UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR)
CURSO DE ENGENHARIA DE COMPUTAÇÃO

CESAR AUGUSTO PEREIRA VIAL
EDUARDO PIACENY RIBAS
GUSTAVO VALENTE GULMINE

AR DRUMS

OFICINA DE INTEGRAÇÃO 2 – RELATÓRIO FINAL

CURITIBA

2022

CESAR AUGUSTO PEREIRA VIAL
EDUARDO PIACENY RIBAS
GUSTAVO VALENTE GULMINE

AR DRUMS

Relatório Final da disciplina Oficina de Integração 2, do curso de Engenharia de Computação, apresentado aos professores que ministram a mesma na Universidade Tecnologica Federal do Paraná como requisito parcial para obtenção da aprovação na disciplina.

Orientador: Prof. Dr. César Manuel Vargas Benítez
Prof. Dr. Heitor S. Lopes

CURITIBA

2022

RESUMO

. AR DRUMS. 55 f. Oficina de Integração 2 – Relatório Final – Curso de Engenharia de Computação, UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR). Curitiba, 2022.

Neste relatório, o projeto dos ARDrums será apresentado. É um projeto que visa criar opção para o transporte e acessibilidade para baterias. Utilizando microcontroladores, múlitplos métodos de comunicação, processamento de vídeo e áudio, foi criada uma simulação, parcialmente customizável via aplicativo, de uma bateria, necessitando apenas duas baquetas, dois pedais, e alguns pequenos circuitos para operar. Todos os componentes e métodos utilizados para a realização deste projeto serão apresentados. Por fim, o projeto apresentou resultados satisfatórios que atendem à grande maioria de suas especificações e objetivos.

Palavras-chave: música, bateria, imagem

ABSTRACT

. AR DRUMS. 55 f. Oficina de Integração 2 – Relatório Final – Curso de Engenharia de Computação, UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR). Curitiba, 2022.

In this report, the ARDrums project will be presented. This is a project with the objective to create an option for the transportation and accessibility of drums. Using microcontrolers, multiple communication methods, video and audio processing, the project creates a simulation of drums, partially customizable via an application, only needing two drumsticks, two pedals, and some small circuits to operate. All the components and methods used for this project will be presented. Finally, the project showed satisfactory results that attend to most of its especifications and objectives.

Keywords: music, drums, image

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Em se tratando de apresentações musicais, um grande problema é o transporte de certos instrumentos. Enquanto alguns são pequenos e podem ser transportados em uma capa própria (como violinos e violões), instrumentos de grande porte, como baterias e pianos são de difícil transporte, além de serem extremamente caros, dificultando o acesso a tais instrumentos por iniciantes. Visando facilitar o acesso a uma bateria, assim como o seu transporte, o projeto desenvolvido mostra uma alternativa a baterias acústicas e eletrônicas, sendo mais acessível e ocupando muito menos espaço.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

O objetivo geral dos AR Drums é ser um conjunto de componentes que, ao utilizados em conjunto, permitem que o usuário seja capaz de tocar uma bateria fazendo apenas movimentos no ar. Dessa forma o usuário teria apenas o que é preciso para realizar o processamento de seus movimentos e tocar sons de acordo, podendo customizar tais sons por meio de um aplicativo. Essa abordagem faz com que os AR Drums ocupem muito menos espaço e, por possuírem uma estrutura reduzida, com poucos componentes, tenham também um custo menor do que aquele de uma bateria acústica ou eletrônica, facilitando tanto o acesso para pessoas que queiram iniciar seus estudos com a bateria, quanto o transporte deste instrumento, caso necessário.

1.2.2 OBJETIVOS ESPECÍFICOS

Os AR Drums possuem os seguintes objetivos específicos:

- Simular o som de uma bateria;

- Ter duas baquetas nas quais seus movimentos e posições poderão ser traduzidos em sons;
- Ter dois pedais nos quais o pressionamento poderá ser traduzido em sons;
- Ter um aplicativo para que o usuário possa:
  - Visualizar a configuração atual da bateria, sabendo quais sons podem ser tocados pelas baquetas e pelos pedais;
  - Modificar a configuração da bateria, alterando os sons que poderão ser tocados pelas baquetas e pelos pedais;
  - Verificar o mapeamento das posições de cada peça da bateria.

2 FUNDAMENTAÇÃO TEÓRICA

Para a realização do projeto, vários tópicos de outras disciplinas ou externos a qualquer disciplina foram estudados e utilizados. As seções deste capítulo explicam os principais tópicos estudados.

2.1 RASBERRY PI 2 MODELO B

Para realizar todo o processamento de vídeo e áudio necessário, foi utilizado o computador reduzido Rasberry Pi 2 Modelo B. Essa placa possui um processador *Broadcom BCM2836 quad core Cortex A7*, com um clock de 900 MHz, 1 GB de memória RAM, GPU VideoCore IV, porta HDMI, saída AV via 3.5mm *jack*, interface Ethernet 10/100M, 4x portas USB 2.0 e 1x porta micro USB para alimentação, além de uma enorme quantidade de GPIOs (SUEIRO, 2015). É importante notar que este modelo da Raspberry Pi não possui um módulo Wi-Fi. A figura 1 mostra todos os pinos, entradas e saídas deste modelo.

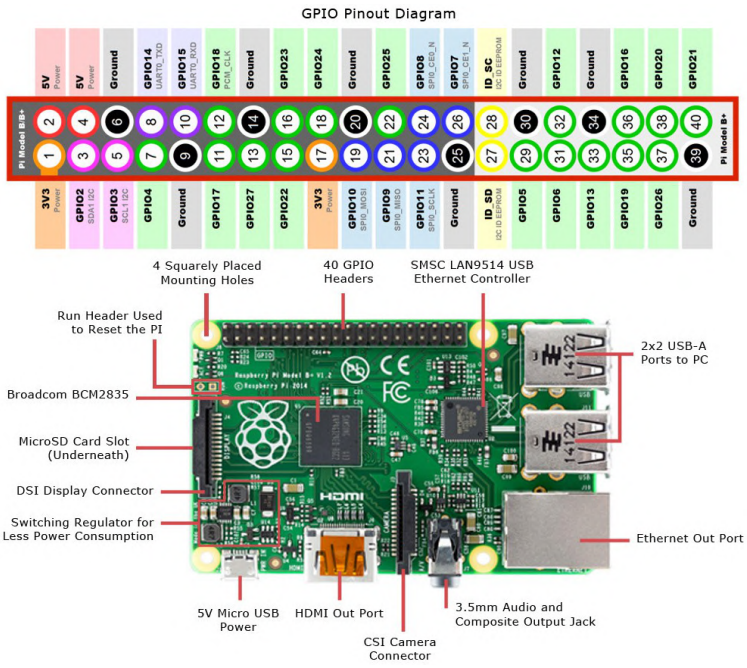

Figura 1: Componentes da Raspberry Pi 2 modelo B.

Um fator importante que vale ressaltar sobre a Rasberry Pi 2 Modelo B, é que os seus núcleos possuem apenas uma *Thread* física, ou seja, ela possui um total de 4 *Threads* físicas. Outro fator importante sobre a Raspberry Pi, é que como ela é um computador que opera sob um sistema operacional *Linux*, várias linguagens de programação são suportadas, incluindo *Python*.

2.1.1 CAMERA MODULE V1

Para o processamento de vídeo em tempo real, foi utilizado um módulo de câmera oficial da Raspberry Pi, o *Camera Module v1* (RASPBERRY, 2013). Essa câmera pode captar até 90 quadros por segundo, e possui uma enorme variedade de resoluções. Tal variedade de resolução é de grande ajuda no projeto, permitindo a escolha de uma resolução menor para aumentar o desempenho.

2.2 ESP 32

A placa de desenvolvimento ESP32 Devkit V1 da DOIT, cuja pinagem se encontra ilustrada na figura 2 (RANDOMNERDTUTORIALS, 2020), foi selecionada para uso no projeto dado sua extensa lista de funcionalidades, aliadas à praticidade de sua programação através da IDE Arduino e de seu custo reduzido. Existindo em modelos de 30 e 36 pinos, com o modelo de 30 sendo utilizado para o projeto, esse microcontrolador (EXPRESSIF, 2022) possui um processador *dual core* capaz de operar a frequências de até 240MHz, além de módulos *Bluetooth* e Wi-Fi integrados, suporte a diversos protocolos de comunicação serial, como I²C e SPI, e dois conversores analógicos-digitais com resolução de 12-bits, capazes de operar em conjunto em até 18 canais diferentes.

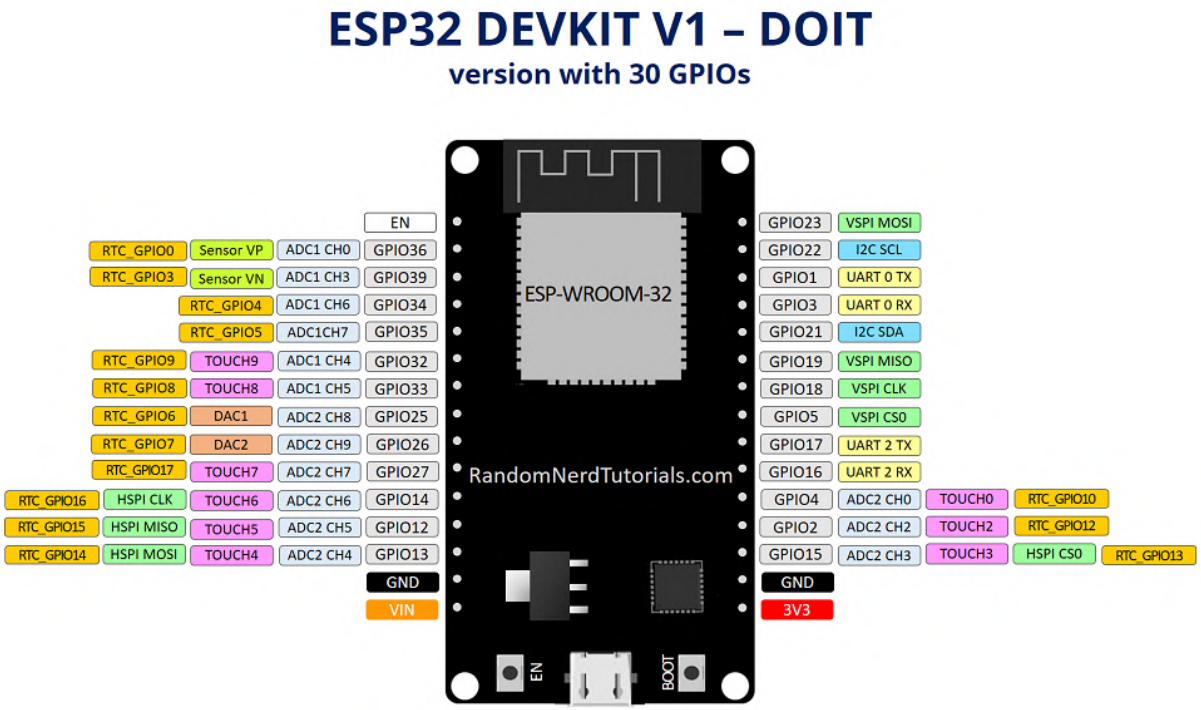

Figura 2: Ilustração e pinagem do módulo DOIT ESP32 Devkit V1.

Um ponto relevante a ressaltar é a versatilidade do módulo WiFi presente na ESP32, que permite que o microcontrolador seja programado para atuar tanto como uma estação conectada à uma rede, quanto como um ponto de acesso, algo extremamente relevante no contexto desse projeto, conforme melhor explicado na seção 2.6.

2.3 TRANSDUTOR PIEZOELÉTRICO

O transdutor piezoelétrico é um sensor capaz de detectar toques em sua superfície. As vibrações realizadas pelo toque em sua superfície emitem um sinal elétrico, dependendo da intensidade de tais vibrações. Ele não possui uma precisão alta, porém um rápido tempo de resposta. Sua tensão de saída varia de 0V a 10V (FERREIRA, 2017). A figura 3 mostra uma foto desses transdutores.

Figura 3: Transdutor piezoelétrico.

2.4 ACELERÔMETRO - MPU6050

Os módulos MPU6050 (Figura 4) agrupam, em uma única estrutura, um conjunto amplo de sensores, contendo um acelerômetro e um giroscópio, ambos operando em 3 dimensões, além de um sensor de temperatura. A interação com um microcontrolador é realizada através de um canal de comunicação I²C, com o endereço sendo configurado através do pino AD0, e permitindo também a detecção de novas leituras através do pino INT, que gera interrupções conforme a configuração realizada (VIDADESILíCIO, 2020).

Figura 4: Módulo MPU6050.

Esses módulos serão utilizados no projeto para a detecção de batidas, o que é feito através da leitura das informações obtidas pelos acelerômetros. A leitura realizada pelos acelerômetros, conforme indicado pelo datasheet do componente (INVENSENSE, 2013), diz respeito à aceleração resultante sobre os três corpos de prova presentes no acelerômetro, cada um com movimento livre em um eixo. Por se tratar da aceleração resultante, todas as leituras sempre estarão acrescidas da própria aceleração da gravidade ($1g$). É possível ainda configurar as margens de leitura para os valores de $\pm 2g$, $\pm 4g$, $\pm 8g$ ou $\pm 16g$, conforme a necessidade,

com as leituras advindas de todos os sensores sendo enviadas, na escala configurada, através do canal I²C.

2.5 COMUNICAÇÃO SERIAL

Comunicação serial é uma parte importante do projeto, já que é dessa maneira que a Raspberry irá receber os dados necessários para seu funcionamento. A comunicação serial utiliza do protocolo UART (atrelada ao uso de um conversor UART para USB), um protocolo de comunicação *full-duplex* ponto-a-ponto, que utiliza de dois canais de comunicação, o canal TX, de envio de dados, e o canal RX, de recebimento de dados. Para que os dados enviados não sejam perdidos, é necessário definir um *baud rate*, em bits por segundo, que deve ser o mesmo para ambos os equipamentos comunicantes. Enquanto dados não são enviados, o sinal fica em nível lógico alto.

Os dados são enviados em pacotes de bytes, acrescidos de sufixos e prefixos de controle, sendo que a ordenação dos bits pode ser em *little endian*, no qual os bits são ordenados do menos significativo ao mais significativo, ou em *big endian*, no qual os bits são ordenados do mais significativo ao menos significativo (SWAROOP, 2018).

2.6 COMUNICAÇÃO POR WIFI

Uma forma prática de realizar a comunicação de múltiplos equipamentos diferentes é através do uso de uma rede local. No caso dos AR Drums, há a necessidade de que tal tipo de comunicação seja ainda realizada sem o uso de fios. Dessa forma, uma opção é o uso de uma rede sem fio local (WLAN), o que pode ser feito ao se criar um ponto de acesso local, e conectando todos os dispositivos que integram o sistema a este ponto de acesso. Por se tratar ainda de uma rede, é possível facilitar a comunicação entre os dispositivos através do uso de protocolos de comunicação em rede, como o UDP ou o TCP.

Para a implementação desse tipo de comunicação no projeto, será realizada a conexão a um ponto de acesso gerado por uma ESP32, conforme melhor apresentado na seção 2.6.1, com a comunicação entre os componentes sendo realizada com o auxílio do protocolo UDP, apresentado na seção 2.6.2.

2.6.1 WLAN ATRAVÉS DE UM PONTO DE ACESSO

Através de bibliotecas nativas do Arduino (ARDUINO, 2022) e da Expressif (EXPRESSIF, 2021), é possível configurar uma ESP32 para atuar como ponto de acesso Wi-Fi. Com isso, é possível conectar quaisquer outros equipamentos que possuam conexão Wi-Fi a este ponto de acesso, estabelecendo uma rede local. No caso de, por exemplo, outra ESP32, isso pode ser feito através do uso das mesmas bibliotecas nativas. Já no caso de outros equipamentos, como um telefone celular, tal conexão pode ser realizada da mesma maneira como comumente é feita a conexão a redes Wi-Fi. A partir do momento em que todos os equipamentos se encontram conectados ao mesmo ponto de acesso, se torna possível o envio de mensagens entre eles.

