-----
162           (_, { rows }) => {
163             console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
164             const appExecutions = rows._array.map(execution => {
165               execution.videoAsset = JSON.parse(execution.videoAsset);
166               return execution;
167             });
168             resolve(appExecutions);

```

```

169     },
170     (_, error) => reject(error) // erro interno em tx.executeSql
171   );
172   });
173   });
174 }
175
176 const update = async (id, execution) => {
177   const dbExecution = execution;
178   dbExecution.videoAsset = typeof execution.videoAsset === 'string' ? execution.videoAsset :
179   ⇐ JSON.stringify(execution.videoAsset);
180   return await new Promise((resolve, reject) => {
181     db.transaction((tx) => {
182       //comando SQL modificável
183       tx.executeSql(
184         `UPDATE ${tableName} SET name=?, initDate=?, initTime=?, endTime=?, videoAsset=? WHERE id=?`,
185         [dbExecution.name, dbExecution.initDate, dbExecution.initTime, dbExecution.endTime,
186         ⇐ dbExecution.videoAsset, id],
187         //-----
188         (_, { rowsAffected }) => {
189           if (rowsAffected > 0) resolve(rowsAffected);
190           else reject(`Error updating execution: id=${id}`); // nenhum registro alterado
191         },
192         (_, error) => reject(error) // erro interno em tx.executeSql
193       );
194     });
195   });
196 };
197
198 export default {
199   tableName,
200   init,
201   deleteAllRecords,
202   create,
203   countRecords,
204   getAllRecords,
205   findExecution,
206   findLastExecution,
207   findAllTempExecutions,
208   update,
209 };

```

B.18 Componente da tela de listagem das execuções

```

1  import { Text, View, StyleSheet } from "react-native";
2  import { useEffect, useState } from "react";
3  import DbOperations from "../../database/DbOperations";
4  import { ScreenBase } from "../../common/ScreenBase";
5  import ExecutionPlayer from "../../player/ExecutionPlayer";
6
7  export default function ExecutionScreen({ route, navigation }) {
8    const { executionId } = route.params;
9    const [execution, setExecution] = useState(null);
10
11    useEffect(() => {

```



```
12     (async () => {
13         if (executionId && typeof executionId === 'number') {
14             const auxExecution = await DbOperations.findExecution(executionId);
15             auxExecution.executionId = auxExecution.id;
16             setExecution(auxExecution);
17         }
18     })();
19 }, [executionId]);
20
21 let screen = <></>;
22 if (executionId && typeof executionId === 'number') {
23     if (execution) {
24         screen = <ExecutionPlayer execution={execution} />;
25     } else {
26         screen = <Text>Could not find an execution from id {executionId}</Text>;
27     }
28 } else {
29     screen = <Text>{executionId} is not a valid execution id</Text>;
30 }
31
32 return (
33     <>
34         {screen}
35         <ScreenBase openRoutesMenu={() => navigation.openDrawer()} />
36     </>
37 );
38 }
39
40 const styles = StyleSheet.create({
41     returnView: {
42         flex: 1,
43         alignItems: 'center',
44         justifyContent: 'center',
45     },
46 });
```

B.19 Declaração da store compartilhada entre os componentes

```
1 import RegisteredSniffersStore from "../sniffers/RegisteredSniffers.store";
2
3
4 const Stores = {
5     RegisteredSniffersStore,
6 }
7
8 export default Stores;
```

B.20 Componente de player do vídeo da execução

```
1 import { AVPlaybackStatus, Audio, Video } from 'expo-av'
2 import {
3   ActivityIndicator,
4   Animated,
5   StyleSheet,
6   Text,
7   TouchableWithoutFeedback,
8   View,
9 } from 'react-native'
10 import { ControlStates, ErrorSeverity, PlaybackStates } from './constants'
11 import {
12   ErrorMessage,
13   TouchableButton,
14   deepMerge,
15   getMinutesSecondsFromMilliseconds,
16   styles,
17 } from './utils'
18 import { MaterialIcons } from '@expo/vector-icons'
19 import { Props, defaultProps } from './props'
20 import { useEffect, useRef, useState } from 'react'
21 import React from 'react'
22 import Slider from '@react-native-community/slider'
23
24 const VideoPlayer = (tempProps: Props) => {
25   const props = deepMerge(defaultProps, tempProps) as Props
26
27   let playbackInstance: Video | null = null
28   let controlsTimer: NodeJS.Timeout | null = null
29   let initialShow = props.defaultControlsVisible
30   const header = props.header
31
32   const [errorMessage, setErrorMessage] = useState('')
33   const controlsOpacity = useRef(new Animated.Value(props.defaultControlsVisible ? 1 : 0)).current
34   const [controlsState, setControlsState] = useState(
35     props.defaultControlsVisible ? ControlStates.Visible : ControlStates.Hidden
36   )
37   const [playbackInstanceInfo, setPlaybackInstanceInfo] = useState({
38     position: 0,
39     duration: 0,
40     state: props.videoProps.source ? PlaybackStates.Loading : PlaybackStates.Error,
41   })
42
43   // We need to extract ref, because of mistypes in <Slider />
44   // eslint-disable-next-line @typescript-eslint/no-unused-vars
45   const { ref: sliderRef, ...sliderProps } = props.slider
46   const screenRatio = props.style.width! / props.style.height!
47
48   let videoHeight = props.style.height
49   let videoWidth = videoHeight! * screenRatio
50
51   if (videoWidth > props.style.width!) {
52     videoWidth = props.style.width!
53     videoHeight = videoWidth / screenRatio
54   }
55
56   useEffect(() => {
57     setAudio()
```

```
58
59   return () => {
60     if (playbackInstance) {
61       playbackInstance.setStatusAsync({
62         shouldPlay: false,
63       })
64     }
65   }
66 }, [])
67
68 useEffect(() => {
69   if (!props.videoProps.source) {
70     console.error(
71       '[VideoPlayer] `Source` is a required in `videoProps`. ' +
72       'Check https://docs.expo.io/versions/latest/sdk/video/#usage'
73     )
74     setErrorMessage(`\`Source` is a required in `videoProps`)
75     setPlaybackInstanceInfo({ ...playbackInstanceInfo, state: PlaybackStates.Error })
76   } else {
77     setPlaybackInstanceInfo({ ...playbackInstanceInfo, state: PlaybackStates.Playing })
78   }
79 }, [props.videoProps.source])
80
81 const hideAnimation = () => {
82   Animated.timing(controlsOpacity, {
83     toValue: 0,
84     duration: props.animation.fadeOutDuration,
85     useNativeDriver: true,
86   }).start(({ finished }) => {
87     if (finished) {
88       setControlsState(ControlStates.Hidden)
89     }
90   })
91 }
92
93 const animationToggle = () => {
94   if (controlsState === ControlStates.Hidden) {
95     Animated.timing(controlsOpacity, {
96       toValue: 1,
97       duration: props.animation.fadeInDuration,
98       useNativeDriver: true,
99     }).start(({ finished }) => {
100       if (finished) {
101         setControlsState(ControlStates.Visible)
102       }
103     })
104   } else if (controlsState === ControlStates.Visible) {
105     hideAnimation()
106   }
107
108   if (controlsTimer === null && props.autoHidePlayer) {
109     controlsTimer = setTimeout(() => {
110       if (
111         playbackInstanceInfo.state === PlaybackStates.Playing &&
112         controlsState === ControlStates.Hidden
113       ) {
114         hideAnimation()
115       }
116       if (controlsTimer) {
117         clearTimeout(controlsTimer)
118       }
119     })
120   }
121 }
```

```
119     controlsTimer = null
120   }, 2000)
121   }
122 }
123
124 // Set audio mode to play even in silent mode (like the YouTube app)
125 const setAudio = async () => {
126   try {
127     await Audio.setAudioModeAsync({
128       playsInSilentModeIOS: true,
129     })
130   } catch (e) {
131     props.errorCallback({
132       type: ErrorSeverity.NonFatal,
133       message: 'Audio.setAudioModeAsync',
134       obj: e as Record<string, unknown>,
135     })
136   }
137 }
138
139 const updatePlaybackCallback = (status: AVPlaybackStatus) => {
140   props.playbackCallback(status)
141
142   if (status.isLoaded) {
143     setPlaybackInstanceInfo({
144       ...playbackInstanceInfo,
145       position: status.positionMillis,
146       duration: status.durationMillis || 0,
147       state:
148         status.positionMillis === status.durationMillis
149           ? PlaybackStates.Ended
150           : status.isBuffering
151             ? PlaybackStates.Buffering
152             : status.shouldPlay
153               ? PlaybackStates.Playing
154               : PlaybackStates.Paused,
155     })
156     if (
157       (status.didJustFinish && controlsState === ControlStates.Hidden) ||
158       (status.isBuffering && controlsState === ControlStates.Hidden && initialShow)
159     ) {
160       animationToggle()
161       initialShow = false
162     }
163   } else {
164     if (status.isLoaded === false && status.error) {
165       const errorMsg = `Encountered a fatal error during playback: ${status.error}`
166       setErrorMessage(errorMsg)
167       props.errorCallback({ type: ErrorSeverity.Fatal, message: errorMsg, obj: {} })
168     }
169   }
170 }
171
172 const togglePlay = async () => {
173   if (controlsState === ControlStates.Hidden) {
174     return
175   }
176   const shouldPlay = playbackInstanceInfo.state !== PlaybackStates.Playing
177   if (playbackInstance !== null) {
178     await playbackInstance.setStatusAsync({
179       shouldPlay,
```

```
180     ...(playbackInstanceInfo.state === PlaybackStates.Ended && { positionMillis: 0 } ),
181   })
182   setPlaybackInstanceInfo({
183     ...playbackInstanceInfo,
184     state:
185       playbackInstanceInfo.state === PlaybackStates.Playing
186         ? PlaybackStates.Paused
187         : PlaybackStates.Playing,
188   })
189   if (shouldPlay) {
190     animationToggle()
191   }
192 }
193 }
194
195 if (playbackInstanceInfo.state === PlaybackStates.Error) {
196   return (
197     <View
198       style={{
199         backgroundColor: props.style.videoBackgroundColor,
200         width: videoWidth,
201         height: videoHeight,
202       }}
203     >
204     <ErrorMessage style={props.textStyle} message={errorMessage} />
205   </View>
206   )
207 }
208
209 if (playbackInstanceInfo.state === PlaybackStates.Loading) {
210   return (
211     <View
212       style={{
213         backgroundColor: props.style.controlsBackgroundColor,
214         width: videoWidth,
215         height: videoHeight,
216         justifyContent: 'center',
217       }}
218     >
219     {props.icon.loading || <ActivityIndicator {...props.activityIndicator} />}
220   </View>
221   )
222 }
223
224 return (
225   <View
226     style={{
227       backgroundColor: props.style.videoBackgroundColor,
228       width: videoWidth,
229       height: videoHeight,
230       maxWidth: '100%',
231     }}
232   >
233     <Video
234       style={styles.videoWrapper}
235       {...props.videoProps}
236       ref={component => {
237         playbackInstance = component
238         if (props.videoProps.ref) {
239           props.videoProps.ref.current = component as Video
240         }
241       }}
242     />
243   </View>
244 )
```

