UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA – DAELN
CURSO DE ENGENHARIA DE COMPUTAÇÃO

JOÃO VICTOR LASKOSKI, LUIS CAMILO JUSSIANI MOREIRA, LUCIANO BONZATTO JUNIOR

TALKER V-EYE: DISPOSITIVO PARA AUXILIAR NA LOCOMOÇÃO DE DEFICIENTES VISUAIS

OFICINA DE INTEGRAÇÃO 2 - RELATÓRIO TÉCNICO

CURITIBA
2022

JOÃO VICTOR LASKOSKI, LUIS CAMILO JUSSIANI MOREIRA, LUCIANO BONZATTO JUNIOR

TALKER V-EYE: DISPOSITIVO PARA AUXILIAR NA LOCOMOÇÃO DE DEFICIENTES VISUAIS

Relatório Final da disciplina Oficinas de Integração 2 (EEX22), do curso de Engenharia de Computação, apresentado aos professores que ministram a mesma na Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção da aprovação da disciplina.

Orientador: Prof. Dr. Cesar Manuel Vargas Benítez, Prof. Dr. Heitor S. Lopes

CURITIBA
2022

Dedicamos este trabalho à todas as pessoas que possuem deficiência visual, para que possam ter mais liberdade e segurança no dia a dia, e que tenham maior desfruto da vida.

AGRADECIMENTOS

Primeiramente ao Laboratório de Engenharia de Sistema Computacionais (LESC) pela disponibilidade dos equipamentos utilizados. Deve-se agradecimento especial também aos professores César Manuel Vargas Benítez e Heitor Silvério Lopes por todo auxílio prestado durante o desenvolvimento do projeto.

RESUMO

. Talker V-EYE: Dispositivo para auxiliar na locomoção de deficientes visuais. 2022. 37 f. OFICINA DE INTEGRAÇÃO 2 - RELATÓRIO TÉCNICO – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2022.

O presente documento relata o desenvolvimento e funcionamento do protótipo do projeto *Talker V-EYE*, que tem como objetivo auxiliar pessoas com deficiência visual em sua locomoção diária, avisando previamente os obstáculos que possam atingir o usuário. O *Talker V-EYE* tem como funcionalidade detectar objetos, informando para o usuário a posição do obstáculo mais próximo através de vibrações. Além disso, de modo opcional, será possível identificar o objeto, informando ao usuário o nome do objeto mais próximo detectado ou poderá ser informando todos os nomes dos objetos presentes na frente do mesmo; sendo que essas informações serão através de áudio. Ainda mais, todo sistema pode ser configurado através de um aplicativo para smartphone.

Palavras-chave: Deficiência Visual. Detecção. Obstáculo. Identificação. Vibração.

ABSTRACT

. Talker V-EYE: Dispositivo para auxiliar na locomoção de deficientes visuais. 2022. 37 f. OFICINA DE INTEGRAÇÃO 2 - RELATÓRIO TÉCNICO – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2022.

This document reports the development and operation of the prototype of the Talker V-EYE project, which aims to help people with visual impairments in their daily locomotion, giving warning of obstacles that may reach the user. The Talker V-EYE has the functionality to detect objects, informing the user of the position of the nearest obstacle through vibrations. Additionally, optionally, it will be possible to identify the object, informing the user of the name of the closest object detected or it can be informing all the names of the objects present in front of it, and this information will be through audio. Furthermore, the entire system can be configured via a smartphone app.

Keywords: Visual impairment. Detection. Obstacle. Identification. Vibration

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Segundo os dados do censo de 2010 do Instituto Brasileiro de Geografia e Estatística (IBGE), há mais de 6,5 milhões de pessoas com deficiência visual no Brasil, sendo destas 582 mil com deficiência total (OLIVEIRA, 2012). Além disso, existem apenas 200 cão-guias no Brasil (VENTURA, 2012), representando uma quantidade baixíssima quando comparado a quantidade de pessoas com deficiência visual, logo sendo um auxiliar com pequena oferta. Além do uso de cão-guia, temos como dispositivo auxiliar a bengala, que é adotada amplamente pelos deficientes visuais. No entanto, é possível perceber que o deficiente visual possui poucos recursos para o auxiliar espacialmente, principalmente em regiões acima do abdome.

Sendo assim, um indivíduo com deficiências na visão, enfrenta grandes dificuldades para se locomover com segurança, seja tanto em locais públicos como parques e museus, quanto shoppings e afins.

1.2 OBJETIVOS

Dessa forma, o dispositivo proposto tem o objetivo de, juntamente com a utilização de bengala, cão guia ou outro dispositivo, auxiliar uma pessoa com deficiência visual. Principalmente, impedindo que a pessoa tenha acidentes perante obstáculos na região acima do abdome enquanto se locomove ou mantendo-se inerte, visto que são regiões que muitos equipamentos não evitariam uma eventual colisão. O modo como o sistema informa obstáculos detectados ocorre por meio de um sistema de vibrações.