2.6.2 PROTOCOLO UDP

Conforme explicado em (KUROSE; ROSS, 2021), o protocolo UDP (*User Datagram Protocol*) é um protocolo de camada de transporte utilizado para enviar mensagens, chamadas de datagramas, contendo a informação a ser enviada, juntamente do endereço IP e de porta do destinatário. Essas informações constituem o mínimo necessário para a implementação de um sistema com comunicação via *sockets*, que podem ser definidos como uma estrutura que interfacea as camadas de aplicação e de transporte de uma aplicação seguindo o modelo OSI. Por conta dessa estrutura simples, o UDP não estabelece conexão, nem oferece nenhum tipo de garantia no envio das mensagens, servindo apenas como um serviço auxiliar para o endereçamento. Apesar disso, é justamente essa estrutura simples que garante que o envio das informações seja muito rápido, em comparação ao tradicional protocolo TCP. Para o caso de um sistema como dos AR Drums, o uso de tal tipo de protocolo é interessante, uma vez que tenderia a reduzir o tempo de comunicação e, como consequência, aumentar a responsividade.

2.7 PROCESSAMENTO DE ÁUDIO

Para tocar os sons da bateria corretamente, é necessário processar os diferentes instrumentos que serão utilizados, a fim de que esses instrumentos possam ser tocados independentemente de outros sons, e para que eles possam ser tocados simultaneamente. Para isso, foi utilizada a bibliotecca *pyaudio* (PHAM, 2006), em conjunto com arquivos WAVE possuindo os exemplos de áudio de cada peça existente na bateria.

2.7.1 ARQUIVOS WAVE

Arquivos WAVE (ou WAV) são arquivos em formato padrão de áudio da *Microsoft* e da IBM. Sua versão mais comum guarda os conteúdos do arquivo em formato PCM (modulação de pulsos), fazendo com que tenha perda mínima no conteúdo do arquivo, e que eles sejam facilmente editáveis (WIKIPEDIA, 2008). Os arquivos com os áudios utilizados para a simulação da bateria estão neste formato. Como a linguagem utilizada nos códigos de processamento de áudio será *Python*, uma biblioteca nativa da linguagem poderá ser utilizada para o tratamento desses arquivos, a biblioteca *wave* (PYTHON, 2020), facilitando a manipulação desses com funções como a modificação da amplitude e a separação do arquivo em conjuntos de bytes.

2.7.2 PYAUDIO

A biblioteca utilizada para manipular múltiplos arquivos WAVE simultaneamente é a *pyaudio*. Essa biblioteca permite que cada byte do arquivo seja tocado por vez, sendo facilmente manipulável para tocar apenas uma parte desejada de um arquivo. Essa biblioteca necessita de uma saída de áudio para seu processamento, entrando aqui a saída AV via 3.5mm presente na Raspberry Pi.

2.8 PROCESSAMENTO DE VÍDEO

Para o processamento de vídeo do projeto, será utilizada a biblioteca OpenCV para *python* (OPENCV, 2022). Para o processamento de vídeo, é necessário aplicar máscaras na imagem, para detectar a cor correta. Máscaras são aplicadas em imagens para focar apenas naquilo que é desejado ver nela. No caso deste projeto, as máscaras serão aplicadas para detectar uma certa cor codificada em HSV, um código que utiliza 3 bytes para deifinir uma cor, em que cada byte tem o seguinte significado:

- H (*Hue* ou Tonalidade): indica a cor do pixel;
- S (*Saturation* ou Saturação): indica a intensidade da cor do pixel;
- V (*Value* ou Valor): indica a claridade da cor do pixel,

A figura 5 mostra todos os valores possíveis utilizando a codificação HSV.

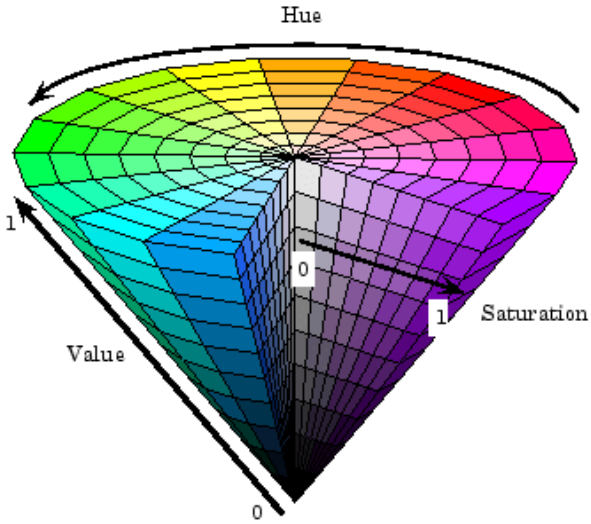

Figura 5: Esquema indicando a codificação de cores HSV (WPILIB, 2022).

2.9 PROGRAMAÇÃO REATIVA

Programação reativa pode ser descrita como um paradigma que conta com uma lógica de programação assíncrona para tratar atualizações em tempo real. Isto é, a programação reativa utiliza fluxos de dados para cuidar de atualizações de conteúdos sempre que há interação com o usuário (NOLLE, 2021). Dito isso, os fluxos utilizados nesse paradigma precisam ser coerentes e são acionados como reação a algum tipo de gatilho, como por exemplo:

- Eventos gerados por software.
- Chamadas de funções.
- Mensagens, que são unidades de informação que o sistema envia ao usuário indicando sucesso, erro, falha, entre outros.

A figura 6 mostra um diagrama explicando o funcionamento de programção reativa.

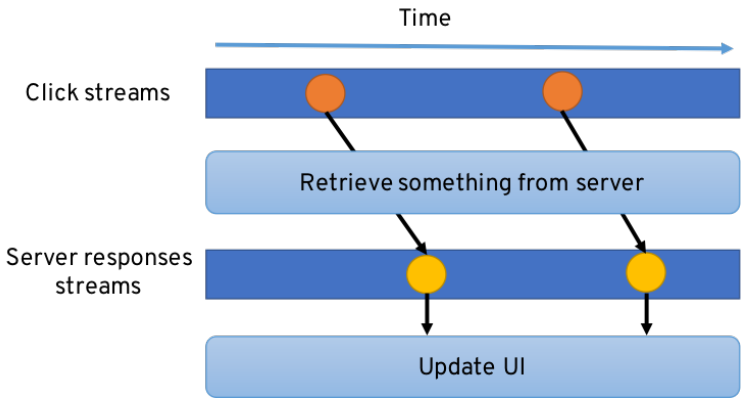

Figura 6: Diagrama indicando tratamentos de eventos e resposta do sistema de maneira reativa.

Dito isso, o funcionamento da programação reativa é fundamentada por inteiro em *streams* sequencialmente ordenadas por tempo relacionadas a mensagens de eventos. Uma *stream* típica inicia com um *observer*, que consiste em um segmento de código que observa algumas condições relacionadas a aplicação como no caso dos eventos, e os transmite para os tratadores que serão responsáveis por processar esses eventos e gerar respostas.

2.10 FLUTTER

Flutter é um conjunto de ferramentas designado para permitir o desenvolvimento de aplicativos de alta performance em diferentes plataformas, de modo que seja possível compartilhar a maior quantidade de código possível. Seu desenvolvimento é feito em camadas através de um conjunto de bibliotecas independentes, de modo que cada camada *framework* seja designada para ser opcional e substitutível (FLUTTER, 2020).

Através da figura 7 é possível ter um maior entendimento das camadas existentes. A camada de menor nível, ou seja, o *embedder*, é responsável por integrar o Flutter em outras aplicações como um módulo. Mais adiante, a *engine* suporta de maneira performática todo o necessário para as aplicações Flutter através de implementações em baixo nível para os gráficos, *layouts*, *inputs* e *outputs* de rede e ferramentas de compilação. Por fim, a Flutter *framework*, parte em que os usuários interagem mais tipicamente, provê uma *framework* reativa escrita em *Dart*, e contém uma grande quantidade de bibliotecas, plataformas e *layouts* para o desenvolvimento *mobile*.

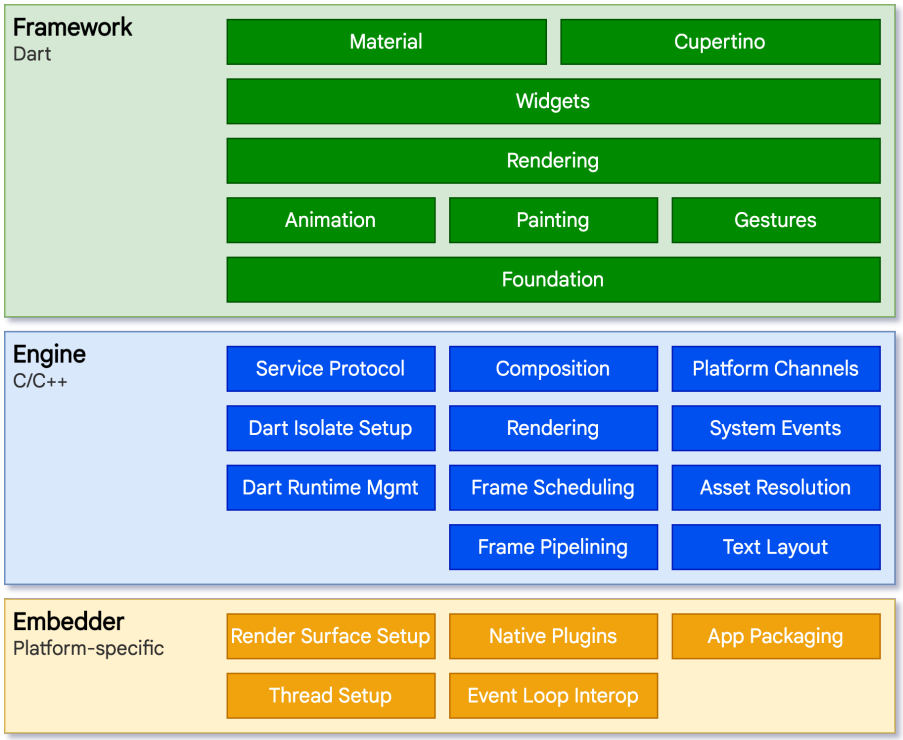

Figura 7: Diagrama indicando tratamentos de eventos e resposta do sistema de maneira reativa.

Desse modo, Flutter se tornou a opção escolhida para o desenvolvimento do aplicativo, tanto pela performance, quanto pela possibilidade de implementar um aplicativo com conceitos de programação reativa, de modo que o usuário tivesse maior responsividade ao utilizá-lo.

3 METODOLOGIA

3.1 VISÃO GERAL

Inicialmente, para o início da implementação do projeto, foram definidos os requisitos funcionais e não funcionais do projeto. Para isso, os requisitos foram divididos em requisitos do sistema microcontrolado e requisitos do aplicativo, que estão listados abaixo:

Requisitos Funcionais do Sistema Microcontrolado:

- **RF01:** o usuário deve ser capaz de saber se a bateria do sistema de detecção e comunicação está acabando;
- **RF02:** o usuário deve ser capaz de trocar a bateria que alimenta o sistema de detecção e comunicação;
- **RF03:** o usuário deve ser capaz de ligar e desligar o sistema;
- **RF04:** o sistema deve permitir modificar a cor dos LEDs para uma melhor detecção na imagem;
  - **RF04.1:** o usuário deve ser capaz de selecionar manualmente a cor dos LEDs;
  - **RF04.2:** o sistema deve conseguir identificar as melhores cores de acordo com o ambiente;
- **RF05:** o sistema deve tocar os sons dos tambores e pratos configurados conforme o movimento das baquetas;
- **RF06:** o sistema deve tocar os sons do bumbo conforme o pressionamento do pedal;
- **RF07:** o sistema deve ser capaz de tocar múltiplos sons simultaneamente;

Requisitos não Funcionais do Sistema Microcontrolado:

- **RN01:** os módulos das baquetas devem possuir um LED;

  - **RF01.1:** a câmera deverá captar a luz dos LEDs para permitir detectar a posição atual da baqueta;
  - **RF01.2:** os LEDs serão RGB, para permitir alterar sua coloração;

- **RN02:** os módulos das baquetas devem ter acelerômetros para medição de velocidade de suas batidas;
- **RN03:** a força no pressionamento dos pedais será medido por transdutores de força piezoeletricos;
- **RN04:** o sistema de detecção e comunicação será controlado por uma ESP32;
  - **RN04.1:** o cálculo do movimento das baquetas utilizando os acelerômetros será realizado por uma ESP32;
  - **RN04.2:** a alimentação desse sistema será feita através de uma bateria convencional de 9V;
- **RN05:** o cálculo da força de pressionamento no pedal será realizado pela Raspberry Pi;
- **RN06:** os microcontroladores ESP32 serão programados em C/C++;
- **RN07:** os movimentos das baquetas serão recebidos pelo Raspberry Pi por meio de sinais enviados pela ESP32;
- **RN08:** o microcontrolador Raspberry Pi será programado utilizando *Python*;
- **RN09:** os microcontroladores deverão se comunicar via Wi-Fi através de uma WLAN;
- **RN10:** devem haver suportes vestíveis para abrigar os fios e o módulo da ESP32 de forma a não atrapalhar o usuário no uso do equipamento.
- **RN11:** a amplitude do som dos tambores e pratos deve variar conforme a aceleração da baqueta;
- **RN12:** a amplitude do som do bumbo deve variar conforme a força do pressionamento dos pedais;

Requisitos Funcionais do Aplicativo:

- **RF08:** o usuário poderá selecionar um modelo de bateria para utilizar;
- **RF09:** o usuário poderá realizar a configuração de um modelo de bateria;

  - **RF09.1:** o usuário poderá selecionar cada peça da bateria, dentre as opções disponíveis;
  - **RF09.2:** o usuário poderá colocar as peças da bateria na posição desejada;

- **RF10:** o usuário poderá visualizar a configuração atual;
- **RF11:** o usuário poderá calibrar a bateria, quando desejado, através do aplicativo.

Requisitos não Funcionais do Aplicativo:

- **RN13:** O modelo de bateria configurado terá seu número de componentes limitado.
  - **RN13.1:** O valor limite será definido com base na capacidade de detecção do sistema implementado.
- **RN14:** o aplicativo será implementado utilizando o *framework Flutter*;
- **RN15:** o aplicativo deverá enviar a configuração da bateria para o Raspberry Pi;
- **RN16:** o aplicativo deverá requisitar a calibração ao usuário caso a configuração tenha sido alterada.

Além dos requisitos, foi realizado um diagrama em blocos mostrando o funcionamento geral do projeto, indicando as ações de cada componente e a comunicação entre componentes. A figura 8 mostra esse diagrama.