```

241     }}
242     onPlaybackStatusUpdate={updatePlaybackCallback}
243   />
244
245   <Animated.View
246     pointerEvents={controlsState === ControlStates.Visible ? 'auto' : 'none'}
247     style={[
248       styles.topInfoWrapper,
249       {
250         opacity: controlsOpacity,
251       },
252     ]}
253   >
254     {header}
255   </Animated.View>
256
257   <TouchableWithoutFeedback onPress={animationToggle}>
258     <Animated.View
259       style={{
260         ...StyleSheet.absoluteFillObject,
261         opacity: controlsOpacity,
262         justifyContent: 'center',
263         alignItems: 'center',
264       }}
265     >
266       <View
267         style={{
268           ...StyleSheet.absoluteFillObject,
269           backgroundColor: props.style.controlsBackgroundColor,
270           opacity: 0.5,
271         }}
272       />
273       <View pointerEvents={controlsState === ControlStates.Visible ? 'auto' : 'none'}>
274         <View style={styles.iconWrapper}>
275           <TouchableButton onPress={togglePlay}>
276             <View>
277               {playbackInstanceInfo.state === PlaybackStates.Buffering &&
278                 (props.icon.loading || <ActivityIndicator {...props.activityIndicator} />)}
279               {playbackInstanceInfo.state === PlaybackStates.Playing && props.icon.pause}
280               {playbackInstanceInfo.state === PlaybackStates.Paused && props.icon.play}
281               {playbackInstanceInfo.state === PlaybackStates.Ended && props.icon.replay}
282               {((playbackInstanceInfo.state === PlaybackStates.Ended && !props.icon.replay) ||
283                 (playbackInstanceInfo.state === PlaybackStates.Playing && !props.icon.pause) ||
284                 (playbackInstanceInfo.state === PlaybackStates.Paused &&
285                   !props.icon.pause)) && (
286                 <MaterialIcons
287                   name={
288                     playbackInstanceInfo.state === PlaybackStates.Playing
289                       ? 'pause'
290                       : playbackInstanceInfo.state === PlaybackStates.Paused
291                         ? 'play-arrow'
292                         : 'replay'
293                   }
294                   style={props.icon.style}
295                   size={props.icon.size}
296                   color={props.icon.color}
297                 />
298               )}
299             </View>
300           </TouchableButton>
301         </View>

```

```

302     </View>
303     </Animated.View>
304     </TouchableWithoutFeedback>
305
306     <Animated.View
307       pointerEvents={controlsState === ControlStates.Visible ? 'auto' : 'none'}
308       style={[
309         styles.bottomInfoWrapper,
310         {
311           opacity: controlsOpacity,
312         },
313       ]}
314     >
315       {props.timeVisible && (
316         <Text style={[props.textStyle, styles.timeLeft]}>
317           {getMinutesSecondsFromMilliseconds(playbackInstanceInfo.position)}
318         </Text>
319       )}
320       {props.slider.visible && (
321         <Slider
322           {...sliderProps}
323           style={[styles.slider, props.slider.style]}
324           value={
325             playbackInstanceInfo.duration
326               ? playbackInstanceInfo.position / playbackInstanceInfo.duration
327               : 0
328           }
329           onSlidingStart={() => {
330             if (playbackInstanceInfo.state === PlaybackStates.Playing) {
331               togglePlay()
332               setPlaybackInstanceInfo({ ...playbackInstanceInfo, state: PlaybackStates.Paused })
333             }
334           }}
335           onSlidingComplete={async e => {
336             const position = e * playbackInstanceInfo.duration
337             if (playbackInstance) {
338               await playbackInstance.setStatusAsync({
339                 positionMillis: position,
340                 shouldPlay: false,
341               })
342             }
343             setPlaybackInstanceInfo({
344               ...playbackInstanceInfo,
345               position,
346             })
347           }}
348         />
349       )}
350       {props.timeVisible && (
351         <Text style={[props.textStyle, styles.timeRight]}>
352           {getMinutesSecondsFromMilliseconds(playbackInstanceInfo.duration)}
353         </Text>
354       )}
355       {props.mute.visible && (
356         <TouchableButton
357           onPress={() => (props.mute.isMute ? props.mute.exitMute?.() : props.mute.enterMute?.())}
358         >
359         <View>
360           {props.icon.mute}
361           {props.icon.exitMute}
362           {(!props.icon.mute && props.mute.isMute) ||

```

```

363         (!props.icon.exitMute && !props.mute.isMute)) && (
364         <MaterialIcons
365           name={props.mute.isMute ? 'volume-up' : 'volume-off'}
366           style={props.icon.style}
367           size={props.icon.size! / 2}
368           color={props.icon.color}
369         />
370       )}
371     </View>
372   </TouchableButton>
373 )}
374 {props.fullscreen.visible && (
375   <TouchableButton
376     onPress={() =>
377       props.fullscreen.inFullscreen
378       ? props.fullscreen.exitFullscreen!()
379       : props.fullscreen.enterFullscreen!()
380     }
381   >
382     <View>
383       {!props.fullscreen.inFullscreen && props.icon.fullscreen}
384       {props.fullscreen.inFullscreen && props.icon.exitFullscreen}
385       {(!props.icon.fullscreen && !props.fullscreen.inFullscreen) ||
386         (!props.icon.exitFullscreen && props.fullscreen.inFullscreen)} && (
387         <MaterialIcons
388           name={props.fullscreen.inFullscreen ? 'fullscreen-exit' : 'fullscreen'}
389           style={props.icon.style}
390           size={props.icon.size! / 2}
391           color={props.icon.color}
392         />
393       )}
394     </View>
395   </TouchableButton>
396 )}
397 </Animated.View>
398 </View>
399 )
400 }
401
402 VideoPlayer.defaultProps = defaultProps
403
404 export {VideoPlayer}

```

B.21 Operações de banco na tabela de logs

```

1  import db from "../DataBase";
2
3
4  const tableName = "logs";
5
6  /**
7   * INICIALIZAÇÃO DA TABELA
8   * - Executa sempre, mas só cria a tabela caso não exista (primeira execução)
9   */
10 const init = async () => {

```



```

11  await new Promise((resolve, reject) => {
12    db.transaction((tx) => {
13      tx.executeSql(
14        `CREATE TABLE IF NOT EXISTS ${tableName} (
15          value TEXT,
16          time INT,
17          portId INTEGER,
18          CONSTRAINT portId FOREIGN KEY (portId)
19            REFERENCES ports(id)
20            ON DELETE CASCADE);`,
21        [],
22        (_, { rowsAffected, insertId }) => resolve(console.log(`created ${tableName} table, rowsAffected
23        ↳ = ${rowsAffected}, insertId = ${insertId}`)),
24        (_, error) => {
25          console.log(`could not create ${tableName} table`);
26          reject(error) // erro interno em tx.executeSql
27        }
28      });
29    });
30  }
31
32  const appendLogsOnPort = (logs, portId) => {
33    let batchInsertSqlStatement = `INSERT INTO ${tableName} (portId, value, time) values `;
34    for (let i = 0; i < logs.length; i++) {
35      batchInsertSqlStatement += `(${portId}, ${logs[i].value}, ${logs[i].time}), `;
36    }
37    batchInsertSqlStatement = batchInsertSqlStatement.slice(0, -2);
38    batchInsertSqlStatement += `;`;
39
40    return new Promise((resolve, reject) => {
41      db.transaction((tx) => {
42        //comando SQL modificável
43        tx.executeSql(
44          batchInsertSqlStatement,
45          [],
46          //-----
47          (_, { rowsAffected, insertId }) => {
48            if (rowsAffected > 0) {
49              console.log(`appendLogs(${logs.length}) success with insertId = ${insertId}`);
50              resolve(insertId);
51            }
52            else reject(`Error inserting logs: [${JSON.stringify(logs[0])}] ...
53            ↳ ${JSON.stringify(logs[logs.length - 1])}]`); // insert falhou
54          },
55          (_, error) => reject(error) // erro interno em tx.executeSql
56        );
57      });
58    });
59
60    /**
61     * BUSCA TODOS OS REGISTROS DE UMA DETERMINADA TABELA
62     * - Não recebe parâmetros;
63     * - Retorna uma Promise:
64     * - O resultado da Promise é uma lista (Array) de objetos;
65     * - Pode retornar erro (reject) caso o ID não exista ou então caso ocorra erro no SQL;
66     * - Pode retornar um array vazio caso não existam registros.
67     */
68    const getLogsFromPortInTimeFrame = ({ begin, end }, portId) => {
69      return new Promise((resolve, reject) => {

```

```

70 db.transaction((tx) => {
71   //comando SQL modificável
72   tx.executeSql(
73     `SELECT * FROM ${tableName}
74     WHERE portId = ${portId} AND
75     time BETWEEN ${begin} AND ${end}
76     ORDER BY time ASC
77     LIMIT 10000;`,
78     [],
79     //-----
80     (_, { rows }) => {
81       console.log(`got ${rows._array.length} logs from portId = ${portId} in timeframe ${begin} to
82         ↵ ${end} ms`);
83       resolve(rows._array);
84     },
85     (_, error) => reject(error) // erro interno em tx.executeSql
86   );
87 });
88 };
89
90 const deleteAllRecords = () => {
91   return new Promise((resolve, reject) => {
92     db.transaction((tx) => {
93       //comando SQL modificável
94       tx.executeSql(
95         `DELETE FROM ${tableName};`,
96         [],
97         //-----
98         (_, { rowsAffected }) => {
99           resolve(rowsAffected);
100         },
101         (_, error) => reject(error) // erro interno em tx.executeSql
102       );
103     });
104   });
105 };
106
107 const countRecords = async () => {
108   return await new Promise((resolve, reject) => {
109     db.transaction((tx) => {
110       //comando SQL modificável
111       tx.executeSql(
112         `SELECT COUNT(*) FROM ${tableName};`,
113         [],
114         //-----
115         (_, { rows }) => resolve(rows._array[0]["COUNT(*)"]),
116         (_, error) => reject(error) // erro interno em tx.executeSql
117       );
118     });
119   });
120 };
121
122 const findLogs = async (portId, { begin, end }) => {
123   return await new Promise((resolve, reject) => {
124     db.transaction((tx) => {
125       //comando SQL modificável
126       tx.executeSql(
127         `SELECT * FROM ${tableName} WHERE portId = ${portId} AND time BETWEEN ${begin} AND ${end} ORDER
128         ↵ BY time ASC LIMIT 10000;`,

```