Para oferecer maior imersão para o usuário, o sistema propõe informar o nome do obstáculo mais proximo ou de todos os objetos identificados na frente do usuário, fornecendo assim informações do ambiente em questão, para que o mesmo possa ter maior conhecimento do que está ao seu redor.

Todas as funções do sistema em funcionamento são controladas por botões em uma pulseira do usuário, que também é provida de um sensor óptico e portanto também é responsável por detectar obstáculos a curtas distâncias dependendo da região em que o usuário aponta o sensor.

1.2.1 OBJETIVOS ESPECÍFICOS

1.2.1.1 REQUISITOS FUNCIONAIS

Durante o projeto do sistema, foram definidos os seguintes requisitos funcionais:

- RF001: O sistema estará contido em uma mochila padrão de tamanho médio, porém necessitando de algumas modificações para portar a câmera RGBD e os motores de vibração, além de conter os cabos que conectam o microcontrolador aos periféricos sem trazer incômodo ao usuário.
- RF002: O sistema deve possuir um procedimento de calibração, por um aplicativo de celular, em que é configurado a profundidade máxima e mínima em que o sistema detecta objetos e a altura da câmera em relação ao chão.
- RF003: O sistema deve informar ao usuário o(s) quadrante(s) que o obstáculo detectado, com auxílio da câmera RGBD, pertence. De maneira que cada motor acionado simboliza um quadrante, variando a intensidade da vibração de acordo com a distância do obstáculo.
- RF004: O sistema deve possuir um sensor de distância óptico adicional, junto de um motor de vibração, localizados no pulso do usuário. De maneira que a intensidade de vibração do motor varia com a distância medida pelo sensor.
- RF005: A detecção com o auxílio da câmera RGBD, deve sinalizar objetos a uma distância maior que 1 m e menor que 3 m em relação à ela. E a detecção com o auxílio do sensor óptico deve sinalizar medições de até 1 m.
- RF006: O sistema pode informar através de áudio ou o nome do objeto mais próximo, caso ele seja identificado, ou de todos os objetos identificados na imagem. Caso não seja identificado, no primeiro caso retorna "obstáculo não identificado" e no segundo "nenhum objeto identificado".
- RF007: O sistema deve estar conectado a uma rede de internet, para que seja utilizado os *softwares* para identificação de objetos(*API Vision*) e conversão de texto em áudio (*API Text to Speech*). Caso o sistema não esteja conectado à internet, a parte da identificação dos objetos estará inabilitada, funcionando apenas a detecção dos obstáculos.
- RF008: A resposta ao detectar um obstáculo e informar o usuário pelos motores de vibração, não pode passar de 1 segundo.
- RF009: O sistema deve estar conectado em um alto-falante, para reproduzir áudios.

1.2.1.2 REQUISITOS NÃO FUNCIONAIS

Durante o projeto do sistema, foram definidos os seguintes requisitos não funcionais:

- RNF001: A altura da câmera em relação ao chão, dada na calibração, é utilizada para o sistema não sinalizar o chão como obstáculo.
- RNF002: O aplicativo utilizado para aplicar configurações no sistema será desenvolvido apenas para Android utilizando a linguagem Java.
- RNF003: A identificação de objetos em imagens utilizará a *API Vision* e a conversão de texto em áudio será feita pela *API Text to Speech*, e ambas serão utilizadas em um programa em Java, necessitando conexão à internet.
- RNF004: A reprodução de áudio será feita no mesmo *software* que conterá o uso das duas APIs citadas, sendo reproduzido em um alto-falante. O áudio é gerado através da *API Text to Speech* citada anteriormente.
- RNF005: As medições de distâncias serão feitas através de uma câmera RGBD.
- RNF006: As medições de distância pela pulseira será através de um sensor óptico.
- RNF007: A comunicação entre o smartphone e o *software* em Java e a comunicação entre o *software* em C++ e o *software* em Java será feita utilizando *Sockets* TCP/IP.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTEL REALSENSE D435

A Intel Realsense D435 é uma câmera de profundidade que é alimentada por USB e possui sensores de profundidade, que conseguem medir a profundidade em uma resolução de 1280 x 720 tendo um alcance ideal de 0,3 m a 3 m (INTEL, 2018), além de funcionar em ambientes internos e externos, o que aumenta a flexibilidade do uso.

Figura 1 – Intel Realsense 435

Fonte: (INTEL, 2018)

2.2 LIDAR-Lite v3

O LIDAR-Lite v3 é um sensor de medição de distância óptico de alto desempenho e com baixo consumo de energia, sendo utilizado para aplicações com drones, robôs ou veículos não tripulados, comumente utilizado na detecção de obstáculos. O sensor funciona com base em um *laser* que possui um alcance máximo de 40 m (GARMIN, 2016) e requer uma fonte de energia e um microcontrolador externo com uma aplicação em andamento.