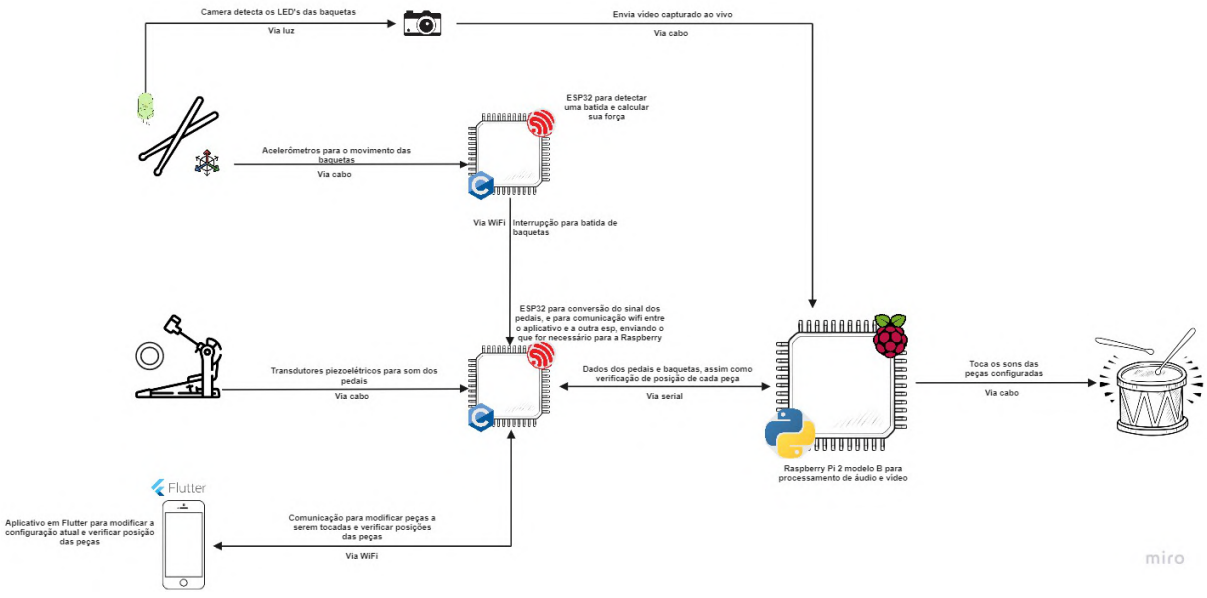

Figura 8: Diagram em blocos do projeto.

3.2 PROJETO MECÂNICO

A parte mecânica do projeto pode ser dividida em duas: as baquetas e os pedais. Para as baquetas, foi planejado utilizar baquetas comuns, e acoplar placas na ponta de cada baqueta. Seriam utilizados cabos *flat*, e seria passado algum tipo de fita em cima dos cabos para que eles não atrapalhassem o usuário.

Para os pedais, seriam utilizadas duas placas de madeira, conectadas por uma dobradiça, e uma mola entre elas para limitar o movimento e para que tenha uma necessidade de força para seu pressionamento. A imagem 9 mostra um diagrama da montagem dos pedais.

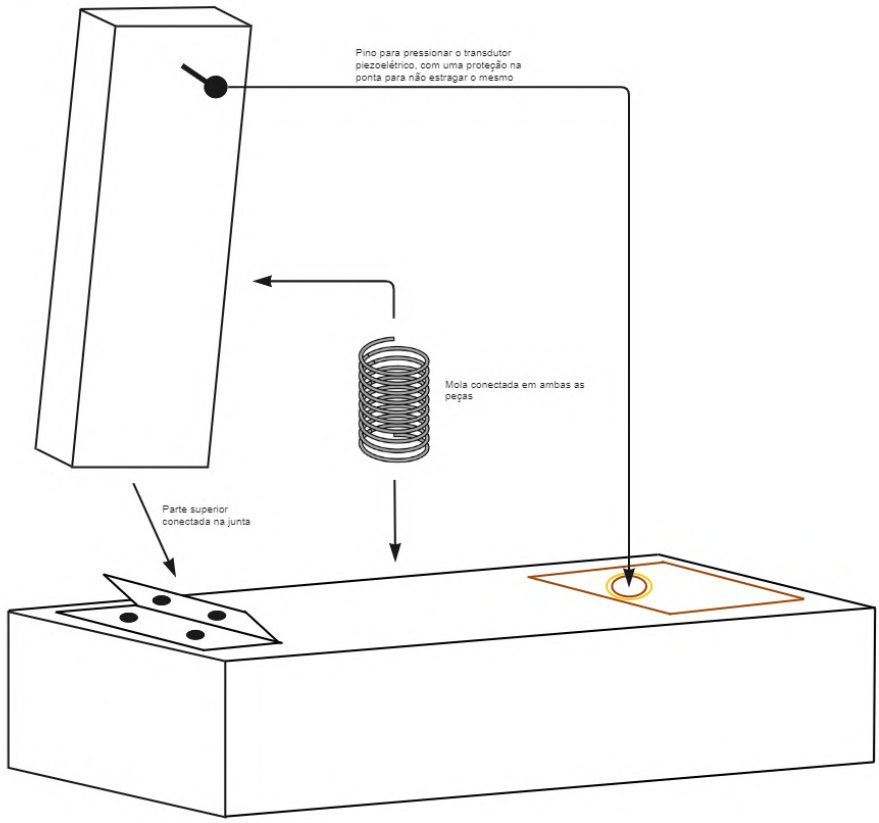

Figura 9: Diagrama indicando a montagem dos pedais.

3.3 PROJETO DE HARDWARE

Ao todo, o *hardware* dos AR Drums engloba 5 diferentes placas, relativas aos pedais, baquetas e ao circuito da ESP32. Todos os esquemáticos, PCBs e se encontram abaixo.

A Figura 10 diz respeito a placa de alimentação do sistema das baquetas. Nela, se encontra a segunda das ESP32, que alimenta os acelerômetros através de uma saída de 3.3V, além de ser a responsável por controlar os LEDs RGB das baquetas e comunicar com os

acelerômetros através de um canal I²C. A alimentação dessa placa é feita com uma bateria de 9V, que passa por um regulador de tensão para reduzir a tensão para 5V, a tensão nominal de alimentação da ESP32. Por fim, a placa possui ainda um *switch* para permitir ativar/desativar o sistema. A PCB projetada para essa placa se encontra na Figura 11.

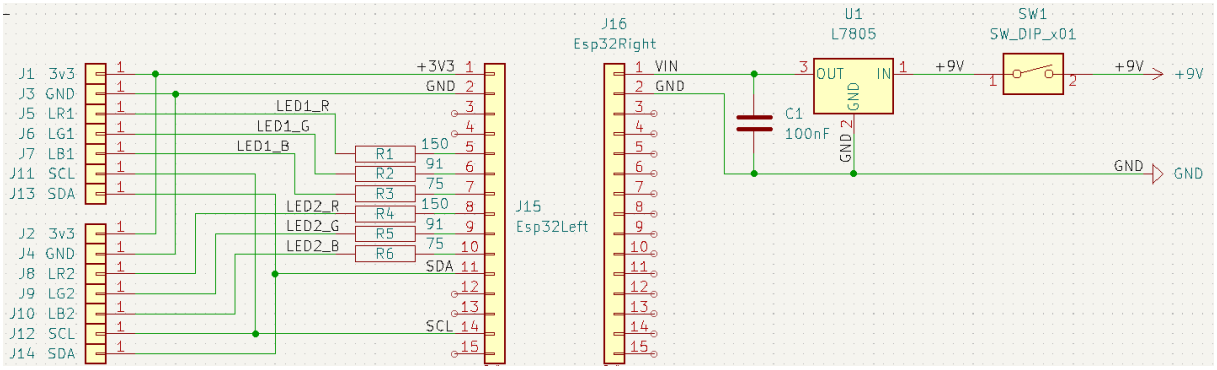

Figura 10: Esquemático da placa de alimentação e da primera ESP32 projetada.

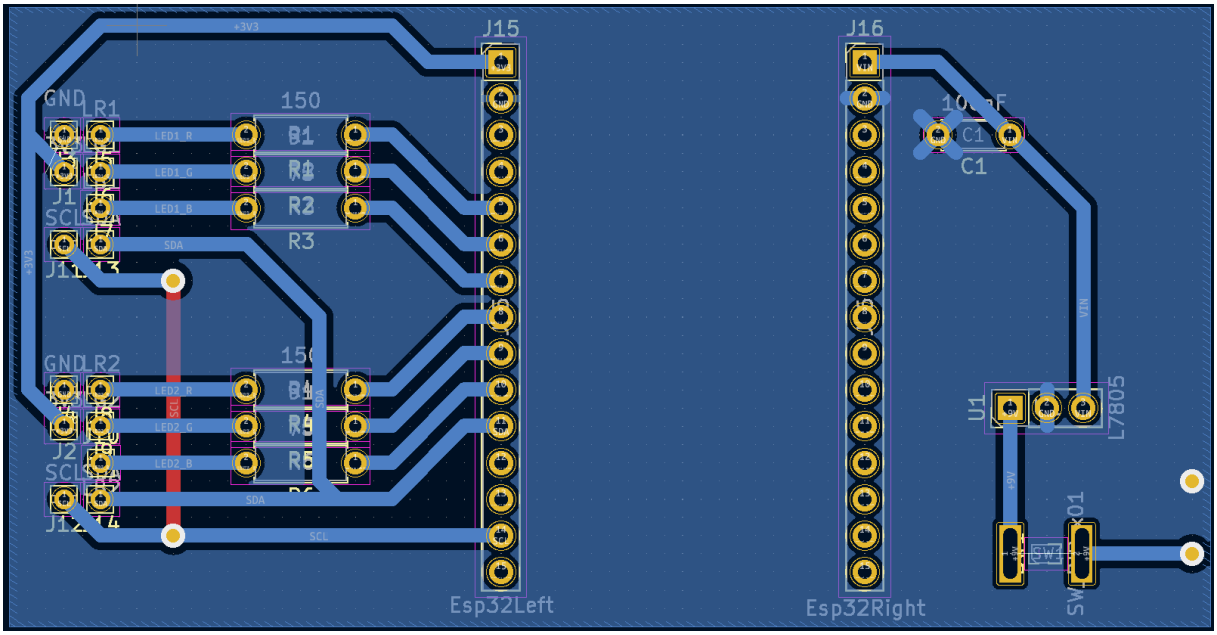

Figura 11: PCB da placa de alimentação e da primera ESP32 projetada.

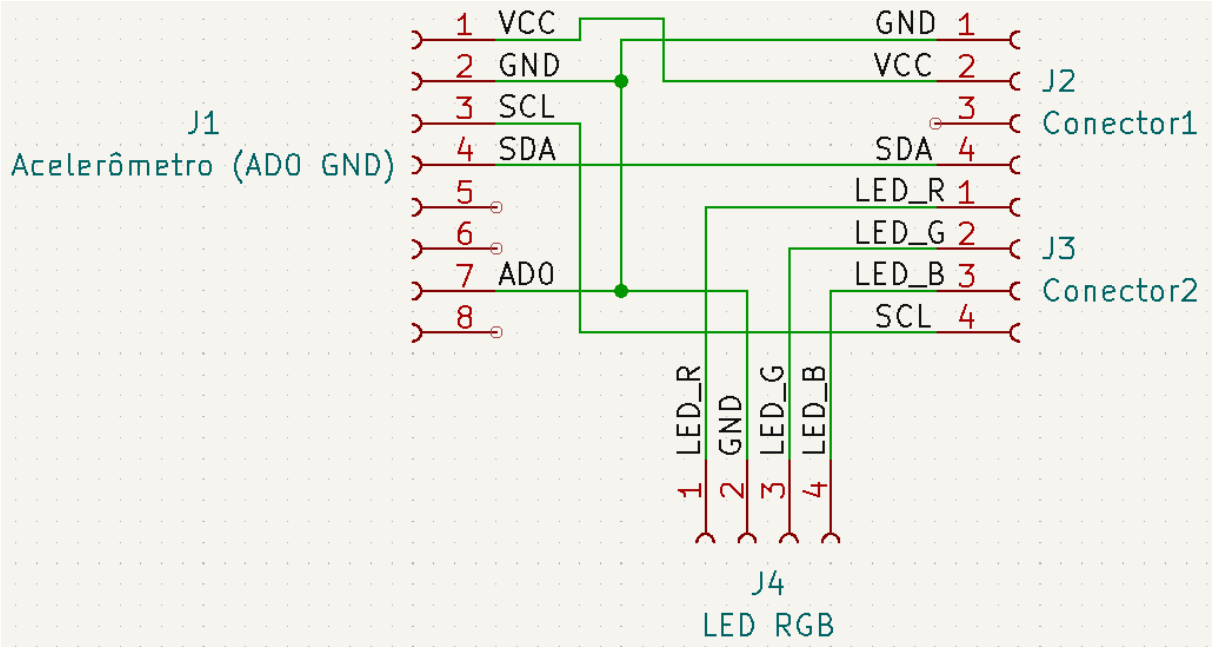

Figura 12: Esquemático da placa da primeira baqueta projetada.

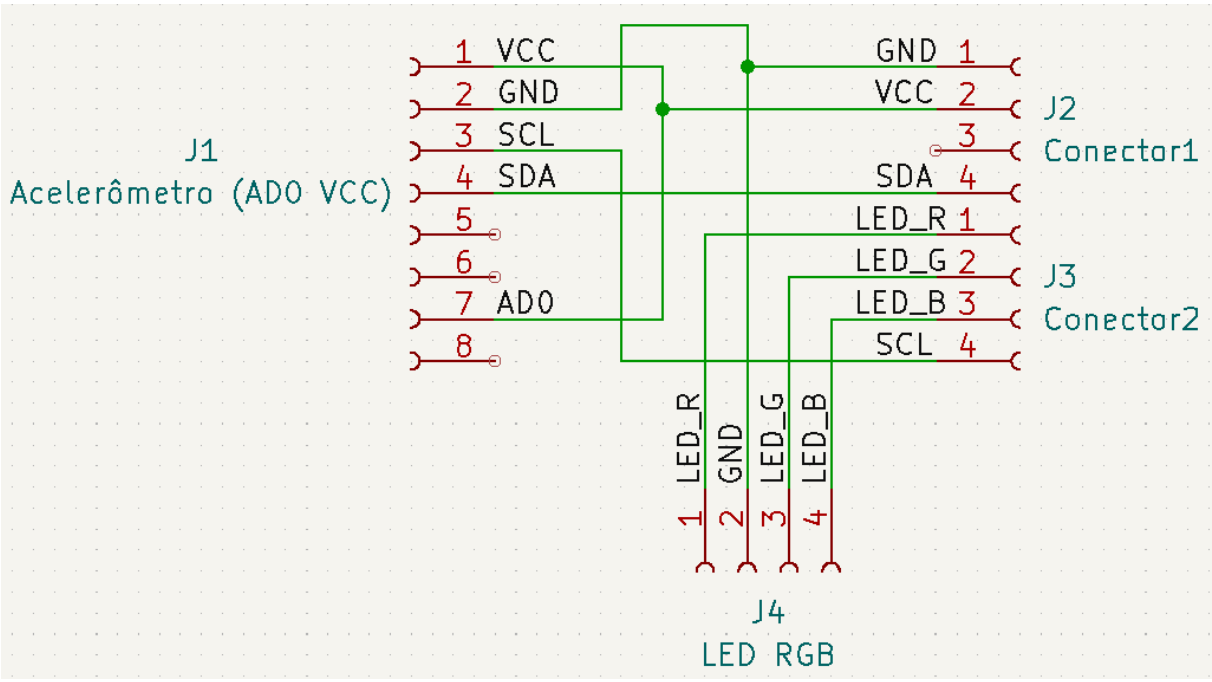

Figura 13: Esquemático da placa da segunda baqueta projetada.

As Figuras 12 e 13 apresentam os esquemáticos das placas das baquetas. Ambos seguem a mesma estrutura, com a única diferença sendo o pino 7 do conector J1 (Acelerômetro), que é referente ao endereço do componente para comunicação I²C. Na primeira baqueta, esse pino está conectado ao GND, correspondente ao endereço 0x68, e na segunda baqueta esse pino está conectado ao VCC, correspondendo ao endereço 0x69. As PCBs de ambas as placas

se encontram, respectivamente, nas Figuras 14 e 15.