```
129 //-----
130 (_, { rows }) => {
131   // console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
132   resolve(rows._array)
133 },
134 (_, error) => reject(error) // erro interno em tx.executeSql
135 );
136 });
137 });
138 }
139
140 const findLogsBuffer = async (portId, { begin, end }, bufferLimit) => {
141   return await new Promise((resolve, reject) => {
142     db.transaction((tx) => {
143       //comando SQL modificável
144       tx.executeSql(
145         `SELECT * FROM ${tableName} WHERE portId = ${portId} AND time BETWEEN ${begin} AND ${end} ORDER
146         ↵ BY time ASC LIMIT ${bufferLimit};`,
147         [],
148         //-----
149         (_, { rows }) => {
150           // console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
151           resolve(rows._array)
152         },
153         (_, error) => reject(error) // erro interno em tx.executeSql
154       );
155     });
156   });
157 }
158 const findAllLogs = async (portId) => {
159   return await new Promise((resolve, reject) => {
160     db.transaction((tx) => {
161       //comando SQL modificável
162       tx.executeSql(
163         `SELECT * FROM ${tableName} WHERE portId = ${portId} ORDER BY time ASC;`,
164         [],
165         //-----
166         (_, { rows }) => {
167           // console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
168           resolve(rows._array)
169         },
170         (_, error) => reject(error) // erro interno em tx.executeSql
171       );
172     });
173   });
174 }
175
176 const remove = async (portId) => {
177   return await new Promise((resolve, reject) => {
178     db.transaction((tx) => {
179       //comando SQL modificável
180       tx.executeSql(
181         `DELETE FROM ${tableName} WHERE portId=?;`,
182         [portId],
183         //-----
184         (_, { rowsAffected }) => {
185           resolve(rowsAffected);
186         },
187         (_, error) => reject(error) // erro interno em tx.executeSql
188       );
189     });
190   });
191 }
```

```

189     });
190   });
191 };
192
193 export default {
194   tableName,
195   findAllLogs,
196   findLogsBuffer,
197   init,
198   deleteAllRecords,
199   appendLogsOnPort,
200   countRecords,
201   getLogsFromPortInTimeFrame,
202   findLogs,
203   remove
204 };

```

B.22 Operações de banco na tabela de portas do sensor

```

1  import db from "../DataBase";
2
3
4  const tableName = "ports";
5
6  /**
7   * INICIALIZAÇÃO DA TABELA
8   * - Executa sempre, mas só cria a tabela caso não exista (primeira execução)
9   */
10 const init = async () => {
11   await new Promise((resolve, reject) => {
12     db.transaction((tx) => {
13       tx.executeSql(
14         `CREATE TABLE IF NOT EXISTS ${tableName} (
15           id INTEGER PRIMARY KEY AUTOINCREMENT,
16           name TEXT,
17           sensorName TEXT,
18           sensorType TEXT,
19           snifferId INTEGER,
20           CONSTRAINT snifferId FOREIGN KEY (snifferId)
21             REFERENCES sniffers(id)
22             ON DELETE CASCADE);`,
23         [],
24         (_, { rowsAffected, insertId }) => resolve(console.log(`created ${tableName} table, rowsAffected
25         ↳ = ${rowsAffected}, insertId = ${insertId}`)),
26         (_, error) => {
27           console.log(`could not create ${tableName} table`);
28           reject(error) // erro interno em tx.executeSql
29         }
30       );
31     });
32   });
33 }
34
35 const appendPortOnSniffer = (port, snifferId) => {
36   return new Promise((resolve, reject) => {

```

```

36 db.transaction((tx) => {
37   //comando SQL modificável
38   tx.executeSql(
39     `INSERT INTO ${tableName} (name, sensorName, sensorType, snifferId) values (?, ?, ?, ?)`,
40     [port.name, port.sensorName, port.sensorType, snifferId],
41     //-----
42     (_, { rowsAffected, insertId }) => {
43       if (rowsAffected > 0) {
44         console.log(`created port with id = ${insertId}`);
45         resolve(insertId);
46       }
47       else reject(`Error inserting execution: ${JSON.stringify(execution)}`); // insert falhou
48     },
49     (_, error) => reject(error) // erro interno em tx.executeSql
50   );
51 });
52 });
53 };
54
55 /**
56  * BUSCA TODOS OS REGISTROS DE UMA DETERMINADA TABELA
57  * - Não recebe parâmetros;
58  * - Retorna uma Promise:
59  * - O resultado da Promise é uma lista (Array) de objetos;
60  * - Pode retornar erro (reject) caso o ID não exista ou então caso ocorra erro no SQL;
61  * - Pode retornar um array vazio caso não existam registros.
62  */
63 const getPortsFromSniffer = snifferId => {
64   return new Promise((resolve, reject) => {
65     db.transaction((tx) => {
66       //comando SQL modificável
67       tx.executeSql(
68         `SELECT * FROM ${tableName} WHERE snifferId = ${snifferId};`,
69         [],
70         //-----
71         (_, { rows }) => {
72           console.log(`got ports from snifferId = ${snifferId}`);
73           resolve(rows._array);
74         },
75         (_, error) => reject(error) // erro interno em tx.executeSql
76       );
77     });
78   });
79 };
80
81 const deleteAllRecords = () => {
82   return new Promise((resolve, reject) => {
83     db.transaction((tx) => {
84       //comando SQL modificável
85       tx.executeSql(
86         `DELETE FROM ${tableName};`,
87         [],
88         //-----
89         (_, { rowsAffected }) => {
90           resolve(rowsAffected);
91         },
92         (_, error) => reject(error) // erro interno em tx.executeSql
93       );
94     });
95   });
96 };

```

```

97
98 const countRecords = async () => {
99   return await new Promise((resolve, reject) => {
100     db.transaction((tx) => {
101       //comando SQL modificável
102       tx.executeSql(
103         `SELECT COUNT(*) FROM ${tableName};`,
104         [],
105         //-----
106         (_, { rows }) => resolve(rows._array[0]["COUNT(*)"]),
107         (_, error) => reject(error) // erro interno em tx.executeSql
108       );
109     });
110   });
111 };
112
113 const findPorts = async (snifferId) => {
114   return await new Promise((resolve, reject) => {
115     db.transaction((tx) => {
116       //comando SQL modificável
117       tx.executeSql(
118         `SELECT * FROM ${tableName} WHERE snifferId = ${snifferId};`,
119         [],
120         //-----
121         (_, { rows }) => {
122           console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
123           resolve(rows._array)
124         },
125         (_, error) => reject(error) // erro interno em tx.executeSql
126       );
127     });
128   });
129 }
130
131 export default {
132   tableName,
133   init,
134   deleteAllRecords,
135   appendPortOnSniffer,
136   countRecords,
137   getPortsFromSniffer,
138   findPorts
139 };

```

B.23 Componente de visualização da execução

```

1 import { Text, View, StyleSheet, TouchableOpacity } from "react-native";
2 import { useEffect, useState } from "react";
3 import DbOperations from "../../database/DbOperations";
4 import { ScreenBase } from "../../common/ScreenBase";
5 import ExecutionPlayer from "../../player/ExecutionPlayer";
6 import * as MediaLibrary from "expo-media-library";
7 import { observer, inject } from "mobx-react";
8
9 function PreviewScreen({ route, navigation, RegisteredSniffersStore }) {

```

```

10   const {
11     setExecutionVideo,
12   } = RegisteredSniffersStore;
13   const { execution } = route.params;
14   const [hasMediaLibraryPermission, setHasMediaLibraryPermission] = useState();
15
16   const saveExecution = async (videoUri) => {
17     const asset = await MediaLibrary.createAssetAsync(videoUri);
18     setExecutionVideo(asset); // continue here, this needs registeredsnifferstore
19   };
20
21   useEffect(() => {
22     (async () => {
23       const mediaLibraryPermission = await MediaLibrary.requestPermissionsAsync();
24       console.log(`mediaLibraryPermission = ${JSON.stringify(mediaLibraryPermission)}`);
25       setHasMediaLibraryPermission(mediaLibraryPermission.status === "granted");
26     })()
27   }, [execution]);
28
29   let screen = <</>;
30   if (execution?.videoAsset?.uri) {
31     screen =
32       <>
33         <ExecutionPlayer execution={execution} />
34         {hasMediaLibraryPermission ? (
35           <TouchableOpacity
36             style={styles.saveButton}
37             onPress={async () => {
38               // it needs to wait to make sure the video execution is saved before exiting
39               ↵ and destroying the page
40               await saveExecution(execution.videoAsset.uri);
41               navigation.navigate('gravar');
42             }}
43           <Text style={{ color: "white" }}>SAVE</Text>
44         </TouchableOpacity>
45         ) : undefined}
46         <TouchableOpacity
47           style={styles.discardButton}
48           onPress={() => navigation.navigate('gravar')}
49         >
50           <Text style={{ color: "white" }}>DISCARD</Text>
51         </TouchableOpacity>
52         { /* <ScreenBase openRoutesMenu={() => navigation.openDrawer()} /> */ }
53       </>;
54   } else {
55     screen =
56       <View style={styles.errorText}>
57         <Text>Could not find the execution video from uri {execution?.videAsset?.uri}</Text>
58       </View>;
59   }
60
61   return (
62     <>
63       {screen}
64       <ScreenBase openRoutesMenu={() => navigation.openDrawer()} />
65     </>
66   );
67 }
68
69 const styles = StyleSheet.create({

```

```
70   errorText: {
71     flex: 1,
72     alignItems: 'center',
73     justifyContent: 'center',
74   },
75   saveButton: {
76     position: "absolute",
77     width: 45,
78     height: 40,
79     alignItems: "center",
80     justifyContent: "center",
81     left: 100,
82     bottom: 10,
83     backgroundColor: "#1299FA",
84     borderRadius: 2,
85   },
86   discardButton: {
87     position: "absolute",
88     width: 70,
89     height: 40,
90     alignItems: "center",
91     justifyContent: "center",
92     left: 180,
93     bottom: 10,
94     backgroundColor: "#1299FA",
95     borderRadius: 2,
96   },
97 });
98
99 export default inject("RegisteredSniffersStore")(observer(PreviewScreen));
```

B.24 Propriedades do componente de video player

```
1 import { AVPlaybackStatus, Video, VideoProps } from 'expo-av'
2 import { ActivityIndicatorProps, Dimensions, Platform, TextStyle } from 'react-native'
3 import { ColorValue } from 'react-native'
4 import { ErrorType } from './constants'
5 import { MutableRefObject, ReactNode } from 'react'
6 import { SliderProps } from '@react-native-community/slider'
7
8 // https://github.com/typescript-cheatsheets/react/issues/415
9 export type Props = RequiredProps & DefaultProps
10
11 export const defaultProps = {
12   errorCallback: error =>
13     console.error(`[VideoPlayer] ${error.type} Error - ${error.message}: ${error.obj}`),
14   // eslint-disable-next-line @typescript-eslint/no-empty-function
15   playbackCallback: () => {},
16   defaultControlsVisible: false,
17   timeVisible: true,
18   slider: {
19     visible: true,
20   },
21   textStyle: {
22     color: '#FFF',
```