Figura 2 – LIDAR-Lite v3

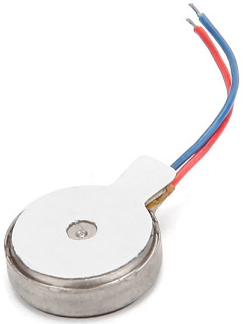
Fonte: (GARMIN, 2016)

O LIDAR-Lite v3 pode ser conectado com um sistema computacional por meio de uma interface I2C.

2.3 MOTORES DE VIBRAÇÃO

O motor de vibração utilizado no projeto foi do tipo 1027 que atua sobre uma tensão de 2,5V - 4V, tendo uma velocidade de rotação máxima de 9000 RPM (FILIPEFLOP, 2018). Esse tipo de motor foi utilizado uma vez que o movimento responsável pela rotação é encapsulado, não havendo assim partes externas em movimento.

Figura 3 – Motor de vibração 1027



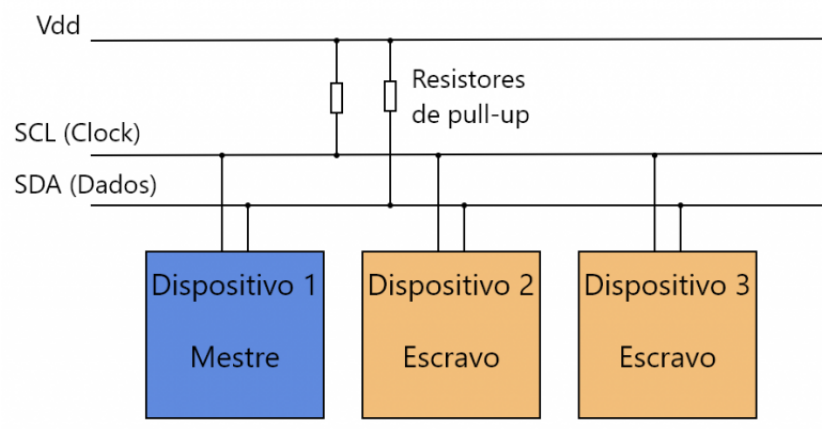
Fonte: (FILIPEFLOP, 2018)

2.4 COMUNICAÇÃO POR I2C

A comunicação entre o LIDAR-Lite v3 e a Raspeberry Pi 4B é feita através do protocolo I2C, que é um protocolo serial e síncrono desenvolvido pela Philips, tendo como principais vantagens a simplicidade e o baixo custo, e, como desvantagem, a velocidade (GUIMARAES, 2018).

O protocolo conta com dois canais de comunicação: O *Serial Clock* (SCL) e o *Serial Data* (SDA). Além disso, é necessário que exista pelo menos um dispositivo "mestre" e um "escravo", onde o mestre é responsável por gerar o *clock* e inicializar a comunicação enquanto que o escravo recebe os dados e responde ao ser chamado. A Figura 4 ilustra a comunicação I2C com um mestre e dois escravos.

Figura 4 – Ilustração da comunicação I2C

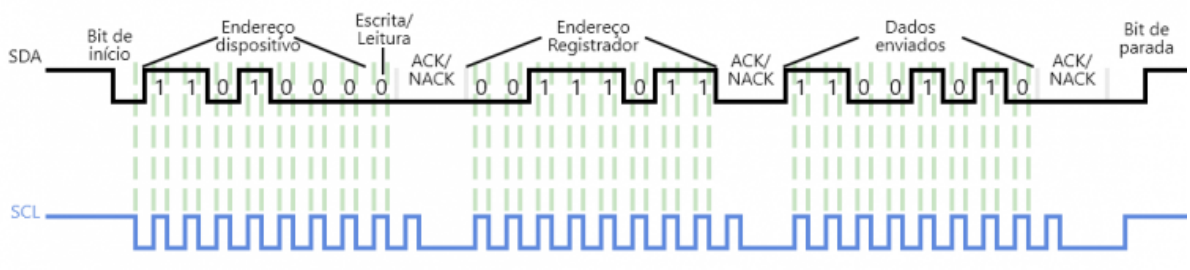


Fonte: (GUIMARAES, 2018)

Na comunicação, podemos classificar basicamente cinco tipos de bits: bit de início (marca o início da comunicação), bits de dado (pode ser bits de endereço, de valor de escrita ou de valor de leitura), bit de leitura/escrita (indica se o mestre vai escrever em algum registrador de outro dispotivo ou se vai ler o valor de algum registrador de algum dispositivo), bit de *Acknowldge* (ACK) (indica que os dados foram recebidos corretamente) ou *Not Acknowldge* (NACK) (indica que os dados não foram recebidos corretamente) e bit de parada (sinaliza o final da comunicação).