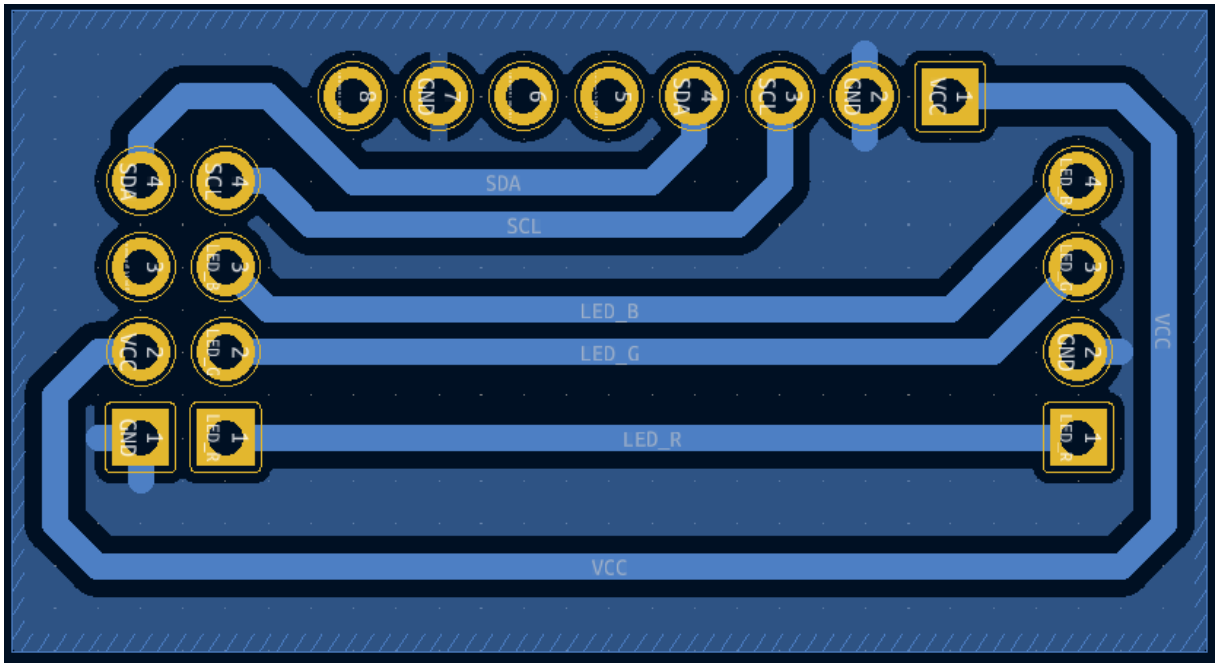

Figura 14: PCB da placa da primera baqueta projetada.

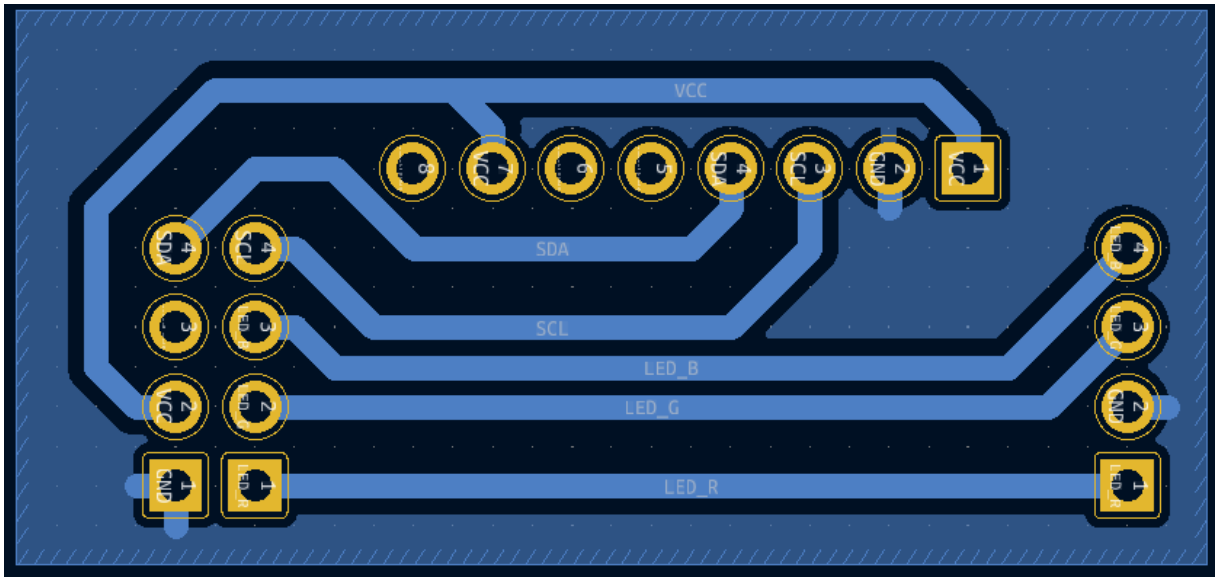

Figura 15: PCB da placa da segunda baqueta projetada.

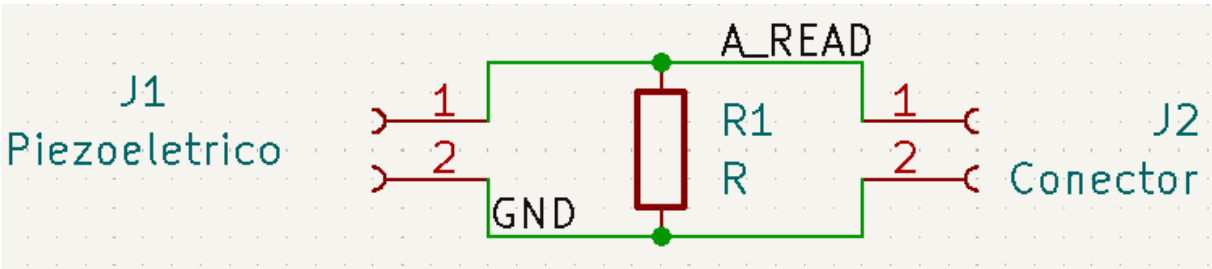

Figura 16: Esquemático da placa dos pedais projetada.

Por fim, a Figura 16 diz respeito à placa dos pedais. Ao todo foram confeccionadas duas dessas placas, idênticas para cada um dos pedais, cujo PCB se encontra na figura 17.

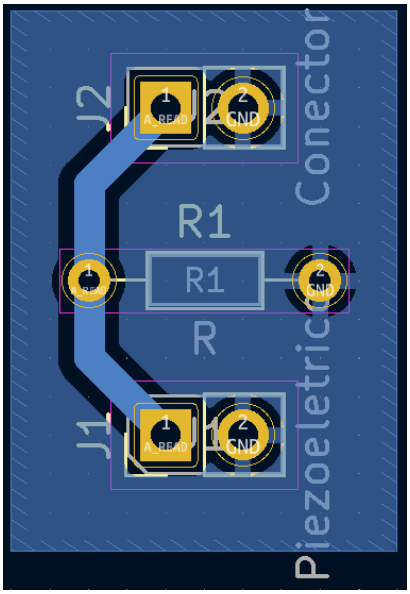

Figura 17: PCB da placa dos pedais projetada.

3.4 PROJETO DE SOFTWARE

O projeto de software é o mais extenso do projeto, sendo separado em múltiplas partes, representadas nas subseções abaixo.

3.4.1 VISÃO GERAL

Antes do projeto específico para cada parte de software do projeto, foi realizada uma visão geral de como o sistema deve se comportar, sendo controlado em essência por mensagens sendo trocadas entre todos os componentes. A figura 18 mostra o diagrama de casos de uso do sistema.

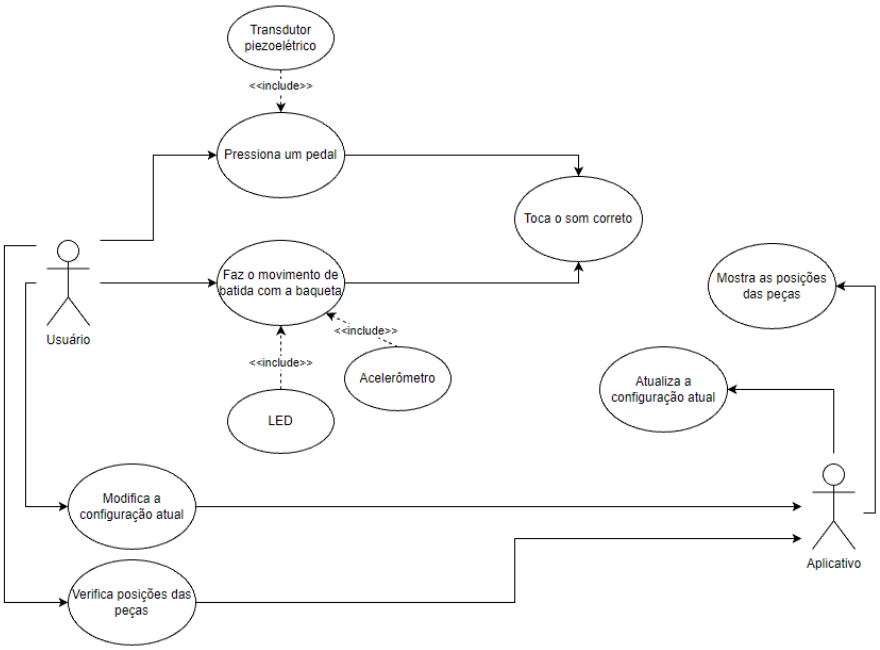

Figura 18: Diagrama de casos de uso do sistema.

Vale ressaltar que nesse diagrama boa parte do projeto foi abstraída para melhor entendimento, as próximas subseções irão mostrar o planejamento detalhado de cada componente de software do projeto.

3.4.2 RASPBERRY

A Raspberry possui três modos de operação:

- Tocando: modo de operação padrão, toca os sons a partir do pressionamento dos pedais, e dos movimentos das baquetas junto com o processamento de imagem;
- Configurando: modo de operação para alterar a configuração atual da bateria recebendo comandos do aplicativo;
- Calibração: modo de operação para enviar a posição atual de uma baqueta detectada pela imagem para visualização no aplicativo.

A alteração dos modos de operação é realizada via aplicativo. O diagrama abaixo indica os estados e transições da Raspberry:

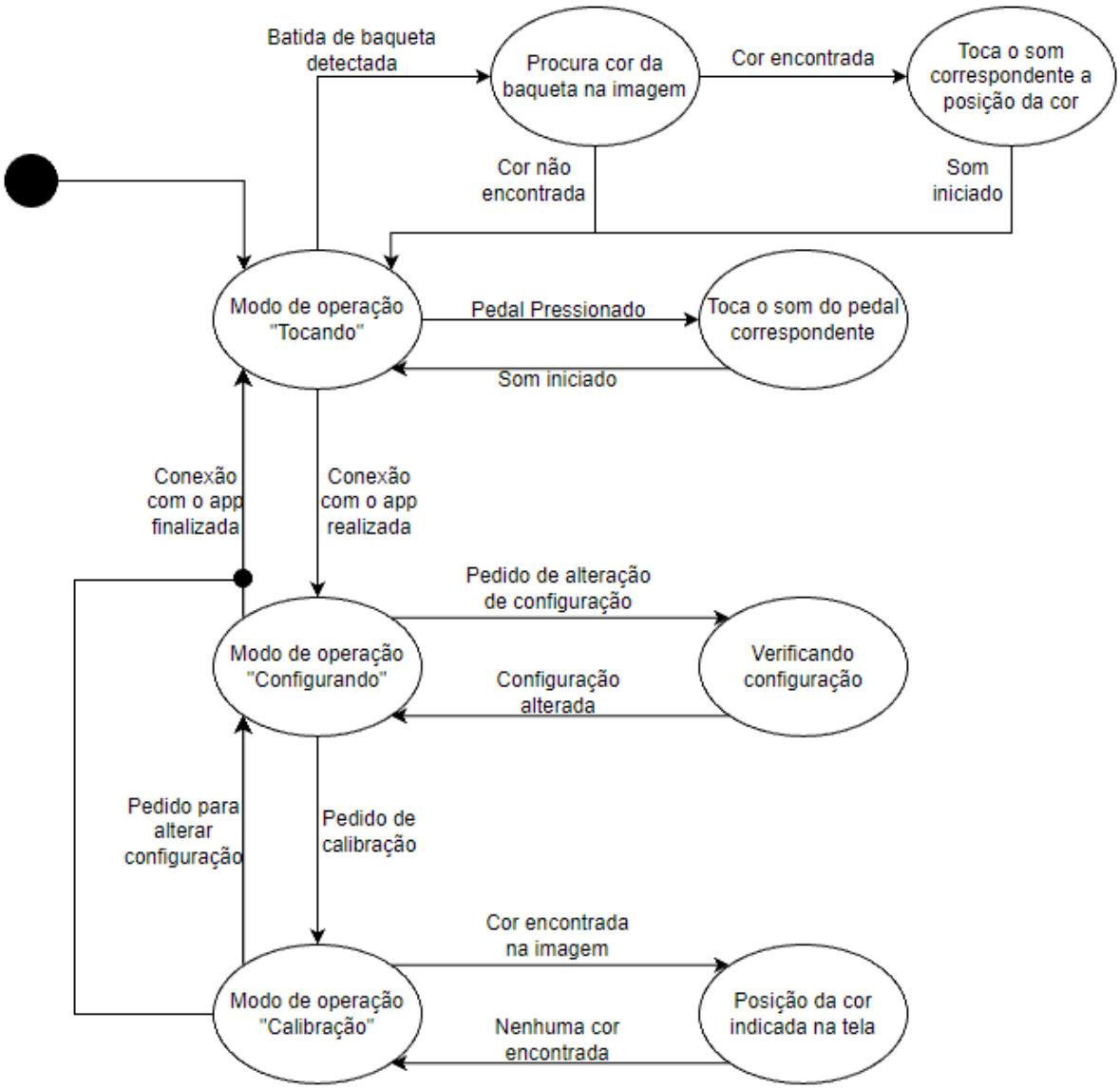

Figura 19: Diagrama de estados e transições do software da Raspberry.

No modo de operação ”Tocando”, a Raspberry está em seu papel principal do projeto, que é juntar todas as informações recebidas via serial pela ESP32 para fazer a simulação da bateria. Neste modo, serão precisas várias *threads*, pois cada peça da bateria roda em uma *thread* separada tocando o seu som conforme o pedal tocado, ou sua posição na imagem após a batida com uma baqueta. Cada *thread* sabe o que ela deve verificar para tocar o seu som, utilizando a biblioteca PyAudio para isso. A *thread* principal verifica se alguma mensagem chegou via serial, modificando variáveis globais para que todas as *threads* de peças possam verificar se seu som deve ou não ser tocado. Recebendo a mensagem correta via serial, a *thread* principal altera o modo de operação para ”Configurando”.

No modo de operação ”Configurando”, existe apenas uma *thread*, que constantemente

irá verificar se alguma mensagem foi recebida via serial. Neste modo, existem apenas três mensagens que serão tratadas: para voltar ao modo ”Tocando”, para ir ao modo ”Calibração”, ou para alterar o som de alguma posição da bateria. A configuração da bateria utiliza um arquivo *json* como base, e este arquivo é modificado caso uma mensagem de alteração da configuração seja recebida.

No modo de operação ”Calibração”, a Raspberry irá constantemente enviar a posição que ela está captando pelo vídeo, para que o aplicativo indique a posição atual da baqueta e, dessa maneira, o usuário pode ter certeza de onde que cada peça está posicionada. Além disso, este modo trata duas possíveis mensagens: para voltar ao modo ”Tocando”e para voltar ao modo ”Configuração”.

Para ilustrar o funcionamento do modo ”Tocando”, foram elaborados dois diagramas de sequência, relacionados aos casos de uso ”Pressiona um pedal”(figura 20) e ”Faz o movimento de batida com a baqueta”(figura 21).