```
23     fontSize: 12,
24     textAlign: 'center',
25   },
26   activityIndicator: {
27     size: 'large',
28     color: '#999',
29   },
30   animation: {
31     fadeInDuration: 300,
32     fadeOutDuration: 300,
33   },
34   style: {
35     width: Platform.OS === 'web' ? '100%' : Dimensions.get('window').width,
36     height: Dimensions.get('window').height,
37     videoBackgroundColor: '#000',
38     controlsBackgroundColor: '#000',
39   },
40   icon: {
41     size: 48,
42     color: '#FFF',
43     style: {
44       padding: 2,
45     },
46   },
47   fullscreen: {
48     enterFullscreen: () =>
49       // eslint-disable-next-line no-console
50       console.log('[VideoPlayer] - missing `enterFullscreen` function in `fullscreen` prop'),
51     exitFullscreen: () =>
52       // eslint-disable-next-line no-console
53       console.log('[VideoPlayer] - missing `exitFullscreen` function in `fullscreen` prop'),
54     inFullscreen: false,
55     visible: true,
56   },
57   autoHidePlayer: true,
58   header: undefined,
59   mute: {
60     enterMute: () =>
61       // eslint-disable-next-line no-console
62       console.log('[VideoPlayer] - missing `enterMute` function in `mute` prop'),
63     exitMute: () =>
64       // eslint-disable-next-line no-console
65       console.log('[VideoPlayer] - missing `exitMute` function in `mute` prop'),
66     isMute: false,
67     visible: false,
68   },
69 } as DefaultProps
70
71 type RequiredProps = {
72   videoProps: VideoProps & {
73     ref?: MutableRefObject<Video>
74   }
75 }
76
77 type DefaultProps = {
78   errorCallback: (error: ErrorType) => void
79   playbackCallback: (status: AVPlaybackStatus) => void
80   defaultControlsVisible: boolean
81   timeVisible: boolean
82   textStyle: TextStyle
83   slider: {
```

```
84     visible?: boolean
85   } & SliderProps
86   activityIndicator: ActivityIndicatorProps
87   animation: {
88     fadeInDuration?: number
89     fadeOutDuration?: number
90   }
91   header: ReactNode
92   style: {
93     width?: number
94     height?: number
95     videoBackgroundColor?: ColorValue
96     controlsBackgroundColor?: ColorValue
97   }
98   icon: {
99     size?: number
100    color?: ColorValue
101    style?: TextStyle
102    pause?: JSX.Element
103    play?: JSX.Element
104    replay?: JSX.Element
105    loading?: JSX.Element
106    fullscreen?: JSX.Element
107    exitFullscreen?: JSX.Element
108    mute?: JSX.Element
109    exitMute?: JSX.Element
110  }
111  fullscreen: {
112    enterFullscreen?: () => void
113    exitFullscreen?: () => void
114    inFullscreen?: boolean
115    visible?: boolean
116  }
117  autoHidePlayer: boolean
118  mute: {
119    enterMute?: () => void
120    exitMute?: () => void
121    isMute?: boolean
122    visible?: boolean
123  }
124 }
```

B.25 Componente da tela de gravação

```
1 import {
2   StyleSheet,
3   Text,
4   View,
5   Button,
6   ScrollView,
7 } from "react-native";
8 import { useEffect, useState, useRef } from "react";
9 import { Camera } from "expo-camera";
10 import * as MediaLibrary from "expo-media-library";
11 import { ScreenBase } from "../common/ScreenBase";
```

```
12 import SensoresList from "../../screens/sensores/SensoresList.js";
13 import { observer, inject } from "mobx-react";
14 import DbOperations from "../../database/DbOperations";
15
16
17 const minutes = 30; // base em minutos
18 const timeLimit = 60 * minutes; // base em 60 segundos = 1 min
19 const seconds32 = 32 * 1000; // base em milisegundos = 1000 -> 32 segundos
20 const timeLimitTimeout = (60000 * minutes) - seconds32; // (60000 = 1 minuto * minutes) - 32 segundos
21
22 let thread = null;
23 let timeoutEvent = null;
24
25 const TimerDisplay = () => {
26   const [timer, setTimer] = useState(30);
27
28   useEffect(() => {
29     thread = setInterval(() => {
30       setTimer((timer) => {
31         if (timer > 0) return timer - 1;
32         else {
33           return timer;
34         }
35       });
36     }, 1000);
37   }, []);
38
39   return (
40     <View style={styles.timerBackground}>
41       <Text style={styles.timer}>{timer}</Text>
42     </View>
43   );
44 }
45
46 function Recording({ navigation, RegisteredSniffersStore }) {
47   const {
48     startLogs,
49     stopLogs,
50     getExecutionInfo,
51   } = RegisteredSniffersStore;
52
53   const [timeOutComponent, setTimeOutComponent] = useState(null);
54
55   let cameraRef = useRef();
56   const [hasCameraPermission, setHasCameraPermission] = useState();
57   const [hasMicrophonePermission, setHasMicrophonePermission] = useState();
58   const [hasMediaLibraryPermission, setHasMediaLibraryPermission] = useState();
59   const [isRecording, setIsRecording] = useState(false);
60   const [video, setVideo] = useState();
61
62   useEffect(() => {
63     (async () => {
64       const cameraPermission = await Camera.requestCameraPermissionsAsync();
65       const microphonePermission =
66         await Camera.requestMicrophonePermissionsAsync();
67       const mediaLibraryPermission =
68         await MediaLibrary.requestPermissionsAsync();
69
70       setHasCameraPermission(cameraPermission.status === "granted");
71       setHasMicrophonePermission(microphonePermission.status === "granted");
72       setHasMediaLibraryPermission(mediaLibraryPermission.status === "granted");
```

```
73
74     await DbOperations.removeAllTempExecutions();
75   }());
76 }, []);
77
78 if (
79   hasCameraPermission === undefined ||
80   hasMicrophonePermission === undefined
81 ) {
82   return <Text>Requestion permissions...</Text>;
83 } else if (!hasCameraPermission) {
84   return <Text>Permission for camera not granted.</Text>;
85 }
86
87 let recordVideo = () => {
88   startLogs();
89   setIsRecording(true);
90   let options = {
91     quality: "480p",
92     maxDuration: timeLimit, // 60 segundos * 30 = 30 min, 30 minuto funciona bem
93     mute: false,
94   };
95
96   cameraRef.current.recordAsync(options).then((recordedVideo) => {
97     // this is called when stopRecording(), maxDuration o maxFileSize is reached
98     stopLogs();
99     setVideo(recordedVideo);
100    setIsRecording(false);
101    // navigation.navigate('execution-preview', {video: recordedVideo, execution});
102  });
103
104   timeoutEvent = setTimeout(() => {
105     setTimeOutComponent(<TimerDisplay />);
106   }, timeLimitTimeout);
107 };
108
109 if (video) {
110   clearInterval(thread);
111   thread = null;
112   clearTimeout(timeoutEvent);
113   timeoutEvent = null;
114   // setTimeOutComponent(null);
115   const execution = getExecutionInfo();
116   execution['videoAsset'] = { uri: video.uri };
117   /* const execution = {
118     executionId: 2,
119     sniffers: [
120       {
121         wsClientUrl: "ws://192.168.1.199:81",
122         id: 2,
123         portIds: [
124           { id: 3, portName: "port1" },
125           { id: 4, portName: "port2" },
126         ],
127       },
128     ],
129     videoAsset: {
130       "mediaType": "video",
131       "modificationTime": 1686517909000,
132       "uri": "file:///storage/emulated/0/DCIM/1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4",
133       "filename": "1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4",
```

```
134     "width": 1080,
135     "id": "1000010523",
136     "creationTime": 1686517904000,
137     "albumId": "-2075821635",
138     "height": 1920,
139     "duration": 7.783
140   },
141   }; */
142
143   navigation.navigate('execution-preview', { execution: execution });
144 }
145
146 return (
147   <View style={styles.returnValue}>
148     <View style={styles.viewContainer}>
149       <Camera style={styles.cameraContainer} ref={cameraRef} >
150         <Button
151           title={isRecording ? "Stop Recording" : "Record Video"}
152           onPress={isRecording ? () => { cameraRef.current.stopRecording(); } : recordVideo}
153         />
154         {timeOutComponent}
155       </Camera>
156     </View>
157     <ScrollView>
158       <SensoresList />
159     </ScrollView>
160     <ScreenBase openRoutesMenu={() => navigation.openDrawer()} />
161   </View>
162 );
163 }
164
165 const styles = StyleSheet.create({
166   viewContainer: {
167     height: 300,
168   },
169   cameraContainer: {
170     flex: 1,
171     alignItems: "center",
172     justifyContent: "flex-end",
173     paddingBottom: 5,
174   },
175   videoContainer: {
176     flex: 1,
177   },
178   videoButtonsView: {
179     flex: 1,
180     alignItems: "center",
181     justifyContent: "flex-end",
182     paddingBottom: 5,
183   },
184   video: {
185     flex: 1,
186   },
187   returnValue: {
188     flex: 1,
189   },
190   timer: {
191     fontSize: 18,
192     color: 'white',
193   },
194   timerBackground: {
```

```

195     width: 40,
196     height: 40,
197     borderRadius: 20, // Half the width and height to create a circle
198     backgroundColor: '#3B3B3B',
199     justifyContent: 'center',
200     alignItems: 'center',
201     position: 'absolute',
202     right: 4,
203     bottom: 4,
204   }
205 });
206
207 export default inject("RegisteredSniffersStore")(observer(Recording));

```

B.26 Componente de definição do sensor

```

1  import { useState } from 'react';
2  import { Text, View, Button } from "react-native";
3  import { styles } from "../RegisteredSniffer.styles";
4  import { Ionicons } from '@expo/vector-icons';
5  import DropDownPicker from 'react-native-dropdown-picker';
6
7
8  function DefineSensor({ url, portName, sensorType, zIndex, setSensorType }) {
9    const [open, setOpen] = useState(false);
10   const [value, setValue] = useState(sensorType);
11
12   return (
13     <View>
14       <Text>{portName}</Text>
15       <DropDownPicker
16         placeholder={'Define sensor type'}
17         zIndex={zIndex}
18         open={open}
19         value={value}
20         items={[
21           { label: 'undefined', value: undefined },
22           { label: 'ultrasonic', value: 'ultrasonic' }
23         ]}
24         setOpen={setOpen}
25         setValue={setValue}
26         onChangeValue={value => setSensorType(url, portName, value)}
27       />
28     </View>
29   );
30 }
31
32 function RegisteredSniffer({ name, url, status, connect, disconnect, sensors, setSensorType }) {
33   const statusColor = {
34     "desconectado": "#666666",
35     "conectado": "#8FF399",
36   }[status];
37
38   let counter = sensors.length + 1;
39

```