A Figura 5 ilustra um exemplo de escrita utilizando o protocolo I2C. No exemplo, o dispositivo mestre escreve os bits 11001010 no registrador de endereço 00111011 do dispositivo de endereço 11010000.

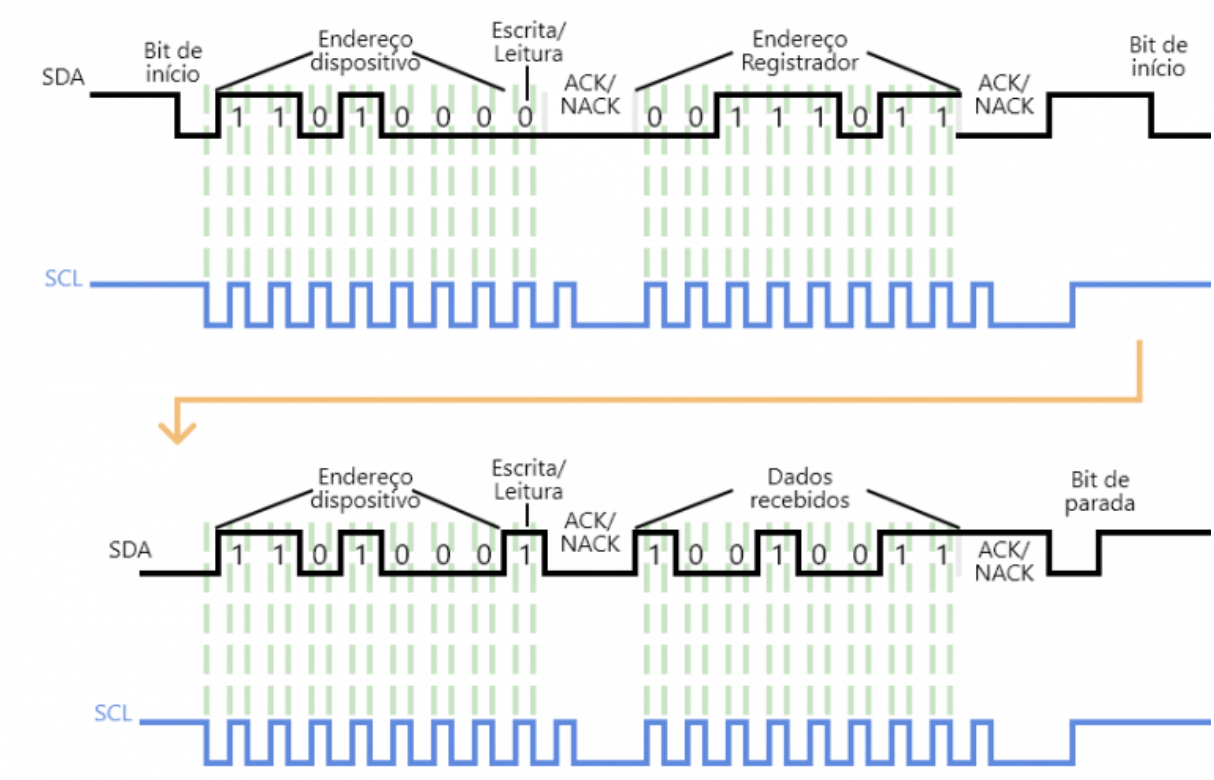
Figura 5 – Ilustração de um exemplo de comunicação I2C para a escrita de um byte em um registrador de um dispositivo (escravo)



Fonte: (GUIMARAES, 2018)

Por outro lado, a Figura 6 ilustra um exemplo de leitura utilizando o protocolo I2C. No exemplo, o dispositivo mestre lê o valor 10010011 do registrador de endereço 00111011 do dispositivo de endereço 1101000.

Figura 6 – Ilustração de um exemplo de comunicação I2C para a leitura de um byte em um registrador de um dispositivo (escravo)



Fonte: (GUIMARAES, 2018)