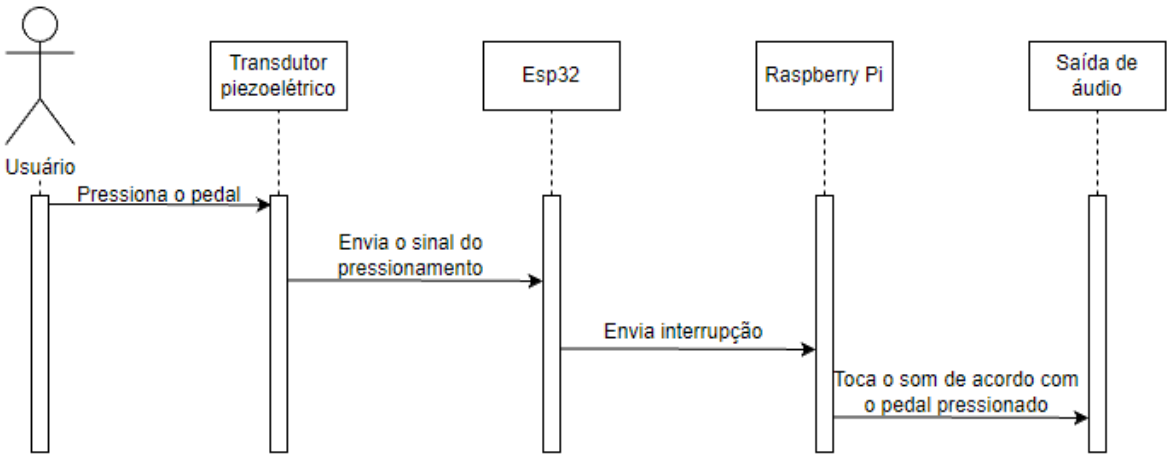

Figura 20: Diagrama de sequência relacionado ao pressionamento de um pedal.

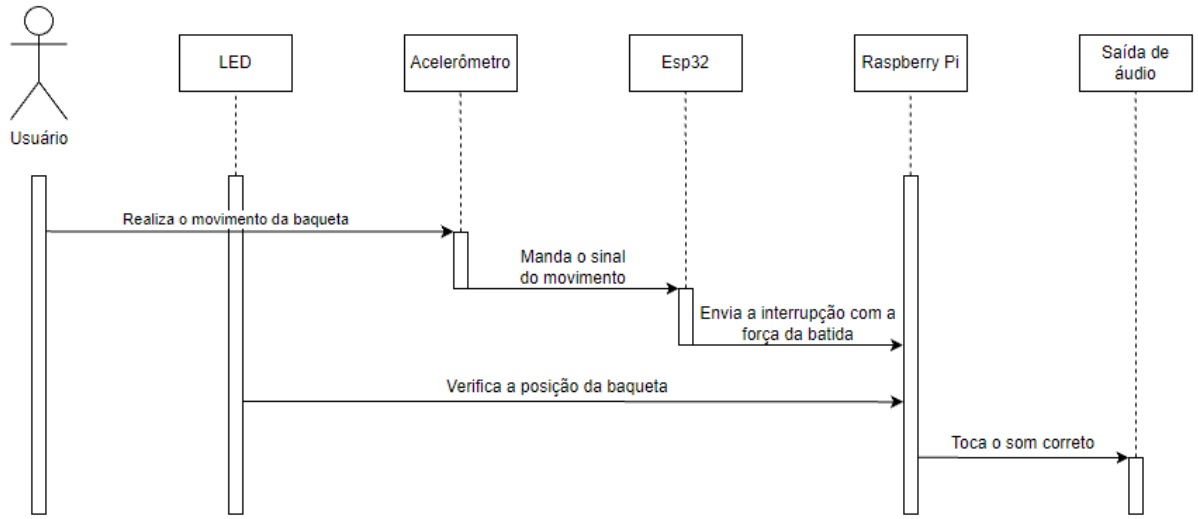

Figura 21: Diagrama de sequência relacionado ao movimento de uma baqueta.

3.4.3 ESP32

Como, para o projeto, serão utilizadas duas ESP32s, podemos dividir essa seção em duas, uma referente a cada uma das ESPs. De maneira geral, a primeira ESP (seção 3.4.3.1) é responsável por gerar o ponto de acesso e receber e enviar dados via Wi-Fi, e já a segunda ESP (seção 3.4.3.2) é responsável pelo processo de detecção de batidas e envio das detecções.

3.4.3.1 ESP32 - PONTO DE ACESSO WI-FI

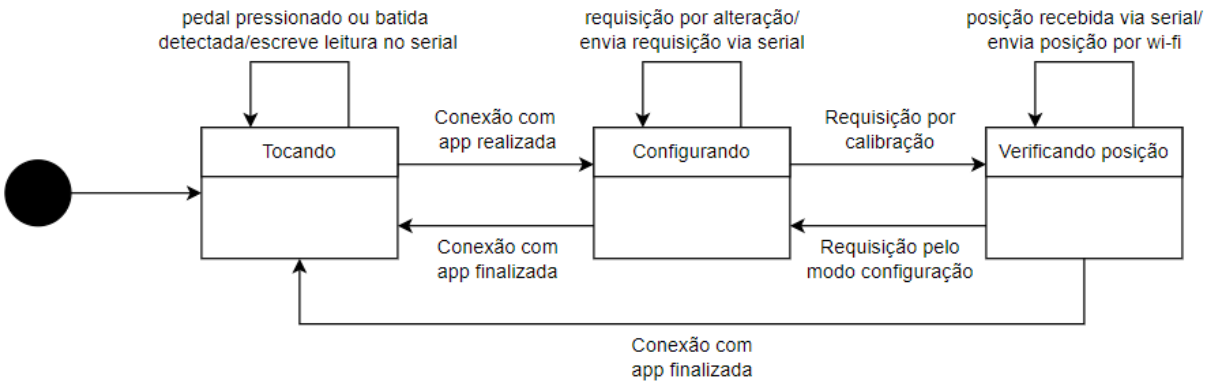

Figura 22: Diagrama de estados relacionado à primeira ESP32.

O funcionamento do *software* presente na ESP32 que gera o ponto de acesso Wi-Fi é apresentado no diagrama da Figura 22. Como é possível perceber, há varias semelhanças entre esse diagrama e aquele presente na Figura 19, visto que ambas a Raspberry Pi e a ESP32 operam em conjunto para o correto funcionamento do sistema. A diferença entre tais diagramas

se encontram nas transições que não alteram o estado atual, já que, no *software* da ESP32, é realizado o processo de comunicação entre a Raspberry, que envia/recebe dados via serial, e os demais equipamentos conectados ao ponto de acesso Wi-Fi.

3.4.3.2 ESP32 - CONTROLE E COMUNICAÇÃO DAS BAQUETAS

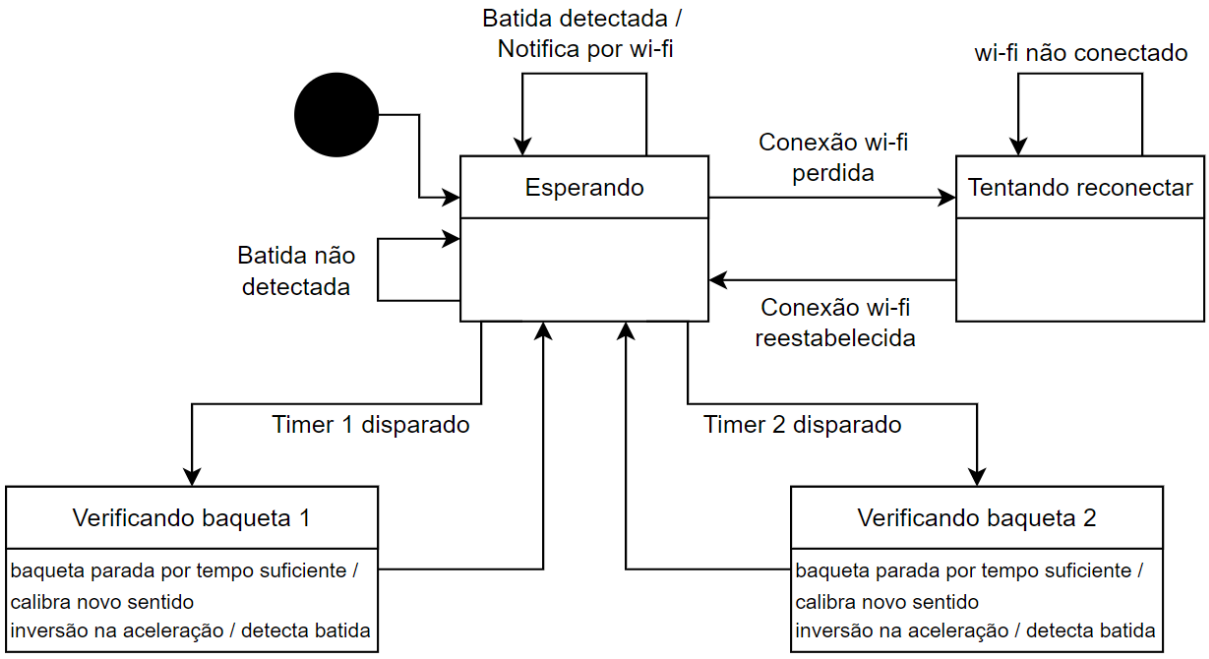

Figura 23: Diagrama de estados relacionado à segunda ESP32.

Já o *software* presente na segunda ESP32, responsável pelo controle das baquetas e detecção de batidas, segue uma estrutura diferente, apresentada na Figura 23. Esse sistema funciona à base de dois timers, que são responsáveis por transicionar o sistema para os estados de verificação das baquetas. Nesses, duas ações podem ocorrer, de acordo com a forma como as baquetas forem movimentadas: a) as baquetas ficam em repouso por tempo suficiente, fazendo com que o sistema possa se recalibrar de acordo com o atual sentido da força gravitacional sendo lida, ou; b) ocorre uma inversão no sentido da aceleração sendo lida, o que indica a ocorrência de uma batida. Além disso, o sistema presente na segunda ESP32 pode acabar adentrando um estado especial caso a conexão Wi-Fi seja perdida, permanecendo nele até que a conexão se reestabeleça. Por fim, o estado padrão do sistema é um estado de espera, que aguarda a detecção de uma batida para realizar a notificação via Wi-Fi.

3.4.4 APLICATIVO

Inicialmente, para o usuário ter acesso às funcionalidades do aplicativo, é necessário iniciar a conexão com os ARDrums, de modo que a comunicação entre ambos seja estabelecida. Assim que a conexão é estabelecida, o usuário pode escolher utilizar o aplicativo em dois modos de operação:

- Seleção de som: Modo de operação em que o usuário pode selecionar o som que deseja tocar em uma posição do grid.
- Calibração: Modo de operação em que o aplicativo indica a posição no *grid* em que o usuário está posicionando as baquetas.

No modo de seleção de som, como pode ser visto na figura 24, o usuário poderá escolher uma das posições do grid e escolher o som que deseja que seja tocado naquela respectiva posição. Essa informação é enviada via *socket UDP* para a Raspberry, onde o tratamento é realizado e o novo som é configurado.

No modo de calibração, como pode ser visto na figura 25, o aplicativo também inicia uma conexão via *socket UDP* com a Raspberry para receber as posições do grid em que as baquetas estão posicionadas, de modo que o aplicativo é atualizado de maneira reativa para indicar ao usuário as posições em que se encontram as baquetas.

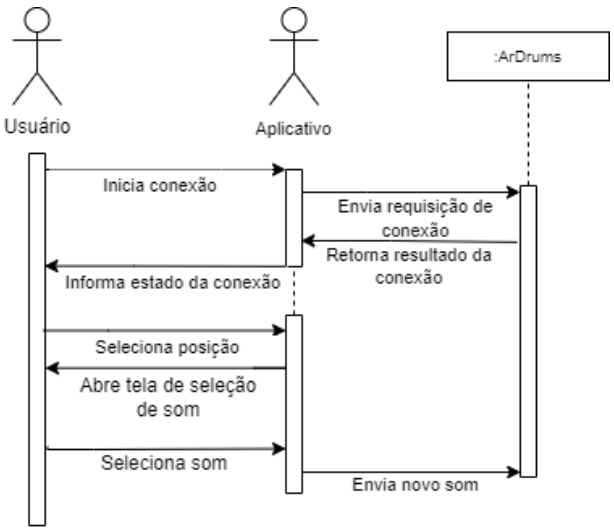

Figura 24: Diagrama de sequência relacionado a seleção de som do app.

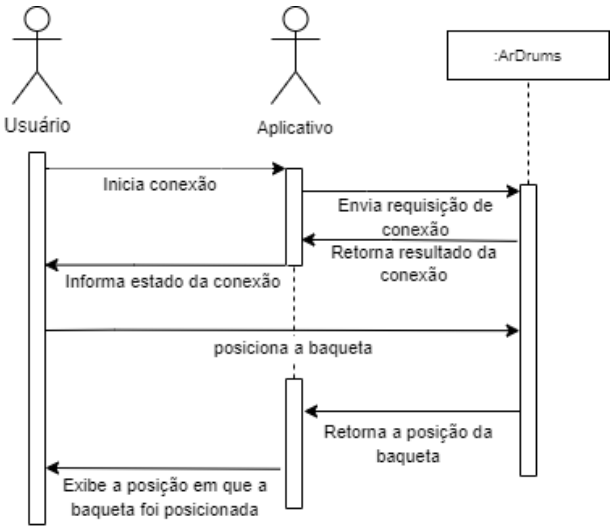

Figura 25: Diagrama de sequência relacionado a calibração da baqueta.

3.5 INTEGRAÇÃO

A integração do projeto teve que ser dividida em múltiplas etapas sequenciais, através das quais diferentes componentes e funcionalidades seriam adicionados ao conjunto, um após o outro, até que todo o projeto fosse devidamente integrado.

A primeira integração a ser realizada seria para que o pressionamento dos pedais fossem corretamente tradados pela sua ESP32, que por sua vez iria enviar a informação do pressionamento para a Raspberry. Eles devem ser conectados aos pinos corretos da esp, que vai tratar o valor recebido pelo pressionamento, indicando se ele deve ou não ser considerado um pressionamento do pedal, e esse valor deve ser enviado para a Raspberry via serial. Ao ver a mensagem do pressionamento chegando na Raspberry, essa parte estaria concluída.

A segunda integração é relacionada a toda a parte de recebimento de batidas e verificação de posição da baqueta que realizou tal movimento. Ao realizar o movimento de batida com uma baqueta, a ESP32 conectada a ela irá tratar os dados dos acelerômetros para verificar se o movimento deve ou não ser considerado uma batida. Caso ele seja, essa informação seria enviada via Wi-Fi para a outra ESP32, que simplesmente envia essa informação para a Raspberry. Com essa mensagem, a Raspberry irá verificar a imagem da câmera procurando pela cor da baqueta que realizou o movimento, tocando o som correto de acordo com a posição.

A terceira parte da integração, deve ser uma mudança de configuração utilizando o aplicativo. O aplicativo deve enviar uma mensagem para a ESP32 para realizar a conexão, que irá alterar o modo de operação da Raspberry. Após isso, uma configuração dever ser alterada

no aplicativo, alterando o arquivo de configuração da bateria na Raspberry, e o aplicativo deve finalizar a conexão, voltando para o modo de operação padrão da Raspberry.

A quarta parte da integração, deve ser uma verificação das posições das peças pelo aplicativo. O aplicativo deve enviar uma mensagem para a ESP32 para realizar a conexão, que irá alterar o modo de operação da Raspberry. Em seguida, o aplicativo deve mudar para o modo de calibração, fazendo o mesmo caminho, indo para a ESP32 e para a Rasp em seguida. Com isso, colocando uma baqueta em uma posição dentro do alcance da câmera, essa posição deve ser indicada no aplicativo.