```

40   return (
41     <View style={[styles.card, styles.shadowProp]}>
42       <Text style={[styles.heading, styles.font]}>Sniffer {name || url}</Text>
43       <View style={styles.statusContainer}>
44         <Ionicons style={styles.statusComponent} name="ios-hardware-chip-sharp" size={24}
45           ↪ color={statusColor} />
46         <Text style={[styles.font]}>{status}</Text>
47       </View>
48       <View style={styles.buttons}>
49         <Button style={styles.button} onPress={() => connect(url)} title='Conectar' />
50         <Button style={styles.button} onPress={() => disconnect(url)} title='Desconectar' />
51       </View>
52       {status == 'conectado' && (
53         <View>
54           {sensors.length == 0 && (<Text>sniffer has no ports connected</Text>)}
55           {sensors.length > 0 && sensors.map(port => {
56             counter -= 1;
57             const definition = { url, ...port, zIndex: counter, setSensorType: setSensorType };
58             return <DefineSensor key={port.portName} {...definition} />
59           })}
60         </View>
61       )}
62     </View>
63   );
64 }
65 export default RegisteredSniffer;

```

B.27 Componente de estilo css

```

1  import { StyleSheet } from "react-native";
2
3
4  export const styles = StyleSheet.create({
5    font: {
6      fontSize: 18,
7    },
8    heading: {
9      marginBottom: 13,
10   },
11   statusContainer: {
12     flexDirection: "row",
13     textAlignVertical: 'center',
14     marginBottom: 13,
15   },
16   statusComponent: {
17     marginRight: 13,
18   },
19   card: {
20     backgroundColor: 'white',
21     borderRadius: 5,
22     padding: 10,
23     marginVertical: 10,
24     marginHorizontal: 20,
25   },

```

```
26   shadowProp: {
27     shadowOffset: {width: -2, height: 4},
28     shadowColor: '#171717',
29     shadowOpacity: 0.2,
30     shadowRadius: 3,
31   },
32   snifferContainer: {
33     borderRadius: '10px',
34     background: '#FFFFFF',
35     boxShadow: '0px 4px 4px rgba(0, 0, 0, 0.25)',
36   },
37   buttons: {
38     flexDirection: "row",
39     margin: 13,
40   },
41   button: {
42     marginRight: 13,
43     margin: 13,
44     padding: 13,
45   },
46 });
```

B.28 Componente de exibição dos sniffers cadastrados

```
1  import { Text, View } from "react-native";
2  import { ScreenBase } from "../common/ScreenBase";
3  import RegisteredSniffer from "../../components/sniffer/RegisteredSniffer";
4  import { observer, inject } from 'mobx-react';
5  import { styles } from './RegisteredSniffers.styles';
6
7
8  function RegisteredSniffers({ navigation, RegisteredSniffersStore }) {
9    const { registeredSniffers, connect, disconnect, setSensorType } = RegisteredSniffersStore;
10
11    return (
12      <View style={styles.view}>
13        {registeredSniffers.map(sniffer => {
14          const item = {...sniffer, connect: connect, disconnect: disconnect, setSensorType:
15            ↪ setSensorType};
16          return <RegisteredSniffer key={item.url} {...item} />;
17        })}
18        <ScreenBase openRoutesMenu={() => {
19          navigation.openDrawer();
20        }} />
21      </View>
22    );
23  }
24  export default inject('RegisteredSniffersStore')(observer(RegisteredSniffers));
```


B.29 Componente de compartilhamento de stados globais da aplicação

```
1 import { action, makeObservable, observable } from 'mobx';
2 import WsClient from '../../components/socket/WsClient';
3 import { ChartDrawPath } from '../../charts/ChartDrawPath';
4 import DbOperations from '../../database/DbOperations';
5
6
7 class RegisteredSniffersStore {
8   // register last command sent to all wsClients
9   lastCmdToAllWsClients = "stop logs";
10
11   // wsClient connections
12   wsClients = [];
13   loadLogsThread = null;
14
15   // observables for sniffers screens
16   registeredSniffers = [];
17
18   // path to the port and sensortype
19   // example:
20   // portChart = [
21   //   {
22   //     url: 'ws://192.168.1.199:81',
23   //     port: 'port1',
24   //     chart: new LineChart([0, 100], [0, 255]),
25   //   },
26   //   ...
27   // ]
28   portChart = []
29
30   // executionInfo = {
31   //   executionId: 'id',
32   //   sniffers: [
33   //     {
34   //       wsClientUrl: 'url do sniffer',
35   //       id: 'identificador do sniffer no banco',
36   //       portIds: [
37   //         {
38   //           id: 'id1',
39   //           portName: 'portName'
40   //         }
41   //       ]
42   //     }
43   //   ]
44   // }
45   executionInfo = {};
46   executionInfoReady = false;
47   database = null;
48
49   // countLogsRecordsSaved = null;
50
51   constructor() {
52     makeObservable(this, {
53       // observables for sniffers screens
54       registeredSniffers: observable,
55
```

```

56     // sniffers registration methods
57     register: action,
58
59     // wsClient methods
60     connect: action,
61     disconnect: action,
62     updateSnifferStatus: action,
63
64     // ports and sensors
65     registerConnectedPorts: action,
66     setSensorType: action,
67
68     // logs rendering methods
69     lastCmdToAllWsClients: observable,
70     startLogs: action,
71     stopLogs: action,
72   })
73
74   this.register('pré cadastrado', 'ws://192.168.1.199:81'); // just for testing
75   // this.register('pré cadastrado', 'ws://192.168.0.246:81'); // just for testing judenilson
76
77   // thread gets logs from wsClient buffers and pushes them to the charts
78   this.loadLogsThread = setInterval(this.getWsClientsBufferedLogs, 0);
79 }
80
81 getWsClientsBufferedLogs = () => {
82   let logs;
83   let ports;
84   for (let i = 0; i < this.wsClients.length; i++) {
85     const url = this.wsClients[i].getUrl();
86     logs = this.wsClients[i].getLogs(120);
87     ports = Object.keys(logs);
88     for (let j = 0; j < ports.length; j++) {
89       const portName = ports[j];
90       this.pushDataPortChart(this.wsClients[i].getUrl(), portName, logs[ports[j]]);
91     }
92   }
93 }
94
95 // ports and sensors
96 getAllPortChart = () => {
97   return this.portChart;
98 }
99
100 getAllportChartForChartCardsList = () => {
101   const array = [];
102   for (let i = 0; i < this.portChart.length; i++) {
103     const port = this.portChart[i];
104     const obj = {
105       sensorName: port.port,
106       sensorType: 'ultrassonic',
107       timeFrame: 10,
108       logsRate: 1000,
109       drawPath: port.chart.getPath()
110     };
111     array.push(obj);
112   }
113   return array;
114 }
115 getSnifferSensorsDescription = (wsClientUrl) => {
116   const filtered = this.portChart.filter(port => port.url == wsClientUrl);

```

```
117     return filtered.map(sensorDescription => {
118         return {
119             sensorType: 'ultrasonic',
120             portName: sensorDescription.port,
121             sensorName: sensorDescription.port,
122         };
123     })
124 }
125 getPortChart = (wsClientUrl, portName) => {
126     return this.portChart.find(port => port.url == wsClientUrl && port.port == portName)
127 }
128 createChart = () => {
129     // return new LineChart([0, 100], [0, 255]);
130     return new ChartDrawPath('ultrasonic');
131 }
132 setPortChart = (wsClientUrl, portName) => {
133     const portChartRef = this.getPortChart(wsClientUrl, portName);
134     const wsClient = this.getWsClient(wsClientUrl);
135     if (portChartRef) {
136         portChartRef.chart = this.createChart();
137     } else {
138         this.portChart.push(
139             {
140                 url: wsClientUrl,
141                 port: portName,
142                 chart: this.createChart(),
143             }
144         );
145         wsClient.setLogsBufferPort(portName);
146     }
147 }
148 removePortChart = (wsClientUrl, portName) => {
149     const portChartIndex = this.portChart.findIndex(port => port.url == wsClientUrl && port.port ==
150     ↵ portName);
151     const wsClient = this.getWsClient(wsClientUrl);
152     if (portChartIndex > -1) {
153         this.portChart.splice(portChartIndex, 1);
154         wsClient.removeLogsBufferPort(portName);
155     }
156 }
157 pushDataPortChart = (wsClientUrl, portName, dataVector) => {
158     const portChartRef = this.getPortChart(wsClientUrl, portName);
159     if (portChartRef) {
160         portChartRef.chart.loadDataVector(dataVector);
161     }
162 }
163 cleanAllCharts = () => {
164     for (let i = 0; i < this.portChart.length; i++) {
165         this.portChart[i].chart.resetDraw();
166     }
167 }
168 registerConnectedPorts = (url, ports) => {
169     const sniffer = this.getRegisteredSniffer(url);
170     sniffer.sensors = ports.map(port => { return { sensorType: undefined, portName: port } });
171 }
172 setSensorType = (url, portName, sensorType) => {
173     const sniffer = this.getRegisteredSniffer(url);
174     const port = sniffer.sensors.find(port => port.portName == portName);
175     port.sensorType = sensorType;
176 }
```

```
177 // send information for sniffer
178 const socket = this.getWsClient(url);
179 if (socket) {
180   socket.send(JSON.stringify({
181     cmd: 'port config',
182     portName: portName,
183     sensorType: sensorType
184   }));
185 }
186
187 // configure its sensor chart
188 if (sensorType) {
189   this.setPortChart(url, portName);
190 } else {
191   this.removePortChart(url, portName);
192 }
193 }
194
195 getWsClient = url => {
196   return this.wsClients.find(socket => socket.getUrl() == url);
197 }
198
199 // logs rendering methods
200 getLogsInTime = seconds => {
201   this.clearPresentLogs();
202   this.startLogs();
203   setTimeout(() => this.stopLogs(), seconds * 1000);
204 }
205
206 // sniffers registration methods
207 register = (name, url) => {
208   this.registeredSniffers.push({
209     name: name,
210     url: url,
211     status: 'desconectado',
212     sensors: [],
213   });
214   this.wsClients.push(new WsClient(name, url));
215 }
216
217 getRegisteredSniffer = url => {
218   return this.registeredSniffers.find(sniffer => sniffer.url == url);
219 }
220
221 // wsClient methods
222 connect = url => {
223   const wsClient = this.wsClients.filter(socket => socket.url == url)[0];
224   if (wsClient) wsClient.connect();
225 }
226
227 disconnect = url => {
228   const wsClient = this.wsClients.filter(socket => socket.url == url)[0];
229   if (wsClient) wsClient.disconnect();
230 }
231
232 updateSnifferStatus = (url, status) => {
233   const sniffer = this.registeredSniffers.filter(sniffer => sniffer.url == url)[0];
234   if (sniffer) sniffer.status = status;
235 }
236
237 startLogs = async () => {
```