2.5 C++

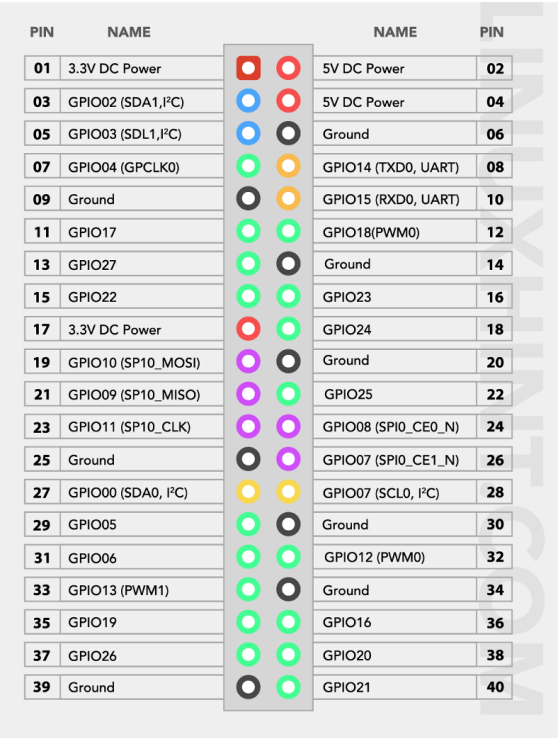
C++ é uma linguagem de programação que surgiu na década de 80 e se tornou uma das linguagens de programação comerciais mais populares. A linguagem C++ é uma linguagem multi-paradigma, ou seja, suporta mais do que um estilo de programação, e é considerada uma linguagem de uso geral (CANELLE; ARAúJO, 2012).

2.6 RASPBERRY PI 4B

A Raspberry pi 4B é um micro-computador integrado em placa única, sendo sua utilização conveniente com estudos básicos para computação ou como utilização de um computador para tarefas básicas por menor custo ao comparar com computadores convencionais. O modelo utilizado é munido de um processador quad-core Cortex A-72 64-bit, possuindo *clock* de 1.5Ghz e memória RAM de 8GB DDR4. Para conectividade, possui um adaptador *wi-fi* (2.4GGHz e 5GHz), bluetooth 5.0 BLE, entradas HDMI, USB, Ethernet, áudio e vídeo. Por fim, possui 40 pinos de GPIO, podendo ter funções alternativas os mesmos.

Abaixo está a esquematização dos pinos GPIOs presentes na Raspberry pi 4B+:

Figura 7 – Disposição dos pinos GPIO no micro-computador



Fonte: (KHAN, 2022)

2.7 COMUNICAÇÃO POR SOCKETS

Sendo inicialmente elaborado para o sistema operacional 4.2BSD, os *sockets* foram um meio de comunicação entre processos (*Interprocess communication*), no qual os processos podem ou não estar no mesmo hardware. Logo, pode-se tomar o arranjo do *socket* como uma API, abstraindo conceitos de comunicação de redes, como a pilha do protocolo TCP/IP.

Figura 8 – Modelo OSI e TCP/IP

Fonte: (PANTUZA, 2017)

Portanto, tomando os modelos de protocolo como o OSI e o TCP/IP, descritos na imagem abaixo, o *socket* atua entre a aplicação e o transporte, o qual faz tal ligação entre a aplicação e a rede de forma transparente.

Na questão de desenvolvimento de softwares, utilizando *sockets*, uma arquitetura amplamente implementada é a Cliente/Servidor, no qual um programa atuará como cliente da comunicação e o outro como o servidor. Vale ressaltar, que os tipos de *sockets* mais utilizados são o TCP e UDP, em que o *socket* TCP garante a ordem dos pacotes e trata as falhas ocorridas, sendo um meio de transporte mais confiável; porém mais lento devido aos mecanismos relatados. Em contrapartida, o *socket* UDP não considera a ordem do pacote, como não trata as falhas que podem ocorrer e devido a não possuir esses mecanismos, possui maior velocidade na comunicação ao comparar com o TCP.

2.8 THREADS

Thread é um fluxo de execução independente, no qual um processo pode conter uma ou mais *threads* no código, as quais podem compartilhar recursos entre si se estiverem no mesmo processo. Cada *thread* tem como característica um contexto local, o qual é composto por registradores da CPU e uma área da pilha de memória, para que possa ser armazenada variáveis e executar funções. Além disso, há também o código em execução.

2.9 APIs Google

A API tem como significado ser uma interface de programação entre aplicações, possuindo um conjunto de definições e protocolos para que possa ser feito integração entre softwares. Há inúmeros tipos de APIs, como SOAP, WebSocket, REST, entre outras, das quais API REST são as utilizadas pela google, além da biblioteca de cliente, que torna o uso das APIs mais amigável.

Desse modo, as APIs da google interagem com diversos dados e tipos de aplicações, como armazenamento, banco de dados, configuração de DNS, análise de dados, *machine learning*, entre inúmeros outros.