A quinta e última parte de integração deve ser a integração final, onde todas as partes separadas devem funcionar na mesma execução.

4 EXPERIMENTOS E RESULTADOS

4.1 PARTE MECÂNICA

Para as baquetas, foram utilizados cabos de rede para a comunicação, que passam pelas baquetas protegidos por uma fita isolante, que também faz com que a baqueta possua um *grip* melhor, já que o usuário não terá contato direto com os cabos. Ambos os terminais dos cabos se utilizam de conectores micro-fit. A figura 26 mostra as baquetas finalizadas.

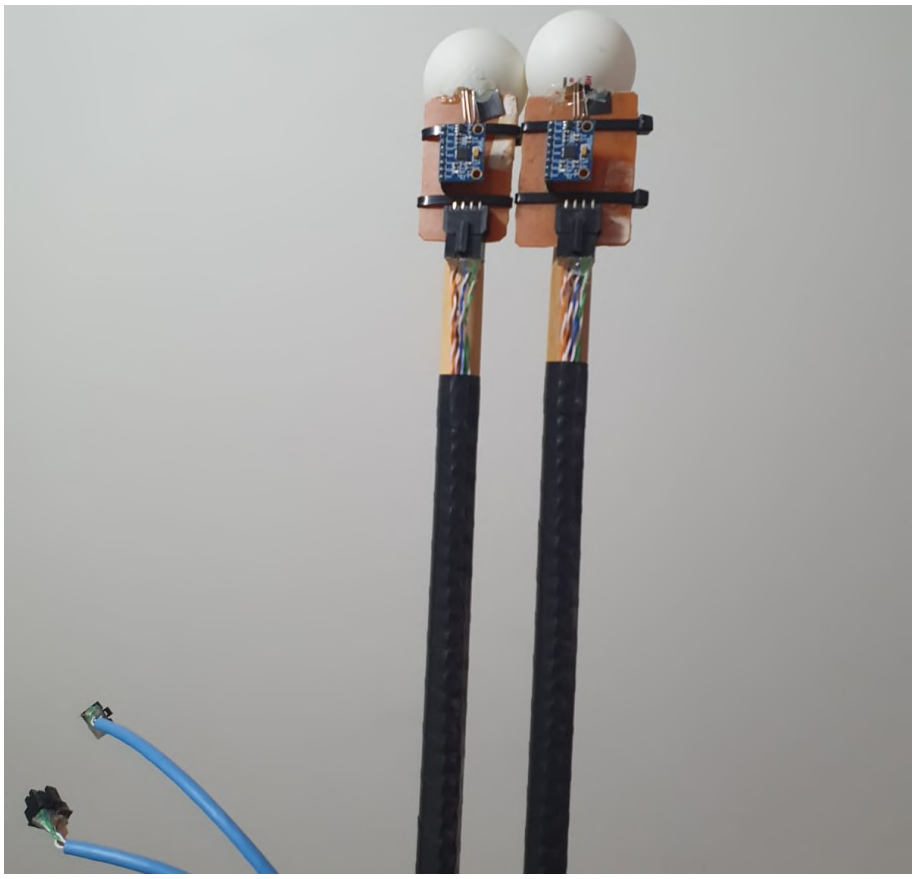

Figura 26: Baquetas finalizadas.

Para os pedais, foram utilizadas placas de madeira, sendo a base maior que a placa superior, que estará em contato com o pé do usuário. As duas placas são conectadas por uma

junta de metal, e uma mola é inserida entre essas partes para que exista o pressionamento. As figuras 27 e 28 mostram uma visão frontal e lateral, respectivamente, de um pedal confeccionado.

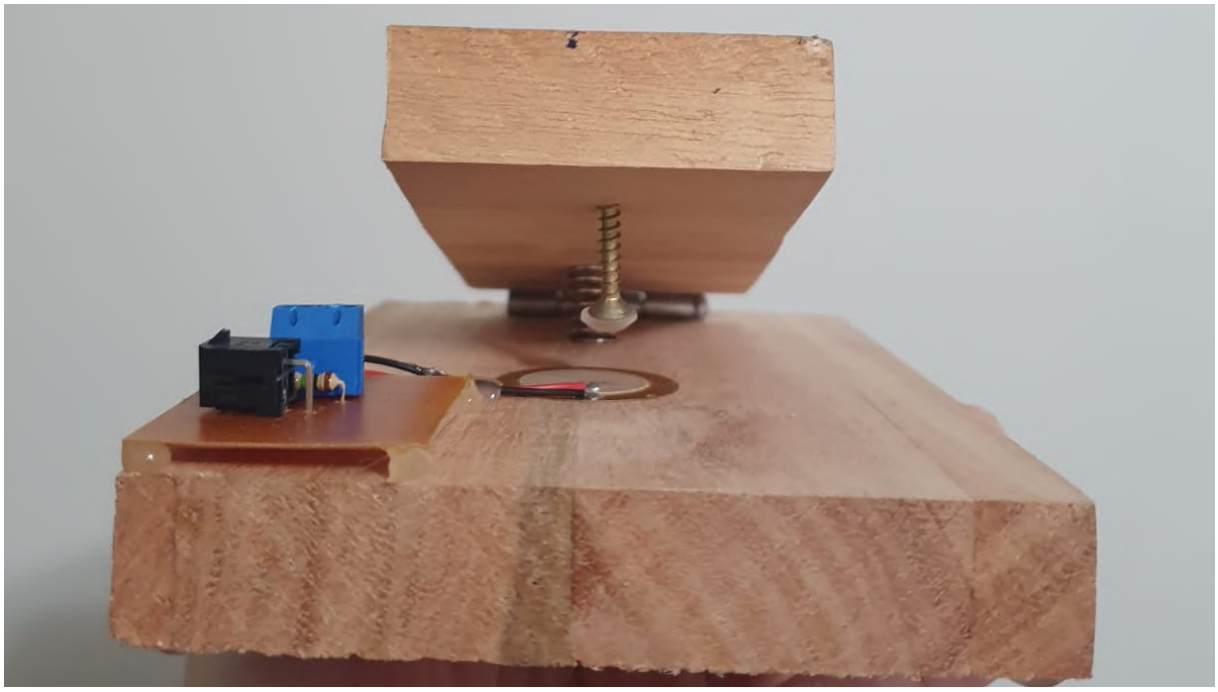

Figura 27: Visão frontal de um pedal.

Figura 28: Visão lateral de um pedal.

4.2 PARTE DE HARDWARE

O *hardware* elaborado pode ser visto nas Figuras 29 a 32. De maneira geral, as PCBs feitas seguem os moldes dos esquemáticos apresentados na seção 3.3 e funcionaram de acordo

com o planejado. Porém, as PCBs de fato feitas correspondem às primeiras versões elaboradas dos projetos esquemáticos e, por conta disso, apresentam falhas de projeto que foram corrigidas tanto nos esquemáticos quanto manualmente, nas próprias placas. Por fim, uma última placa foi feita, que não estava incluída no projeto original, para abrigar a ESP32 que gera o ponto de acesso Wi-Fi, como também os conectores borne para os cabos de comunicação com os pedais, conforme apresentado na Figura 32.

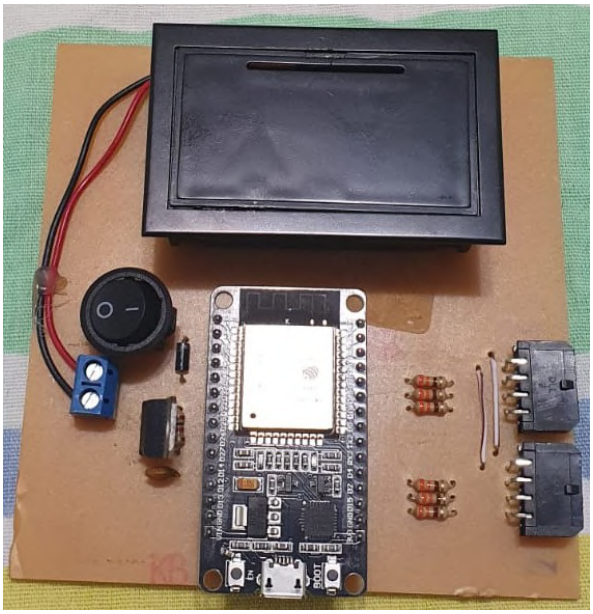

Figura 29: Imagem da PCB finalizada da placa de alimentação.

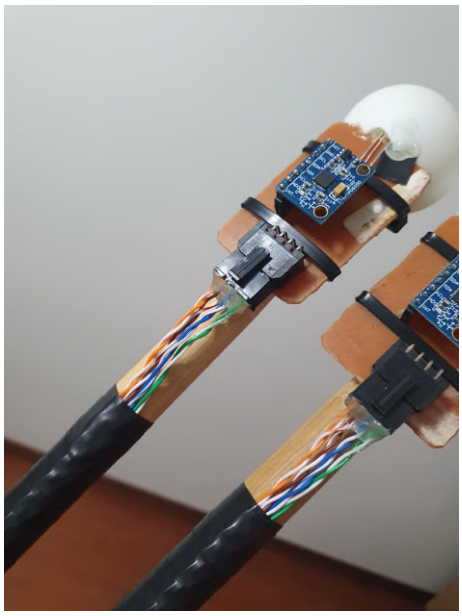

Figura 30: Imagem das PCBs finalizadas das baquetas.

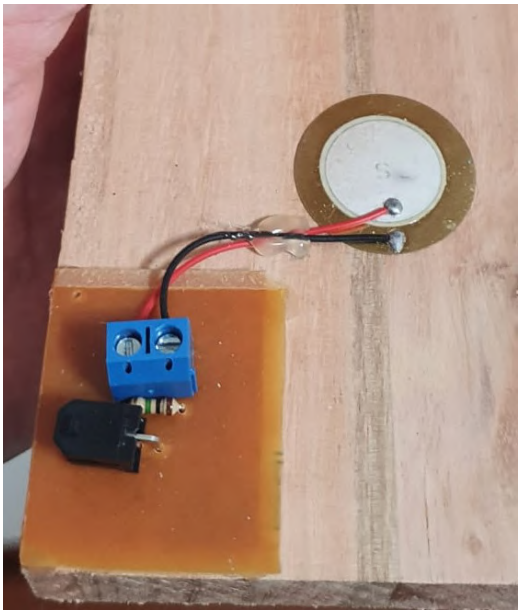

Figura 31: Imagem da PCB finalizada de um pedal.

Figura 32: Imagem da PCB finalizada da segunda ESP32, ligada aos cabos de comunicação com os pedais.

4.3 PARTE DE SOFTWARE

Os resultados de software são divididos em três partes, o software da Raspberry Pi, os softwares das Esp32, e o software dos aplicativos.

4.3.1 SOFTWARE DA RASPBERRY PI

O processamento de áudio realizado pela Raspberry Pi possui algumas limitações. A saída de áudio *jack* da placa não suporta mais que quatro arquivos sendo tocados simultaneamente. Isso traz problemas com pratos, que possuem arquivos de áudio longos, então caso vários pratos sejam tocados ao mesmo tempo, todos os instrumentos após o quarto prato não seriam tocados. Apesar disso, esse problema não chegou a ser tratado por conta de problema de desempenho da Raspberry Pi, limitando a quantidade de peças configuradas para que o sistema tenha um bom desempenho, o que será discutido em mais detalhes na seção 4.4. Apesar deste problema, os diferentes sons podem ser tocados simultaneamente, com um tempo de resposta variável, mas dentro do esperado.

Já para a parte do processamento de vídeo, a câmera utilizada foi configurada para ter uma resolução de 96x66, para reduzir o processamento necessário, já que o sistema é executado em tempo real. As máscaras para a detecção de cores funcionam muito bem, para uma iluminação correta. Caso exista alguma fonte de luz mais forte que os leds das baquetas no campo de visão da câmera, ela pode ter problemas para encontrar as cores. O ambiente ideal para que as cores sejam captadas corretamente é com uma fonte de luz atrás da câmera iluminando o ambiente, com as baquetas sendo as fontes de luz predominantes na frete da câmera.

A figura 33 mostra uma foto com uma cor ciano predominante. A figura 34 mostra uma máscara para encontrar a cor ciano sendo aplicada na imagem original. Nessas imagens é possível visualizar que mesmo tendo outros tons de azul na imagem além do celular (inclusive ciano, nos detalhes do moletom), apenas a luz do celular com a cor é captada na máscara, o que facilita no tratamento para encontrar a posição da cor. Vale ressaltar que a resolução da câmera foi aumentada nessas imagens, para fins de apresentação.

Figura 33: Imagem original.

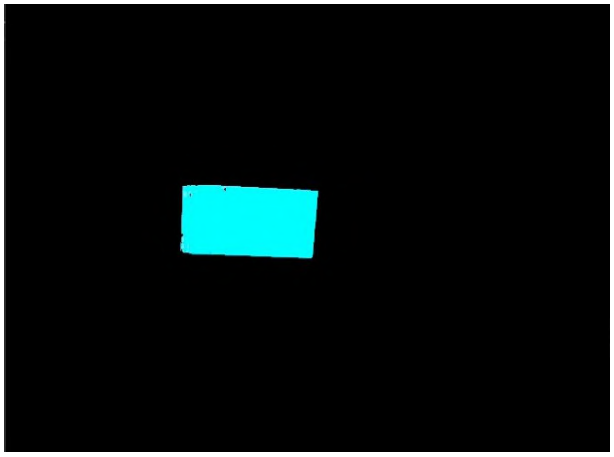

Figura 34: Imagem com uma máscara para a detecção da cor ciano.

Por fim, é importante deixar claro como que as posições das peças são mapeadas na imagem. A imagem é dividida em quatro quadrantes, de forma que cada um pode ter um instrumento. Para saber a posição exata da cor e identificar qual peça está sendo tocada, é utilizada a média das posições encontradas, evitando assim que dois instrumentos sejam tocados simultaneamente com uma única batida de uma baqueta.

4.3.2 SOFTWARE DAS ESP32

Ambos os softwares elaborados para as ESP32s apresentaram resultados satisfatórios. Com relação ao software da ESP32 que gera o ponto de acesso Wi-Fi, seu funcionamento se enquadrou totalmente dentro do esperado, apresentando tempos de resposta muito satisfatórios para o processo de recepção de dados via Wi-Fi e envio destes pelo canal serial, e para o processo inverso também.

Os resultados apresentados pelo software da ESP32 responsável pelo controle das baquetas se enquadraram, também, dentro do esperado, porém, é valido ressaltar alguns pontos.

Uma adição feita ao projeto inicial desse software, com a intenção de melhorar a interação com o usuário, foi a de adicionar um indicador visual para a conexão Wi-Fi. Isso é feito acendendo as baquetas na cor vermelha, caso a conexão não esteja estabelecida ou seja perdida, e nas cores padrão (ciano e magenta), caso a ESP32 esteja devidamente conectada à rede Wi-Fi, conforme apresentado na figura 35.