```

238     if (this.wsClients.length > 0) {
239         await this.setUpExecutionInfo();
240         this.cleanAllCharts();
241         this.wsClients.forEach(socket => socket.send('start logs'));
242         this.lastCmdToAllWsClients = "start logs";
243     }
244 }
245
246 stopLogs = async () => {
247     if (this.wsClients.length > 0) {
248         this.lastCmdToAllWsClients = "stop logs";
249         this.wsClients.forEach(socket => socket.send('stop logs'));
250         const count = await DbOperations.countRecords();
251         console.log(`count = ${JSON.stringify(count)}`);
252         const execution = await DbOperations.findExecution(this.executionInfo.executionId);
253         execution['name'] = 'new name inserted by user';
254         const date = new Date();
255         execution['endTime'] =
256             ↵ ` ${date.getHours()}:${date.getMinutes()}:${date.getSeconds()}:${date.getMilliseconds()}`;
257         await DbOperations.updateExecution(this.executionInfo.executionId, execution);
258         // const executionInfo = await DbOperations.findExecutionInfo(this.executionInfo.executionId);
259         // console.log(`executionInfo = ${JSON.stringify(executionInfo)}`);
260         // await DbOperations.removeExecution(this.executionInfo.executionId);
261         // const count_after_remove = await DbOperations.countRecords();
262         // console.log(`count_after_remove = ${JSON.stringify(count_after_remove)}`);
263     }
264 }
265
266 setExecutionVideo = async asset => {
267     // salva a referencia do video no banco
268     const execution = await DbOperations.findLastExecutionInfo();
269     execution['videoAsset'] = asset;
270     await DbOperations.updateExecution(execution.id, execution);
271 }
272
273 getExecutionInfo = () => { return this.executionInfo; }
274
275 setUpExecutionInfo = async () => {
276     this.executionInfoReady = false;
277     this.executionInfo = {};
278     const date = new Date();
279     const execution = {
280         name: 'temporary name',
281         initDate: `${date.getUTCFullYear()}-${date.getUTCMonth() + 1}-${date.getUTCDate()}`,
282         initTime: `${date.getHours()}:${date.getMinutes()}:${date.getSeconds()}:${date.getMilliseconds()}`,
283     };
284     this.executionInfo = await DbOperations.createExecution(execution, this.wsClients, this.portChart);
285     this.executionInfoReady = true;
286 }
287
288 printDbExecutionInfo = () => {
289     console.log(`this.executionInfo = ${JSON.stringify(this.executionInfo)}`);
290 }
291
292 getDbPortsIds = wsServerUrl => {
293     // console.log(`on store getDbPortsIds(wsServerUrl = ${wsServerUrl})`);
294     const executionInfo = this.executionInfo;
295     // console.log(`on store this.executionInfo = ${JSON.stringify(executionInfo)}`);
296     if (this.executionInfoReady) {
297         const portsInfo = {};
298         const portsIds = executionInfo.sniffers.find(sniffer => sniffer.wsClientUrl ==
299             ↵ wsServerUrl).portIds;

```

```
298     for (let i = 0; i < portsIds.length; i++) {
299         const port = portsIds[i];
300         portsInfo[port.portName] = { id: port.id, logs: [] }
301     }
302     // console.log(`on store getDbPortsIds return portsInfo = ${JSON.stringify(portsInfo)}`);
303     return portsInfo;
304 } else {
305     return null;
306 }
307 }
308 }
309
310 export default new RegisteredSniffersStore();
```

B.30 Componente de estilo css

```
1 import { StatusBar, StyleSheet } from "react-native";
2
3 export const styles = StyleSheet.create({
4   view: {
5     flex: 1,
6     marginTop: StatusBar.currentHeight || 20,
7   }
8 });
```

B.31 Componente de declaração das rotas das telas do sensor

```
1 import { createDrawerNavigator } from '@react-navigation/drawer';
2 import RegisteredSniffers from '../screens/sniffers/RegisteredSniffers';
3 import Sensores from '../screens/sensores/Sensores';
4 import Videos from '../screens/videos/Videos';
5 import Recording from '../screens/recording/Recording';
6 import ExecutionScreen from '../screens/videos/ExecutionScreen';
7 import PreviewScreen from '../screens/recording/PreviewScreen';
8 import { Ionicons, MaterialIcons } from '@expo/vector-icons';
9
10
11 const { Screen, Navigator } = createDrawerNavigator();
12
13 const defaultIconStyles = {
14   size: 24,
15   color: 'black',
16 }
17
18 export function AppRoutes() {
19   return (
20     <Navigator
21       screenOptions={{
```

```
22     /* unmountOnBlur: true, faz com que as telas sejam que não estão sendo utilizadas sejam destruídas,
23     ↵
24     * economiza recursos mem e cpu, além de ser necessário para garantir o refresh das telas cada vez
25     ↵ que forem acessadas */
26     unmountOnBlur: true,
27     headerShown: false,
28   }}
29   >
30   <Screen
31     name="sniffers"
32     component={RegisteredSniffers}
33     options={{
34       drawerIcon: () => <Ionicons name="ios-hardware-chip-sharp" size={defaultIconStyles.size}
35         ↵ color={defaultIconStyles.color} />
36     }}
37   />
38   <Screen
39     name="sensores"
40     component={Sensores}
41     options={{
42       drawerIcon: () => <MaterialIcons name="insert-chart" size={defaultIconStyles.size}
43         ↵ color={defaultIconStyles.color} />
44     }}
45   />
46   <Screen
47     name="videos"
48     component={Videos}
49     options={{
50       drawerIcon: () => <MaterialIcons name="video-collection" size={defaultIconStyles.size}
51         ↵ color={defaultIconStyles.color} />
52     }}
53   />
54   <Screen
55     name="gravar"
56     component={Recording}
57     options={{
58       drawerIcon: () => <MaterialIcons name="photo-camera" size={defaultIconStyles.size}
59         ↵ color={defaultIconStyles.color} />
60     }}
61   />
62   <Screen
63     name="execution-player"
64     /* drawerItemStyle: { display: 'none' } esconde o ícone de seleção no drawer, isso serve para as
65     ↵ telas que só devem ser acessadas a partir de outras telas */
66     options={{ drawerItemStyle: { display: 'none' } }}
67     component={ExecutionScreen}
68   />
69   <Screen
70     name="execution-preview"
71     options={{ drawerItemStyle: { display: 'none' } }}
72     component={PreviewScreen}
73   />
74   </Navigator >
75 )
76 }
```

B.32 Componente do menu de rotas

```
1 import { TouchableOpacity } from 'react-native';
2 import { StyleSheet } from "react-native";
3
4 import { MaterialIcons } from '@expo/vector-icons';
5 import { styles } from './RoutesMenu.styles';
6
7
8 export function RoutesMenu({ open }) {
9   return (
10     <TouchableOpacity
11       style={styles.touchableOpacity}
12       onPress={open}
13     >
14       <MaterialIcons name="menu" size={24} color={'black'} />
15     </TouchableOpacity>
16   )
17 }
```

B.33 Componente de estilo css

```
1 import { StyleSheet } from "react-native";
2
3
4 export const styles = StyleSheet.create({
5   touchableOpacity: {
6     width: 50,
7     height: 50,
8     alignItems: 'center',
9     justifyContent: 'center',
10    position: 'absolute',
11    left: 15,
12    bottom: 15,
13    backgroundColor: '#1299FA',
14    borderRadius: 5,
15  }
16 });
```

B.34 Componente genérico das telas

```
1 import { RoutesMenu } from './RoutesMenu';
2
3
4 export function ScreenBase({ openRoutesMenu }) {
5   return (
```



```

6     <RoutesMenu open={openRoutesMenu}/>
7   )
8 }

```

B.35 Componente da tela de sensores

```

1  import { View, Button, Dimensions, Text, ScrollView } from "react-native";
2  import { ScreenBase } from "../common/ScreenBase";
3  import { observer, inject } from 'mobx-react';
4  import { StyleSheet } from "react-native";
5  import { ChartCardsList } from '../charts/ChartCardsList';
6
7
8  function Sensores({ navigation, RegisteredSniffersStore }) {
9    const { getAllportChartForChartCardsList, lastCmdToAllWsClients, startLogs, stopLogs,
10     ↪ countLogsRecordsSaved } = RegisteredSniffersStore;
11
12    const viewMargining = 24;
13    const strokeWidth = 16;
14    const screeWidth = Dimensions.get('window').width;
15    const viewWidth = screeWidth - (2 * viewMargining);
16    const drawWidth = viewWidth - (strokeWidth * 2);
17
18    // calculando dimensões
19    const screenDimensions = Dimensions.get('window');
20    const horizontalScrollViewMargin = 10;
21    const axisLabelThickness = 15;
22    const yAxisDimensions = { width: axisLabelThickness, height: 256 + 10 } // height: valor máximo do
23     ↪ sensor de ultrassônico + tamanho do texto dos números do eixo y
24    const canvasWidth = screenDimensions.width - horizontalScrollViewMargin;
25
26    const styles = StyleSheet.create({
27      returnView: {
28        flex: 1,
29        alignItems: 'center',
30        justifyContent: 'center',
31      },
32      ButtonAndScrollView: { marginHorizontal: horizontalScrollViewMargin, marginTop: 24 },
33      Button: {},
34      ScrollView: { marginHorizontal: horizontalScrollViewMargin, marginTop: 8 },
35      Canvas: { height: yAxisDimensions.height, width: canvasWidth },
36      LastCanvas: { height: yAxisDimensions.height, width: canvasWidth, marginBottom: 110 }
37    });
38
39    return (
40      <View style={styles.returnView}>
41        <View style={styles.ButtonAndScrollView}>
42          <Button style={styles.Button} title={lastCmdToAllWsClients == "stop logs" ? "get logs" : "stop
43           ↪ logs"} onPress={() => {
44            if (lastCmdToAllWsClients == "stop logs") {
45              startLogs();
46            } else {
47              stopLogs();
48            }
49          }} />
50        <Text>DataBase has {countLogsRecordsSaved} logs</Text>

```

```

47     <ScrollView>
48       <ChartCardsList sensorConfigsArray={getAllportChartForChartCardsList()} />
49     </ScrollView>
50   </View>
51   <ScreenBase openRoutesMenu={() => navigation.openDrawer()} />
52 </View>
53 );
54 }
55
56 export default inject('RegisteredSniffersStore')(observer(Sensores));

```

B.36 Componente de listagem de sensores