2.10 JAVA

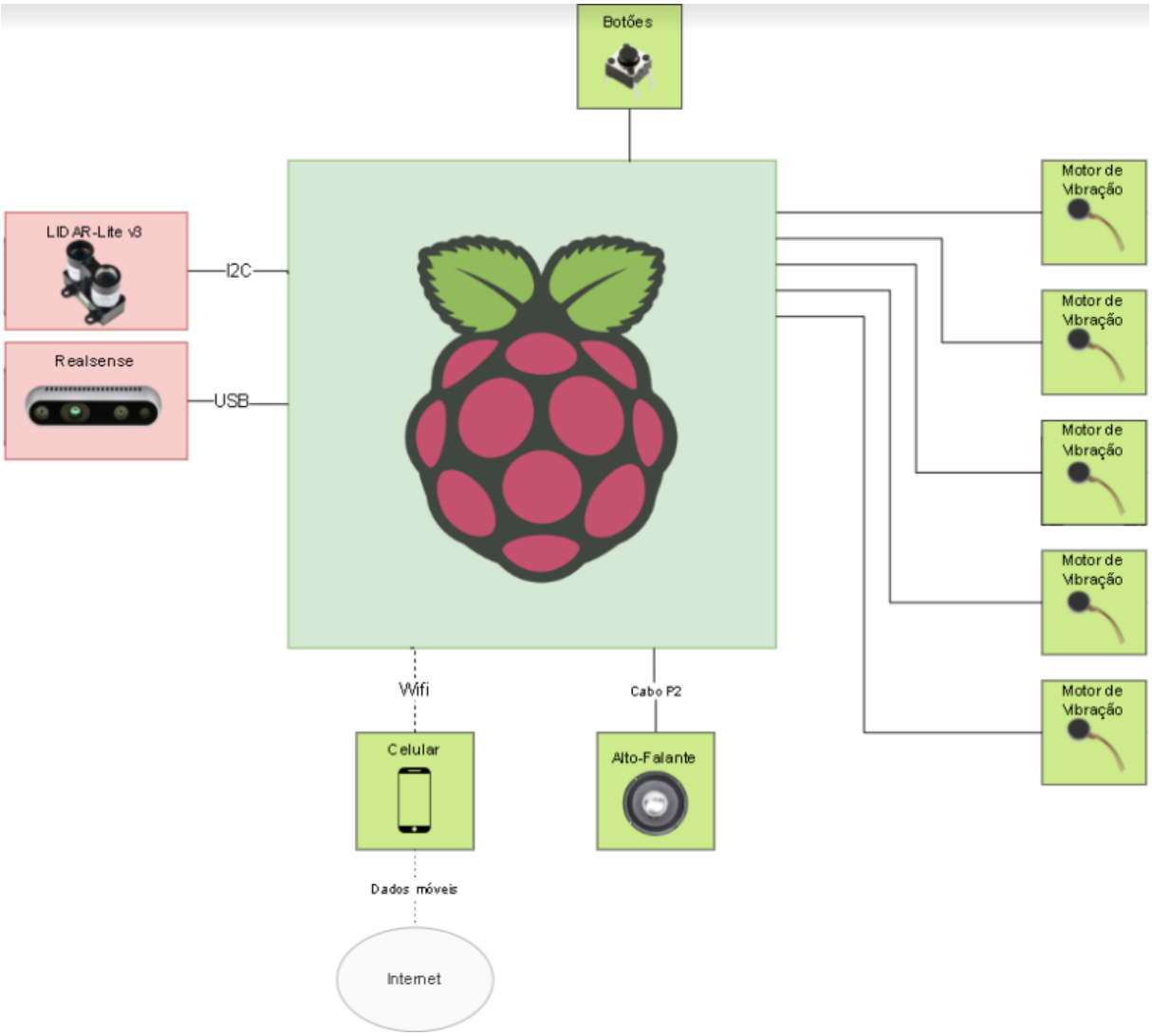
Java é uma linguagem de programação orientada a objetos, a qual foi inicialmente desenvolvida durante a década de 90. A linguagem Java diferencia de algumas outras linguagens devido ao seu módulo de compilação, não sendo direto para o código nativo, mas tendo uma máquina virtual (JVM) que interpreta o código compilado (*bytecode*) e traduz para o código nativo. Tal linguagem possui vasto uso devido a alta disponibilidade de bibliotecas e APIs, alta portabilidade, simplicidade na criação de programas distribuídos e multitarefas, uso de recursos de rede, entre outras caractísticas da linguagem.

3 METODOLOGIA

3.1 VISÃO GERAL

Todo o sistema do *Talker V-EYE* é projetado em torno do microcontrolador Raspberry pi 4B, como ilustra o diagrama de blocos da Figura 9.

Figura 9 – Diagrama de blocos do *Talker V-EYE*



Fonte: Autoria Própria (2022)

Assim, com os pinos de GPIO configurados e com a programação via software do PWM para variar a intensidade de vibração dos motores, foi possível construir o sistema de vibração do *Talker V-EYE*, que compõe uma parte da saída do projeto.