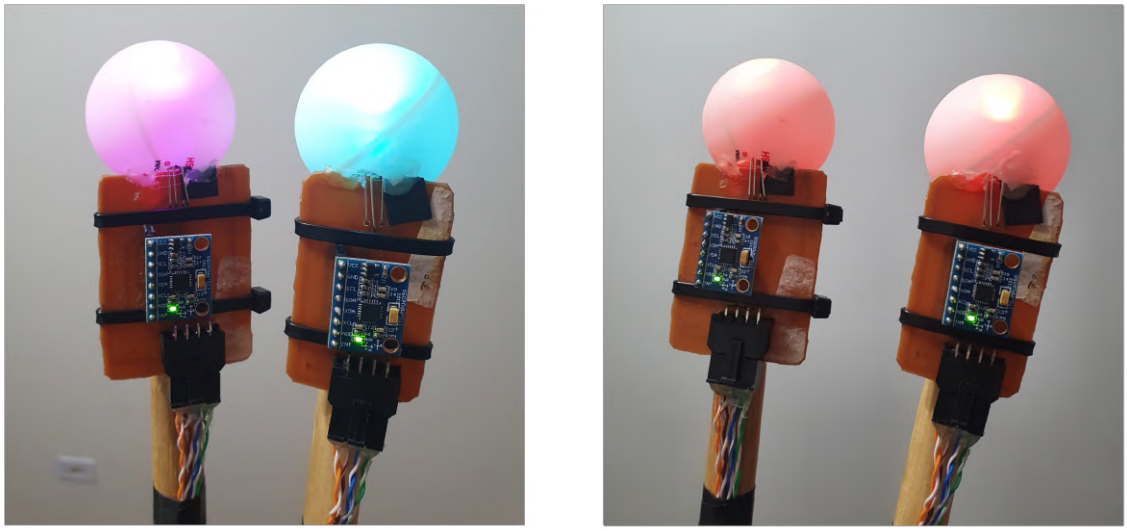

Figura 35: Imagens ilustrando as baquetas nas cores padrão, à esquerda, e em vermelho, quando há perda de conexão, à direita.

Outro ponto importante é com relação à calibração das baquetas. Conforme apresentado no diagrama de projeto do software (Figura 23), o sistema realiza uma verificação para identificar se uma baqueta ficou aproximadamente em repouso, o que é feito verificando-se que a aceleração total se encontra abaixo de um valor limite, próximo à aceleração da gravidade. Através disso, é possível identificar para qual direção aponta, em repouso, a aceleração gravitacional, e com isso definir para qual direção as baquetas devem ser aceleradas para que uma batida seja detectada. A importância desse tópico é para evitar batidas não detectadas, uma vez que, caso as baquetas sejam giradas nas mãos do usuário durante o uso, a direção de detecção pode acabar ficando errada, sendo necessário re-calibrar as baquetas.

Por fim, uma outra alteração realizada com relação ao projeto de software realizado diz respeito à inclusão de um *timeout* após a detecção de uma batida. A utilidade de tal recurso é evitar a detecção múltipla de uma batida e, com isso, evitando que o hardware responsável pelo processamento do vídeo e do audio seja sobrecarregado. Tal valor de *timeout* foi configurado para cerca de 100ms, valor suficientemente pequeno para que ainda seja possível realizar

diversas batidas sequencialmente.

4.3.3 SOFTWARE DO APLICATIVO

De modo geral, o aplicativo foi construído de maneira a cumprir com as espectativas. Em seu desenvolvimento foi possível realizar conexões *full-duplex* com a ESP de modo a aumentar a integração do projeto. Através dessa conexão foi possível criar a interatividade entre o usuário e o sistema, de modo que ele pudesse escolher os sons desejados no modo de seleção de som e se orientar quanto as posições do aplicativo no modo calibração. As figuras 36 a 42 exemplificam as funcionalidades obtidas.

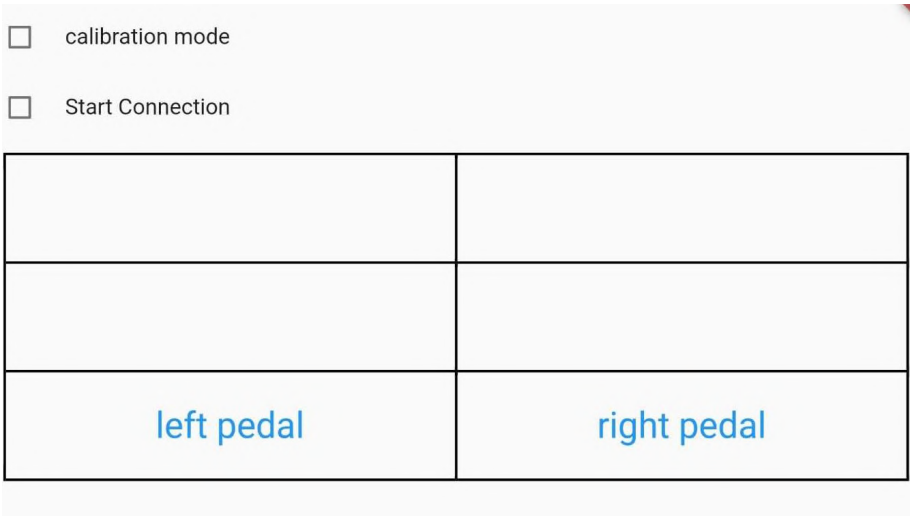

Figura 36: Tela inicial do aplicativo.

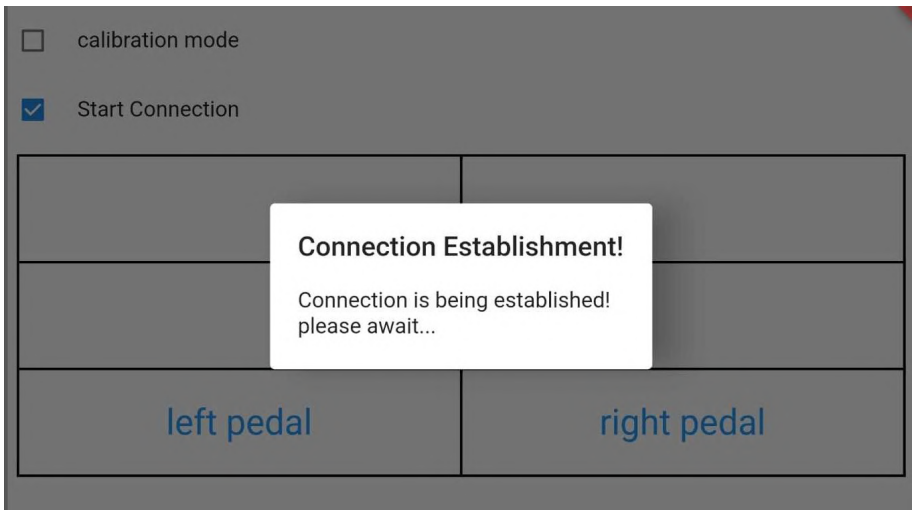

Figura 37: Tela de estabelecimento de conexão.

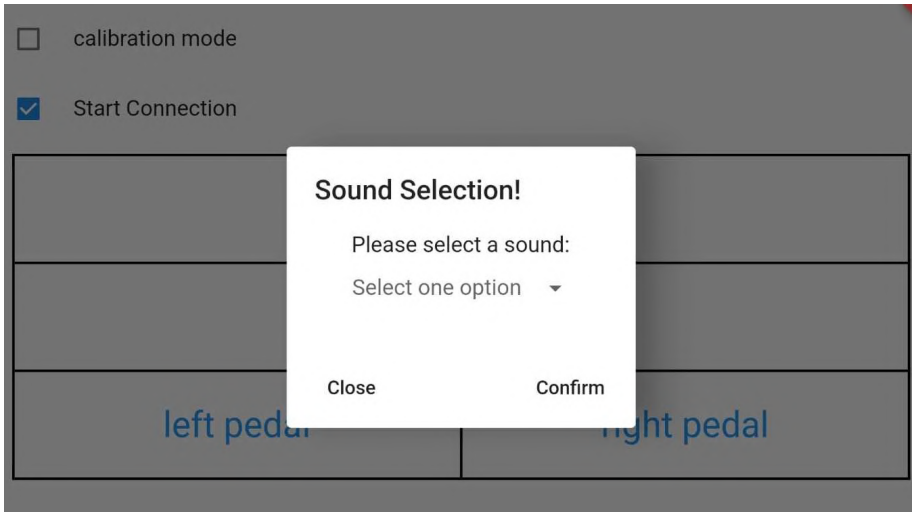

Figura 38: Tela para selecionar sons.

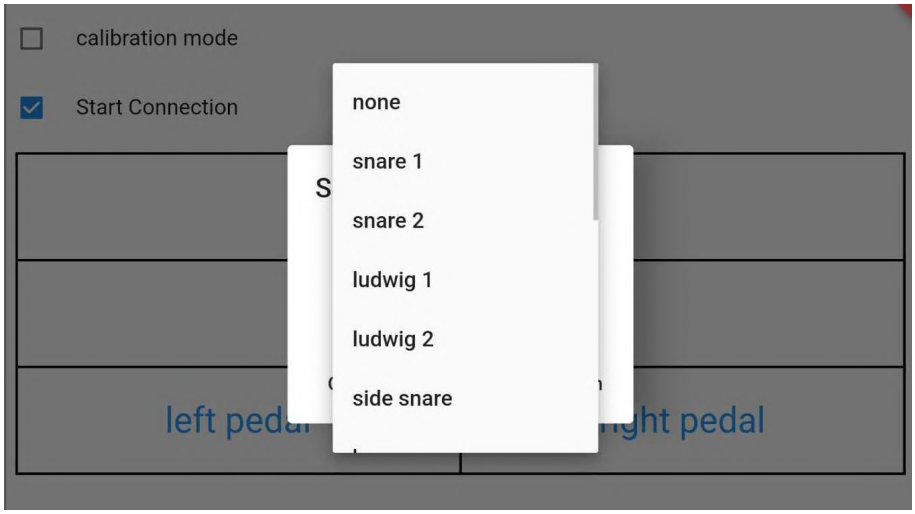

Figura 39: Tela com a lista dos sons selecionáveis.

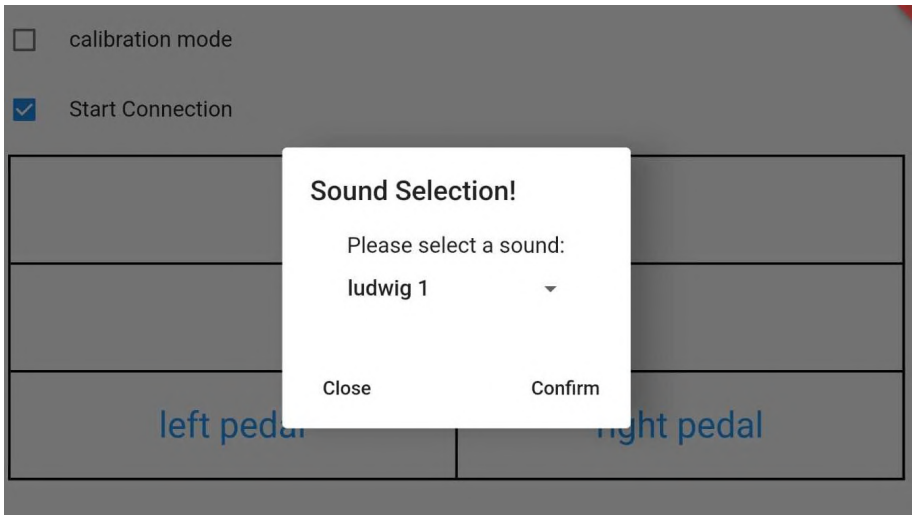

Figura 40: Tela de confirmação de seleção de som.

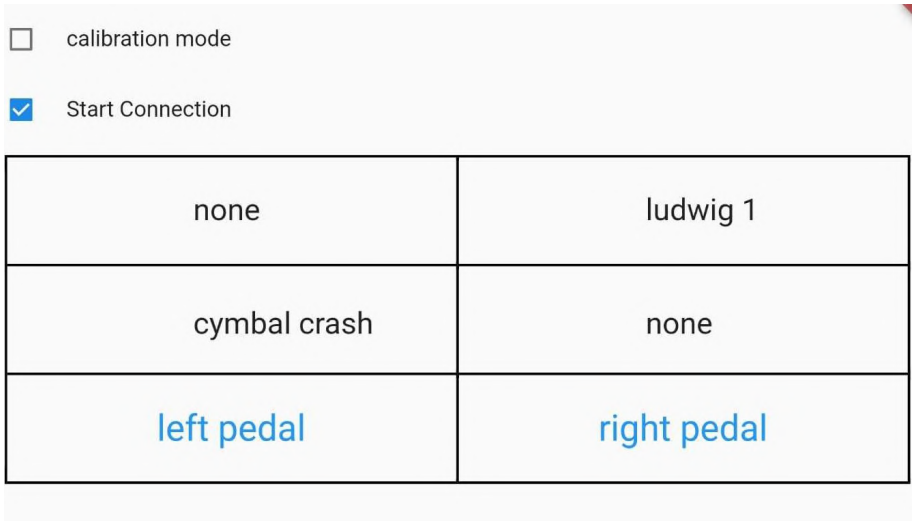

Figura 41: Tela de exibição dos sons escolhidos nas posições configuradas.

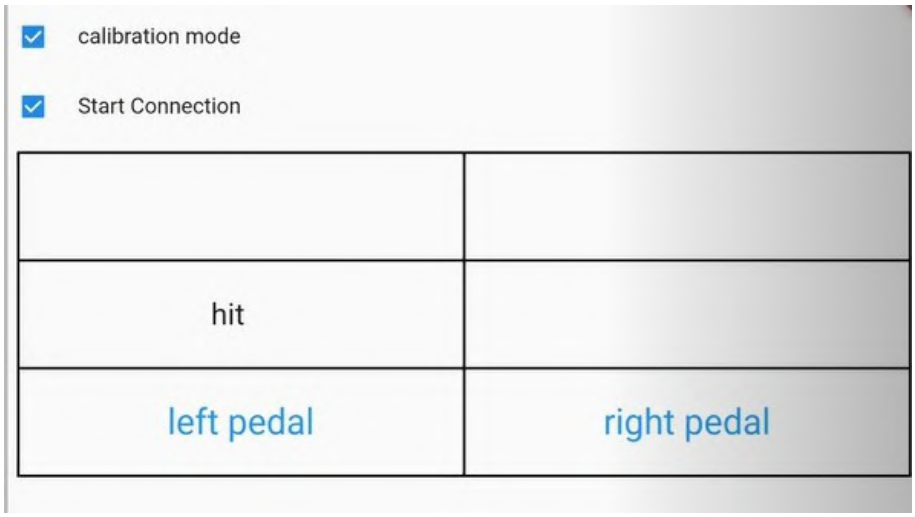

Figura 42: Tela de calibração indicando a posição das baquetas.

4.4 INTEGRAÇÃO

Ao integrar as diferentes partes do projeto, ele funcionou próximo ao esperado. A comunicação da Raspberry com o aplicativo, para indicar a posição atual da baqueta durante o modo de calibração, possui um atraso maior do que o esperado. Já as demais funções do aplicativo funcionaram dentro das expectativas.

O processo de detecção das batidas possui algumas inconsistências, fazendo com que, certas vezes, uma batida realizada pelo usuário não seja captada, para na próxima batida duas serem registradas, fazendo um som duplicado. Os pedais tem um delay relativamente alto em seu pressionamento, por conta do processamento do resto do sistema, já que é tudo feito

simultaneamente.

Além disso, pelo fato de a Raspberry possuir apenas quatro *Threads* físicas, a execução para mais que 3 instrumentos tem altos atrasos. Essa situação não ocorreu para *multithreading* em partes separadas da integração, apenas na versão final. A causa identificada para isso é uma sobrecarga de processamento na *thread* principal, responsável por tratar todos os dados recebidos. Por esta passar as informações para que as *threads* das peças da bateria toquem o som quando preciso, atrasos na *thread* principal implicam em atrasos nas demais *threads*.

Apesar desses problemas, os resultados foram satisfatórios. Toda a parte de comunicação entre as Esps, a Raspberry e o aplicativo funcionam como planejado, a detecção de cores é precisa no ambiente correto, e os áudios tocam independentemente. Ao final, o maior problema do projeto ocorre por conta do processamento paralelo da Raspberry, não sendo suficiente para a execução em tempo real deste projeto.

5 CRONOGRAMA E CUSTOS DO PROJETO

5.1 CRONOGRAMA

Para a realização do projeto, um cronograma foi planejado no ínicio de seu desenvolvimento, e foi preenchido durante o desenvolvimento para indicar as horas gastas com cada tarefa. Para saber o estado de uma tarefa, foi criada uma legenda, indicada na figura 43.