```

1  import { View, Button, Dimensions, Text } from "react-native";
2  import { observer, inject } from 'mobx-react';
3  import { StyleSheet } from "react-native";
4  import { ChartCardsList } from '../charts/ChartCardsList';
5
6
7  function SensoresList({ RegisteredSniffersStore }) {
8     const { getAllportChartForChartCardsList } = RegisteredSniffersStore;
9
10    const viewMargining = 24;
11    const strokeWidth = 16;
12    const screenWidth = Dimensions.get('window').width;
13    const viewWidth = screenWidth - (2 * viewMargining);
14    const drawWidth = viewWidth - (strokeWidth * 2);
15
16    // calculando dimensões
17    const screenDimensions = Dimensions.get('window');
18    const horizontalScrollViewMargin = 10;
19    const axisLabelThickness = 15;
20    const yAxisDimensions = { width: axisLabelThickness, height: 256 + 10 } // height: valor máximo do
    ↪ sensor de ultrassônico + tamanho do texto dos números do eixo y
21    const canvasWidth = screenDimensions.width - horizontalScrollViewMargin;
22
23    const styles = StyleSheet.create({
24      ButtonAndScrollView: { marginHorizontal: horizontalScrollViewMargin, marginTop: 24 },
25      Button: {},
26      ScrollView: { marginHorizontal: horizontalScrollViewMargin, marginTop: 8 },
27      Canvas: { height: yAxisDimensions.height, width: canvasWidth },
28      LastCanvas: { height: yAxisDimensions.height, width: canvasWidth, marginBottom: 110 }
29    });
30    return (
31      <View style={styles.ButtonAndScrollView}>
32        <ChartCardsList sensorConfigsArray={getAllportChartForChartCardsList()} />
33      </View>
34    );
35  }
36
37  export default inject('RegisteredSniffersStore')(observer(SensoresList));

```

B.37 Funções de manipulação da tabelas de sniffers

```

1  import db from "../DataBase";
2
3
4  const tableName = "sniffers";
5
6  /**
7   * INICIALIZAÇÃO DA TABELA
8   * - Executa sempre, mas só cria a tabela caso não exista (primeira execução)
9   */
10 const init = async () => {
11   await new Promise((resolve, reject) => {
12     db.transaction((tx) => {
13       tx.executeSql(
14         `CREATE TABLE IF NOT EXISTS ${tableName} (
15           id INTEGER PRIMARY KEY AUTOINCREMENT,
16           name TEXT,
17           wsClientUrl TEXT,
18           executionId INTEGER,
19           CONSTRAINT executionId FOREIGN KEY (executionId)
20             REFERENCES executions(id)
21             ON DELETE CASCADE);`,
22         [],
23         (_, { rowsAffected, insertId }) => resolve(console.log(`created ${tableName} table, rowsAffected
24         ↵ = ${rowsAffected}, insertId = ${insertId}`)),
25         (_, error) => {
26           console.log(`could not create ${tableName} table`);
27           reject(error) // erro interno em tx.executeSql
28         }
29       );
30     });
31   })
32 }
33
34 const appendSnifferOnExecution = (sniffer, executionId) => {
35   return new Promise((resolve, reject) => {
36     db.transaction((tx) => {
37       //comando SQL modificável
38       tx.executeSql(
39         `INSERT INTO ${tableName} (name, wsClientUrl, executionId) values (?, ?, ?)`,
40         [sniffer.name, sniffer.wsClientUrl, executionId],
41         //-----
42         (_, { rowsAffected, insertId }) => {
43           if (rowsAffected > 0) {
44             console.log(`created execution with id = ${insertId}`);
45             resolve(insertId);
46           }
47           else reject(`Error inserting execution: ${JSON.stringify(execution)}`); // insert falhou
48         },
49         (_, error) => reject(error) // erro interno em tx.executeSql
50       );
51     });
52   });
53 }
54 /**
55 * BUSCA TODOS OS REGISTROS DE UMA DETERMINADA TABELA
56 * - Não recebe parâmetros;

```

```
57 * - Retorna uma Promise:
58 * - O resultado da Promise é uma lista (Array) de objetos;
59 * - Pode retornar erro (reject) caso o ID não exista ou então caso ocorra erro no SQL;
60 * - Pode retornar um array vazio caso não existam registros.
61 */
62 const getSniffersFromExecution = executionId => {
63   return new Promise((resolve, reject) => {
64     db.transaction((tx) => {
65       //comando SQL modificável
66       tx.executeSql(
67         `SELECT * FROM ${tableName} WHERE executionId = ${executionId};`,
68         [],
69         //-----
70         (_, { rows }) => {
71           console.log(`got sniffers from executionId = ${executionId}`);
72           resolve(rows._array);
73         },
74         (_, error) => reject(error) // erro interno em tx.executeSql
75       );
76     });
77   });
78 };
79
80 const deleteAllRecords = () => {
81   return new Promise((resolve, reject) => {
82     db.transaction((tx) => {
83       //comando SQL modificável
84       tx.executeSql(
85         `DELETE FROM ${tableName};`,
86         [],
87         //-----
88         (_, { rowsAffected }) => {
89           resolve(rowsAffected);
90         },
91         (_, error) => reject(error) // erro interno em tx.executeSql
92       );
93     });
94   });
95 };
96
97 const countRecords = async () => {
98   return await new Promise((resolve, reject) => {
99     db.transaction((tx) => {
100       //comando SQL modificável
101       tx.executeSql(
102         `SELECT COUNT(*) FROM ${tableName};`,
103         [],
104         //-----
105         (_, { rows }) => resolve(rows._array[0]["COUNT(*)"]),
106         (_, error) => reject(error) // erro interno em tx.executeSql
107       );
108     });
109   });
110 };
111
112 const findSniffers = async (executionId) => {
113   return await new Promise((resolve, reject) => {
114     db.transaction((tx) => {
115       //comando SQL modificável
116       tx.executeSql(
117         `SELECT * FROM ${tableName} WHERE executionId = ${executionId};`,
```

```

118     [],
119     //-----
120     (_, { rows }) => {
121       console.log(`find ${tableName} rows = ${JSON.stringify(rows)}`);
122       resolve(rows._array)
123     },
124     (_, error) => reject(error) // erro interno em tx.executeSql
125   );
126 });
127 });
128 }
129
130 export default {
131   tableName,
132   init,
133   deleteAllRecords,
134   appendSnifferOnExecution,
135   countRecords,
136   getSniffersFromExecution,
137   findSniffers
138 };

```

B.38 Componentes de stilo css

```

1 import {
2   Platform,
3   StyleSheet,
4   Text,
5   TextStyle,
6   TouchableNativeFeedback,
7   TouchableNativeFeedbackProps,
8   TouchableOpacity,
9   TouchableOpacityProps,
10  View,
11 } from 'react-native'
12 import React from 'react'
13
14 export const ErrorMessage = ({ message, style }: { message: string; style: TextStyle }) => (
15   <View style={styles.errorWrapper}>
16     <Text style={style}>{message}</Text>
17   </View>
18 )
19
20 export const getMinutesSecondsFromMilliseconds = (ms: number) => {
21   const totalSeconds = ms / 1000
22   const seconds = String(Math.floor(totalSeconds % 60))
23   const minutes = String(Math.floor(totalSeconds / 60))
24
25   return minutes.padStart(1, '0') + ':' + seconds.padStart(2, '0')
26 }
27
28 type ButtonProps = (TouchableNativeFeedbackProps | TouchableOpacityProps) & {
29   children: React.ReactNode
30 }
31 export const TouchableButton = (props: ButtonProps) =>

```

```
32 Platform.OS === 'android' ? (  
33   <TouchableNativeFeedback  
34     background={TouchableNativeFeedback.Ripple('white', true)}  
35     {...props}  
36   />  
37 ) : (  
38   <TouchableOpacity {...props} />  
39 )  
40  
41 // https://gist.github.com/ahtcx/0cd94e62691f539160b32ecda18af3d6#gistcomment-3585151  
42 // eslint-disable-next-line @typescript-eslint/no-explicit-any  
43 export const deepMerge = (target: { [x: string]: any }, source: { [x: string]: any }) => {  
44   const result = { ...target, ...source }  
45   const keys = Object.keys(result)  
46  
47   for (const key of keys) {  
48     const tprop = target[key]  
49     const sprop = source[key]  
50     if (typeof tprop === 'object' && typeof sprop === 'object') {  
51       result[key] = deepMerge(tprop, sprop)  
52     }  
53   }  
54  
55   return result  
56 }  
57 export const styles = StyleSheet.create({  
58   errorWrapper: {  
59     ...StyleSheet.absoluteFillObject,  
60     paddingHorizontal: 20,  
61     justifyContent: 'center',  
62   },  
63   videoWrapper: {  
64     flex: 1,  
65     justifyContent: 'center',  
66   },  
67   iconWrapper: {  
68     borderRadius: 100,  
69     overflow: 'hidden',  
70     padding: 10,  
71   },  
72   bottomInfoWrapper: {  
73     position: 'absolute',  
74     flexDirection: 'row',  
75     alignItems: 'center',  
76     justifyContent: 'space-between',  
77     flex: 1,  
78     bottom: 0,  
79     left: 0,  
80     right: 0,  
81   },  
82   topInfoWrapper: {  
83     position: 'absolute',  
84     flexDirection: 'row',  
85     alignItems: 'center',  
86     justifyContent: 'space-between',  
87     flex: 1,  
88     top: 0,  
89     left: 0,  
90     right: 0,  
91     zIndex: 999,  
92   },
```

```
93   timeLeft: { backgroundColor: 'transparent', marginLeft: 5 },
94   timeRight: { backgroundColor: 'transparent', marginRight: 5 },
95   slider: { flex: 1, paddingHorizontal: 10 },
96 })
```

B.39 Componente de menu das execuções

```
1  import { View, Text, Image, StyleSheet, ScrollView, TouchableOpacity } from "react-native";
2  import { ScreenBase } from "../common/ScreenBase";
3  import { useEffect, useState } from "react";
4  import DbOperations from "../database/DbOperations";
5  import * as VideoThumbnails from 'expo-video-thumbnails';
6  import { useNavigation } from '@react-navigation/native';
7  import { MaterialIcons, AntDesign } from '@expo/vector-icons';
8  import * as MediaLibrary from 'expo-media-library';
9
10
11  async function deleteAsset(videoAsset) {
12    try {
13      // Request permission to access the media library
14      const { status } = await MediaLibrary.requestPermissionsAsync();
15
16      if (status === 'granted') {
17        await MediaLibrary.deleteAssetsAsync([videoAsset]);
18        console.log('Assets deleted successfully.');
```

```
19      } else {
20        console.log('Permission to access the media library was not granted.');
```