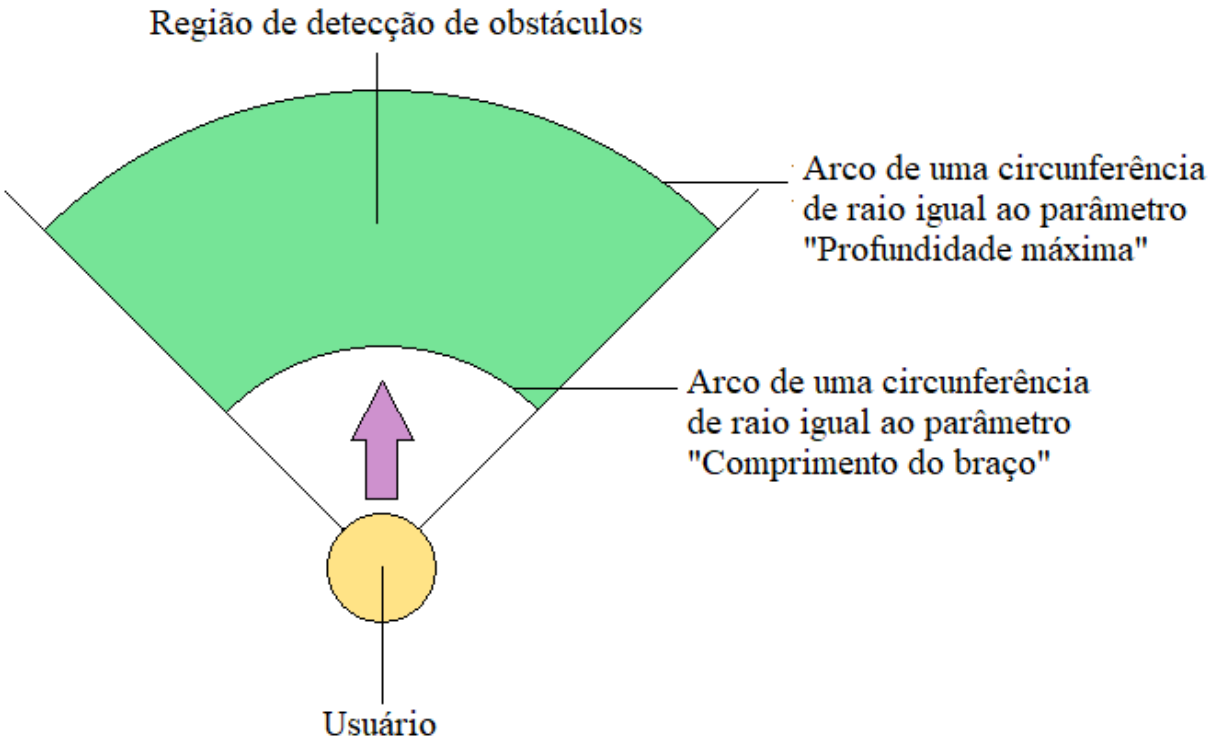
Referente as entradas do usuário, também foram configurados pinos GPIO para os botões do sistema presentes na pulseira, o que possibilitou a programação nos botões das funções de identificação de obstáculos e para habilitar e desabilitar o sistema.

Tanto o LIDAR-Lite v3 quanto a Realsense d435, produzem dados de entrada para o sistema que serão utilizados pelos algoritmos de detecção de obstáculos, para que então o sistema possa gerar as saídas na forma de vibração.

Além disso, por intermédio de um cabo p2, o sistema conta com um alto-falante responsável por reproduzir os sons dos nomes dos objetos identificados após o usuário clicar um dos botões de indentificação.

Por fim, ainda como entrada do sistema, o usuário pode alterar o campo de ação da detecção de obstáculos do sistema, definindo um valor mínimo e um máximo onde os obstáculos serão detectados. Essa configuração é feita por meio do aplicativo do *Talker V-EYE*. A Figura 10 ilustra melhor o campo de atuação do *Talker V-EYE* para a detecção de obstáculos de acordo com as entradas do usuário para os parâmetros "Comprimento de braço" e "Profundidade máxima".

Figura 10 – Ilustração da área de detecção do *Talker V-EYE*: A ilustração representa uma visão acima do usuário. A região em verde indica a área em que o sistema detectará obstáculos.