Legenda		
Completo	Em progresso	Cancelado
Completo atrasado	A ser realizado	
Completo adiantado	Atrasado	

Figura 43: Legenda do cronograma utilizado.

A figura 44 mostra o cronograma da entrega do plano de porjeto.

									PP	E1	E2	E3	E4			E5		DF		RF
PP: Plano de Projeto detalhado																				
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/03	30/03	06/04	13/04	20/04	27/04	04/05	11/05	18/05	25/05	01/06	08/06	15/06	22/06
Confecção do plano de projeto	Todos	-	15	4,5	19,5	16														
Criação do blog	Eduardo	-	3	0,9	3,9	3														
Criação do logotipo	Gustavo	-	3	0,9	3,9	4														
Levantamento de requisitos	Todos	-	5	1,5	6,5	5														
Análise de riscos	Cesar	-	2	0,6	2,6	1,5														
Confecção do cronograma	Todos	-	6	1,8	7,8	6														
Análise de materiais e métodos	Gustavo	-	2	0,6	2,6	1,5														
Diagrama em blocos	Todos	-	5	1,5	6,5	7														
Compra dos materiais	Todos	-	2	0,6	2,6	2														
Total Time			43	12,9	55,9	46														

Figura 44: Cronograma da entrega do plano de projeto.

As figuras 45, 46, 47, 48 e 49 mostra o cronograma completo para todos os entregáveis do projeto, em ordem.

									PP	E1	E2	E3	E4			E5		DF		RF
E1: Site/blog de acompanhamento																				
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/03	30/03	06/04	13/04	20/04	27/04	04/05	11/05	18/05	25/05	01/06	08/06	15/06	22/06
Testes utilizando OpenCV em python com a Raspberry	Cesar	Eduardo/Gustavo	2,4	0,72	3,12	3														
Testes utilizando Pyaudio	Cesar	Eduardo/Gustavo	2,4	0,72	3,12	4														
Melhorias visuais no blog	Eduardo	-	1,6	0,48	2,08	1,5														
Teste das Esp32 e sensores	Todos	-	4	1,2	5,2	6														
Planejamento inicial das placas relacionadas a Esp32	Gustavo	-	4,8	1,44	6,24	3,5														
Total Time			15,2	4,56	19,76	18														

Figura 45: Cronograma completo do entregável 1.

										PP	E1	E2	E3	E4			E5		DF		RF
E2: Projeto/design e montagem da estrutura mecânica																					
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/03	30/03	06/04	13/04	20/04	27/04	04/05	11/05	18/05	25/05	01/06	08/06	15/06	22/06
Planejamento do suporte vestível de placas/fios	Todos	-	2,4	0,72	3,12	1,5														
Planejamento do suporte ao transdutor piezoelétrico no pé	Todos	-	2,4	0,72	3,12	1,5														
Planejamento das baquetas	Todos	-	2,4	0,72	3,12	2														
Confecção do suporte vestível de placas/fios	Gustavo	Eduardo	4,8	1,44	6,24	3														
Confecção do suporte ao transdutor piezoelétrico no pé	Eduardo	Cesar	4,8	1,44	6,24	6														
Confecção das baquetas	Cesar	Gustavo	4,8	1,44	6,24	1														
Atualização do blog	Eduardo	-	1,2	0,36	1,56	1														
Total Time			22,8	6,84	29,64	16														

Figura 46: Cronograma completo do entregável 2.

										PP	E1	E2	E3	E4			E5		DF		RF
E3: Projeto eletrônico e PCB. Testes de módulos isolados de hardware																					
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/03	30/03	06/04	13/04	20/04	27/04	04/05	11/05	18/05	25/05	01/06	08/06	15/06	22/06
Projeto das placas com acelerômetro e LED's	Todos	-	4	1,2	5,2	4,5														
Confecção das placas com acelerômetro e LED's	Cesar	Eduardo	6,4	1,92	8,32	9														
Testes das placas com acelerômetro e LED's	Cesar	-	3,2	0,96	4,16	4,5														
Projeto da placa de alimentação	Todos	-	4	1,2	5,2	4,5														
Confecção do suporte de bateria trocável do circuito	Gustavo	Cesar/Eduardo	2,4	0,72	3,12	1														
Confecção do circuito de detecção de carga da bateria	Gustavo	Cesar/Eduardo	4,8	1,44	6,24															
Confecção do circuito de ativação do sistema	Gustavo	Cesar/Eduardo	4,8	1,44	6,24	6														
Teste das placas de alimentação do sistema	Gustavo	-	3,2	0,96	4,16	2														
Projeto da placa para os transdutores piezoelétricos	Todos	-	4	1,2	5,2	1,5														
Confecção da placa para os transdutores piezoelétricos	Eduardo	Cesar	4	1,2	5,2	6														
Testes da placa dos transdutores piezoelétricos	Eduardo	-	3,2	0,96	4,16	1														
Atualização do blog	Eduardo	-	1,2	0,36	1,56	1														
Total Time			45,2	13,56	58,76	41														

Figura 47: Cronograma completo do entregável 3.

										PP	E1	E2	E3	E4			E5		DF		RF
E4: Projeto do software com UML. Testes de módulos individuais de software																					
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/03	30/03	06/04	13/04	20/04	27/04	04/05	11/05	18/05	25/05	01/06	08/06	15/06	22/06
Diagrama UML do código do aplicativo	Todos	-	6,4	1,92	8,32	6														
Diagrama da comunicação entre os sistemas	Todos	-	2,4	0,72	3,12	4,5														
Implementação do processamento do vídeo	Cesar	Gustavo	3,2	0,96	4,16	3														
Teste do processamento do vídeo	Cesar	Gustavo	4	1,2	5,2	3														
Busca dos exemplos de áudios que serão utilizados para os sons da bateria	Todos	-	1,6	0,48	2,08	1														
Implementação do áudio gerado pelas baquetas	Cesar	Eduardo	1,6	0,48	2,08	1,5														
Implementação do áudio gerado pelos pedais	Cesar	Eduardo	1,6	0,48	2,08	1,5														
Implementação da integração dos sons tocados	Cesar	Eduardo	4,8	1,44	6,24	5														
Teste do processamento do áudio	Cesar	Eduardo	3,2	0,96	4,16	2,5														
Integração entre o vídeo processado com o áudio a ser tocado	Cesar	Eduardo/Gustavo	2,4	0,72	3,12	3,5														
Implementação da configuração base da bateria	Eduardo	Cesar	3,2	0,96	4,16	2														
Implementação da configuração das cores dos LED's pela Esp32	Gustavo	Cesar	8	2,4	10,4	3														
Planejamento do tratamento dos acelerômetros pela Esp32	Gustavo	Cesar	4	1,2	5,2	6														
Implementação do tratamento dos acelerômetros pela Esp32	Gustavo	Eduardo	6,4	1,92	8,32	14														
Implementação da interface do aplicativo	Todos	-	19,2	5,76	24,96	27														
Implementação da criação e visualização de uma configuração de bateria no aplicativo	Eduardo	Cesar	9,6	2,88	12,48	2														
Atualização do blog	Eduardo	-	1,2	0,36	1,56	2														
Total Time			82,8	24,84	107,64	87,5														

Figura 48: Cronograma completo do entregável 4.

										PP	E1	E2	E3	E4			E5		DF		RF
E5: Integração mecânica + hardware + software																					
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/03	30/03	06/04	13/04	20/04	27/04	04/05	11/05	18/05	25/05	01/06	08/06	15/06	22/06
Implementação da comunicação entre o aplicativo e a RaspBerry Pi	Eduardo	Gustavo	6,4	1,92	8,32	12														
Implementação da comunicação RaspBerry Pi -> Esp32	Gustavo	-	4,8	1,44	6,24															
Implementação da comunicação Esp32 -> RaspBerry Pi	Gustavo	-	4,8	1,44	6,24	10														
Implementação da verificação de conexão WiFi	Gustavo		1	0,3	1,3	1,5														
Implementação da calibração no lado do aplicativo	Eduardo	Gustavo	12,8	3,84	16,64	15														
Implementação da calibração no lado da RaspBerry Pi	Cesar	Gustavo	12,8	3,84	16,64	20														
Integração do sistema de calibração	Todos	-	24	7,2	31,2	24														
Integrar as placas com seus respectivos suportes	Todos	-	1,6	0,48	2,08	16														
Atualização do blog	Eduardo	-	1,2	0,36	1,56	1														
Total Time			69,4	20,82	90,22	99,5														

Figura 49: Cronograma completo do entregável 5.

5.2 CUSTOS

A tabela 1 mostra os materiais utilizados para a confecção do projeto, junto com seus respectivos custos.

Tabela 1: Lista de materiais e seus custos.

Materiais	Quantidade	Custo Unidade (R$)	Custo Total (R$)
Raspberry Pi 2	1	250,00	250,00
Módulo Câmera	1	49,90	49,90
ESP32	2	55,00	110,00
LED RGB	2	1,09	2,18
Acelerômetro MPU6050	2	16,50	33,00
Disco piezoelétrico	2	2,79	5,58
Cabo de rede (3m)	1	8,90	8,90
Cabo de cobre duplo (1m)	7	1,40	9,80
Conector microfit 8 vias (M)	4	1,20	4,80
Conector microfit 8 vias (F)	4	1,20	4,80
Conector microfit 2 vias (M)	2	0,60	1,20
Conector microfit 2 vias (F)	2	0,60	1,20
Terminal para conector microfit (x10)	4	1,50	6,00
Cartela de resistores (330Ω)	1	0,70	0,70
Cartela de resistores (150Ω)	1	0,70	0,70
Cartela de resistores (91Ω)	1	0,70	0,70
Cartela de resistores (75Ω)	1	0,70	0,70
Cartela de resistores (1MΩ)	1	1,90	1,90
Capacitor 100nF	1	0,10	0,10
Borne 2 pólos	4	0,80	3,20
Regulador de Tensão 7805	1	2,50	2,50
Madeira Pinus 17x300x300 (mm)	1	30,00	30,00
Total	-	-	527,86

6 CONCLUSÕES

6.1 CONCLUSÕES

De maneira geral, foi possível concluir a elaboração do projeto com sucesso. O sistema é capaz de reproduzir sons escolhidos pelo usuário através de um aplicativo, ao se realizar o pressionamento dos pedais ou a batida de uma das baquetas. Além disso, o usuário é capaz também de visualizar a posição atual dos componentes através do aplicativo em um modo de calibração dedicado. Dessa forma, o protótipo elaborado atendeu a grande maioria dos requisitos propostos.

Porém, há algumas ressalvas com relação às limitações presentes nesse protótipo. A primeira delas diz respeito a limitação na quantidade de peças que o sistema suporta para ter um bom desempenho, que é inferior ao número de componentes buscado inicialmente. A segunda é ressalva é com relação ao local em que o sistema pode ser utilizado, limitado por conta da iluminação do ambiente. Por fim, a última ressalva é com relação ao próprio hardware escolhido para o projeto, dado que uma Raspberry Pi, especialmente do modelo utilizado, não é a mais adequada para a execução de um sistema em tempo real que requisite um alto desempenho.

6.2 TRABALHOS FUTUROS

Várias melhorias podem aplicadas ao projeto atual, visando deixar o protótipo ainda mais perto de uma bateria real. Para amenizar a limitação relacionada ao ambiente em que os AR Drums podem ser tocados, pode ser implementada uma maneira de customizar as cores das baquetas. Isso poderia ser controlado via aplicativo, ou até mesmo ser feito de maneira automática, utilizando a câmera para verificar o ambiente atual do usuário para alterar as cores das baquetas. Ainda no mesmo problema, outros tipos de leds poderiam ser usados, tendo um brilho mais alto para que o sistema possa ser tocado em locais mais iluminados.

A configuração das baterias via aplicativo também pode ser mais completa. Adicionando uma quantidade maior de opções de peças, com uma prévia da peça a ser

modificada sendo tocada no próprio aplicativo. As posições de cada peça também poderiam ser mais customizáveis, alterando não só a posição como o tamanho de cada peça.

Por fim, a melhoria mais crucial ao projeto é ter um hardware mais adequado para sua execução. Os maiores problemas do projeto ocorreram por conta do desempenho do hardware utilizado. Assim, ter um hardware mais apropriado para sua execução iria facilitar a implementação de outras melhorias, como também melhorar as implementações já existentes.

REFERÊNCIAS

ARDUINO. **WiFi Library for Arduino**. 2022. Disponível em: <https://www.arduino.cc/reference/en/libraries/wifi/>. Acesso em: 19 de junho de 2022.

EXPRESSIF. **esp_wifi library**. 2021. Disponível em: <https://github.com/espressif/esp-idf/tree/master/components/esp_wifi>. Acesso em: 19 de junho de 2022.

EXPRESSIF. **ESP32 Series: Datasheet**. 2022. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>. Acesso em: 13 de junho de 2022.

FERREIRA, A. L. **Transdutor Piezoelétrico**. 2017. Disponível em: <http://www.squids.com.br/arduino/index.php/hardware/componentes-eletronicos/119-disco-piezoeletrico>. Acesso em: 21 de junho de 2022.

FLUTTER. **Flutter architectural overview**. 2020. Disponível em: <https://docs.flutter.dev/resources/architectural-overview>. Acesso em: 21 de junho de 2022.

INVENSENSE. **MPU-6000 and MPU-6050 Product Specification Rev. 3.4**. 2013. Disponível em: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>. Acesso em: 13 de junho de 2022.

KUROSE, J.; ROSS, K. **Computer Networking: A Top-Down Approach**. [S.l.]: Bookman Editora, 2021. ISBN 9788582605592.

NOLLE, T. **reactive programming**. 2021. Disponível em: <https://www.techtarget.com/searchapparchitecture/definition/reactive-programming>. Acesso em: 21 de junho de 2022.

OPENCV. **Open Source Computer Vision**. 2022. Disponível em: <https://docs.opencv.org/4.x/index.html>. Acesso em: 14 de junho de 2022.

PHAM, H. **Pyaudio**. 2006. Disponível em: <http://people.csail.mit.edu/hubert/pyaudio/>. Acesso em: 13 de junho de 2022.

PYTHON. **wave — Read and write WAV files**. 2020. Disponível em: <https://docs.python.org/3/library/wave.html>. Acesso em: 13 de junho de 2022.

RANDOMNERDTUTORIALS. **ESP32 Pinout Reference: Which GPIO pins should you use?** 2020. Disponível em: <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>. Acesso em: 21 de junho de 2022.

RASPBERRY. **Introducing the Raspberry Pi Cameras**. 2013. Disponível em: <https://www.raspberrypi.com/documentation/accessories/camera.html>. Acesso em: 13 de junho de 2022.

SUEIRO, D. **Raspberry Pi 2 Modelo B**. 2015. Disponível em: <https://www.embarcados.com.br/lancamento-raspberry-pi-2-model-b>. Acesso em: 13 de junho de 2022.

SWAROOP. **What is Serial Communication and How it works?** 2018. Disponível em: <https://www.codrey.com/embedded-systems/serial-communication-basics/>. Acesso em: 13 de junho de 2022.

VIDADESILíCIO. **Módulo GY-521 MPU6050 – Acelerômetro e Giroscópio**. 2020. Disponível em: <https://www.vidadesilicio.com.br/produto/modulo-gy-521-acelerometro-giroscopio/>. Acesso em: 21 de junho de 2022.

WIKIPEDIA. **WAV**. 2008. Disponível em: <https://pt.wikipedia.org/wiki/WAV>. Acesso em: 13 de junho de 2022.

WPILIB. **Thresholding an Image**. 2022. Disponível em: <https://docs.wpilib.org/en/stable/docs/software/vision-processing/wpilibpi/image-thresholding.html>. Acesso em: 14 de junho de 2022.