```
21      }
22    } catch (error) {
23      console.log('An error occurred while deleting assets:', error);
24    }
25  }
26
27  function Line() {
28    return (
29      <View style={{
30        borderBottomColor: 'black',
31        borderBottomWidth: 0.5,
32        width: 70,
33        marginBottom: 2
34      }} />
35    )
36  }
37
38  function ExecutionMenu({ executionId, videoAsset, removeExecution }) {
39    const [menuIsOpen, setMenuIsOpen] = useState(false);
40
41    return (
42      <>
43        {menuIsOpen ?
44          <View style={styles.executionMenuOptionsList}>
45            <TouchableOpacity style={styles.executionMenuItem} onPress={() => {
46              DbOperations.removeExecution(executionId);
47              removeExecution(executionId);
48              deleteAsset(videoAsset);
```

```

49     </Text></Text>
50     </TouchableOpacity>
51     <Line />
52     <TouchableOpacity style={styles.executionMenuItem} onPress={() => setMenuIsOpen(false)}>
53       <AntDesign name="closecircle" size={20} color={'black'} />
54     </TouchableOpacity>
55   </View> :
56   <TouchableOpacity style={styles.executionMenu} onPress={() => setMenuIsOpen(true)}>
57     <MaterialIcons name="menu" size={20} color={'black'} />
58   </TouchableOpacity>
59 </>
60 );
61 }
62 }
63
64 function Execution({ id, name, initDate, videoAsset, removeExecution }) {
65   const navigation = useNavigation();
66   // 'file:///storage/emulated/0/DCIM/1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4'
67   const [thumbnailImageUri, setThumbnailImageUri] = useState('');
68
69   useEffect(() => {
70     (async () => {
71       if (videoAsset?.uri) {
72         try {
73           const { uri } = await VideoThumbnails.getThumbnailAsync(videoAsset?.uri);
74           setThumbnailImageUri(uri);
75         } catch (e) {
76           console.warn(e);
77         }
78       }
79     })();
80   }, []);
81
82   return (
83     <TouchableOpacity style={styles.execution} onPress={() => {
84       navigation.navigate('execution-player', { executionId: id });
85     }}>
86     <View style={styles.noThumbnailView}>
87       {thumbnailImageUri ? <Image source={{ uri: thumbnailImageUri }} style={styles.thumbnail} /> :
88       ↪ <Text>no thumbnail available</Text>}
89     <ExecutionMenu executionId={id} removeExecution={removeExecution} videoAsset={videoAsset} />
90   </View>
91 </TouchableOpacity>
92 );
93 }
94
95 export default function Videos({ navigation }) {
96   /* allExecutions = [{
97     "id":2,
98     "name":"temporary name",
99     "initDate":"2023-5-1",
100    "initTime":"20:40:16:483",
101    "endTime":"",
102    "videoAsset":{
103      "mediaType": "video",
104      "modificationTime": 1686517909000,
105      "uri": "file:///storage/emulated/0/DCIM/1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4",
106      "filename": "1e37dd68-3a55-462e-9a66-7d2c7dcc77d2.mp4",
107      "width": 1080,
108      "id": "1000010523",

```



```

109     "creationTime": 1686517904000,
110     "albumId": "-2075821635",
111     "height": 1920,
112     "duration": 7.783
113   }
114 }]*/*
115 const [allExecutions, setAllExecutions] = useState([]);
116 const removeExecution = executionId => {
117   setAllExecutions(allExecutions.filter(execution => execution.id !== executionId));
118 }
119
120 useEffect(() => {
121   (async () => {
122     const executions = await DbOperations.getAllExecutions();
123     setAllExecutions(executions);
124   })();
125 }, []);
126
127 return (
128   <View style={styles.view}>
129     {allExecutions.length > 0 ?
130       <ScrollView style={styles.scrollView}>
131         <View style={styles.scrollViewInternalViewToPutItemsSideBySide}>
132           {allExecutions.map(execution => {
133             console.log(`execution = ${JSON.stringify(execution)}`);
134             return <Execution {...execution} removeExecution={removeExecution} key={execution.id} />;
135           })}
136         </View>
137       </ScrollView>
138       : <Text>no executions found</Text>}
139     <ScreenBase openRoutesMenu={() => navigation.openDrawer()} />
140   </View>
141 )
142 }
143
144 const styles = StyleSheet.create({
145   view: {
146     flex: 1,
147     padding: 30,
148     alignItems: 'center',
149     justifyContent: 'center',
150   },
151   scrollView: {},
152   scrollViewInternalViewToPutItemsSideBySide: { flex: 1, flexDirection: "row", flexWrap: "wrap" },
153   execution: { width: 150, height: 150, marginBottom: 10, marginLeft: 10 },
154   thumbnail: { width: '100%', height: 150, borderRadius: 10 },
155   noThumbnailView: {
156     flex: 1, alignItems: 'center', justifyContent: 'center',
157     backgroundColor: '#d9d9d9',
158     shadowOpacity: 0.8,
159     borderRadius: 10,
160   },
161   executionMenu: {
162     position: "absolute",
163     right: 7,
164     top: 7,
165     width: 30,
166     height: 30,
167     backgroundColor: 'white',
168     borderRadius: 5,
169     alignItems: 'center',

```


```
170     justifyContent: 'center',
171   },
172   executionMenuOptionsList: {
173     position: "absolute",
174     right: 7,
175     top: 7,
176     width: 80,
177     height: 50,
178     backgroundColor: 'white',
179     borderRadius: 5,
180     alignItems: 'center',
181     justifyContent: 'center',
182   },
183   executionMenuItem: { height: 22 },
184 });
```

B.40 Objeto de conexão cliente WebSocket

```
1 import RegisteredSniffersStore from '../stores/sniffers/RegisteredSniffers.store';
2 import DbOperations from '../database/DbOperations';
3
4
5 class WsClient {
6   name = '';
7   url = '';
8   ws = null;
9   logsBuffer = {};
10
11   detDbInfoThread = null;
12   database = null;
13   // dbLogsBuffer = {
14   //   port1: {
15   //     id: 'id do banco',
16   //     logs: []
17   //   },
18   //   port2: {
19   //     id: 'id do banco',
20   //     logs: []
21   //   }
22   // };
23   dbLogsBuffer = {};
24   dbLogsBufferTimer = 10000;
25   dbLastSaveTime = new Date().getTime(); // tempo em ms da ultima vez em que salvou os logs no banco
26
27   constructor(name, url) {
28     this.name = name;
29     this.url = url;
30   }
31
32   isConnected = () => {
33     return this.getStatusString() == 'OPEN' ? true : false;
34   }
35
36   getUrl = () => { return this.url };
37
```

```
38   getStatusString = () => {
39     const connectionStates = {
40       0: 'CONNECTING',
41       1: 'OPEN',
42       2: 'CLOSING',
43       3: 'CLOSED',
44     }
45
46     if (this.ws) return connectionStates[this.ws.readyState];
47     else return 'There is no websocket'
48   }
49
50   send = cmd => {
51     if (cmd == "start logs") {
52       this.setToSaveLogs();
53     } else if (cmd == "stop logs") {
54       this.saveLogs(true);
55       this.resetToSaveLogs();
56     }
57     this.ws.send(cmd);
58   }
59
60   resetToSaveLogs = () => {
61     this.dbLogsBuffer = {};
62   }
63
64   setToSaveLogs = () => {
65     const { printDbExecutionInfo } = RegisteredSniffersStore;
66     printDbExecutionInfo();
67     const { getDbPortsIds } = RegisteredSniffersStore;
68     this.dbLogsBuffer = getDbPortsIds(this.getUrl());
69     this.dbLastSaveTime = new Date().getTime();
70   }
71
72
73   bufferDbLogs = logs => {
74     const ports = Object.keys(this.dbLogsBuffer);
75     for (let i = 0; i < ports.length; i++) {
76       const port = ports[i];
77       this.dbLogsBuffer[port].logs = [...this.dbLogsBuffer[port].logs, ...logs[port]];
78     }
79   }
80
81   saveLogs = (onStopLogs = false) => {
82     const actualTime = new Date().getTime();
83     if (onStopLogs || actualTime - this.dbLastSaveTime > this.dbLogsBufferTimer) {
84       const ports = Object.keys(this.dbLogsBuffer);
85       for (let i = 0; i < ports.length; i++) {
86         const key = ports[i];
87         const port = this.dbLogsBuffer[key];
88         DbOperations.appendLogsOnPort(this.dbLogsBuffer[key].logs.splice(0), port.id);
89       }
90       this.dbLastSaveTime = new Date().getTime();
91     }
92   }
93
94   connect = () => {
95     if (this.ws) this.ws = null;
96     this.ws = new WebSocket(this.url);
97
98     this.ws.onopen = open => {
```

```
99     const { updateSnifferStatus } = RegisteredSniffersStore;
100     this.send("ports");
101     updateSnifferStatus(this.url, 'conectado');
102   }
103
104   this.ws.onclose = close => {
105     const { updateSnifferStatus } = RegisteredSniffersStore;
106     updateSnifferStatus(this.url, 'desconectado');
107   }
108
109   this.ws.onerror = error => {
110     this.ws.close();
111   }
112
113   this.ws.onmessage = message => {
114     const { connectedPorts, logs } = JSON.parse(message.data);
115     if (logs) {
116       const ports = Object.keys(logs);
117       for (let i = 0; i < ports.length; i++) {
118         this.logsBuffer[ports[i]] = this.logsBuffer[ports[i]] ? [...this.logsBuffer[ports[i]],
119           ↵ ...logs[ports[i]]] : [...logs[ports[i]]];
120       }
121
122       this.bufferDbLogs(logs);
123       this.saveLogs();
124     }
125     else if (connectedPorts) {
126       // { connectedPorts: ["port1", "port2"] };
127       const { registerConnectedPorts } = RegisteredSniffersStore;
128       registerConnectedPorts(this.getUrl(), connectedPorts);
129     }
130   }
131
132   disconnect = () => {
133     if (this.ws) this.ws.close();
134   }
135
136   getLogs = (logsQuantByPort = 1) => {
137     const ports = Object.keys(this.logsBuffer);
138     let logs = {};
139     for (let i = 0; i < ports.length; i++) {
140       logs[ports[i]] = this.logsBuffer[ports[i]].splice(0, logsQuantByPort);
141     }
142     return logs;
143   }
144
145   setLogsBufferPort = portName => {
146     this.logsBuffer[portName] = this.logsBuffer[portName] ? [...this.logsBuffer[portName]] : [];
147   }
148
149   removeLogsBufferPort = portName => {
150     delete this.logsBuffer[portName];
151   }
152 }
153
154 export default WsClient;
```

	INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
	Campus Campina Grande
	R. Tranquílino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
	CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

TCC

Assunto:	TCC
Assinado por:	Judenilson Araujo
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Judenilson Araujo Silva, ALUNO (201811250050) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 25/01/2024 11:22:14.

Este documento foi armazenado no SUAP em 25/01/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifpb.edu.br/verificar-documento-externo/> e forneça os dados abaixo:

Código Verificador: 1061996

Código de Autenticação: ec6f07ce1e