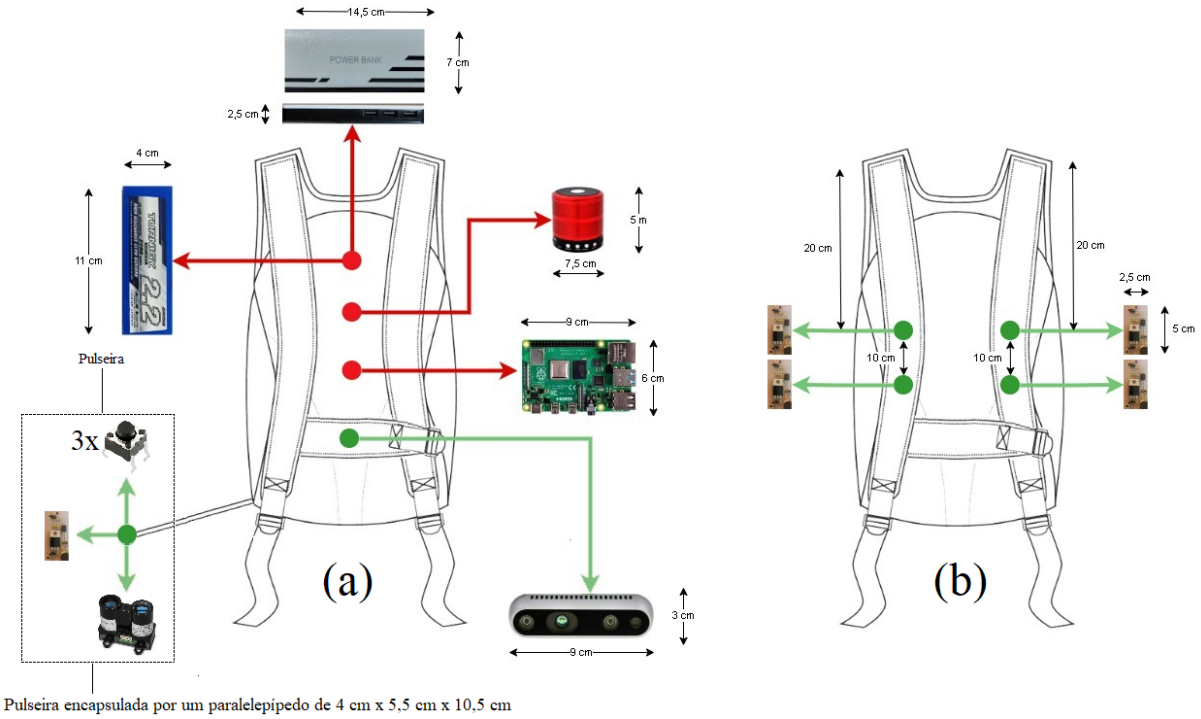


Fonte: Autoria Própria (2022)

3.2 PROJETO MECÂNICO

Todo o sistema do *Talker V-EYE* foi acoplado em uma mochila, o que possibilitou modelar o sistema de forma que a maior parte dos componentes eletrônicos ficassem no seu interior, com excessão da câmera RGBD e da pulseira, o que facilitou a proteção dos componentes, além de tornar o protótipo mais discreto. A Figura 11 ilustra o projeto mecânico do *Talker V-EYE*, destacando as dimensões dos componentes e a disposição dos mesmos na estrutura (mochila), onde os componentes indicados por setas vermelhas se encontram dentro da mochila, enquanto que os componentes indicados por setas verdes se encontram fora.

Figura 11 – Ilustração do projeto mecânico do *Talker V-EYE*: Organização dos componentes na estrutura com excessão dos motores de vibração das alças (a) e a disposição dos motores de vibração nas alças da mochila (b).

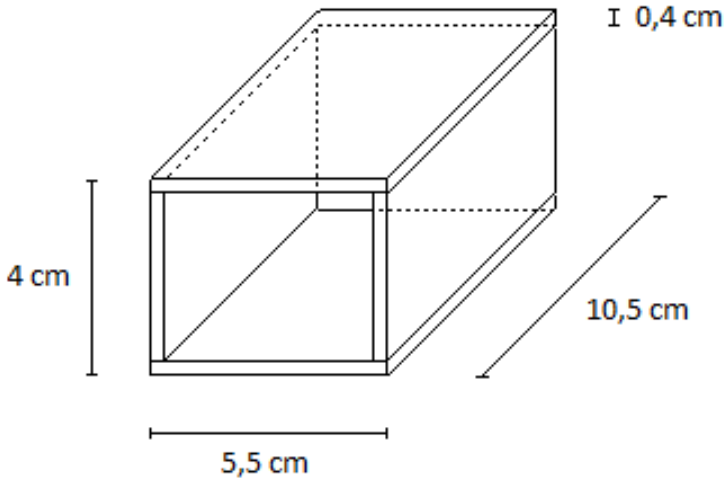


Fonte: Autoria Própria (2022)

A adaptação de uma mochila comum para comportar o sistema consistiu em adicionar quatro pequenos bolsos na parte interna das alças para a fixação dos motores, além de adicionar mais uma alça para fixar a câmera RGBD de modo que se localize no peito do usuário. O resultado da adaptação para os motores nas alças e da adptação da alça para a câmera RGBD se encontram na Figura 14 e Figura 15, respectivamente.

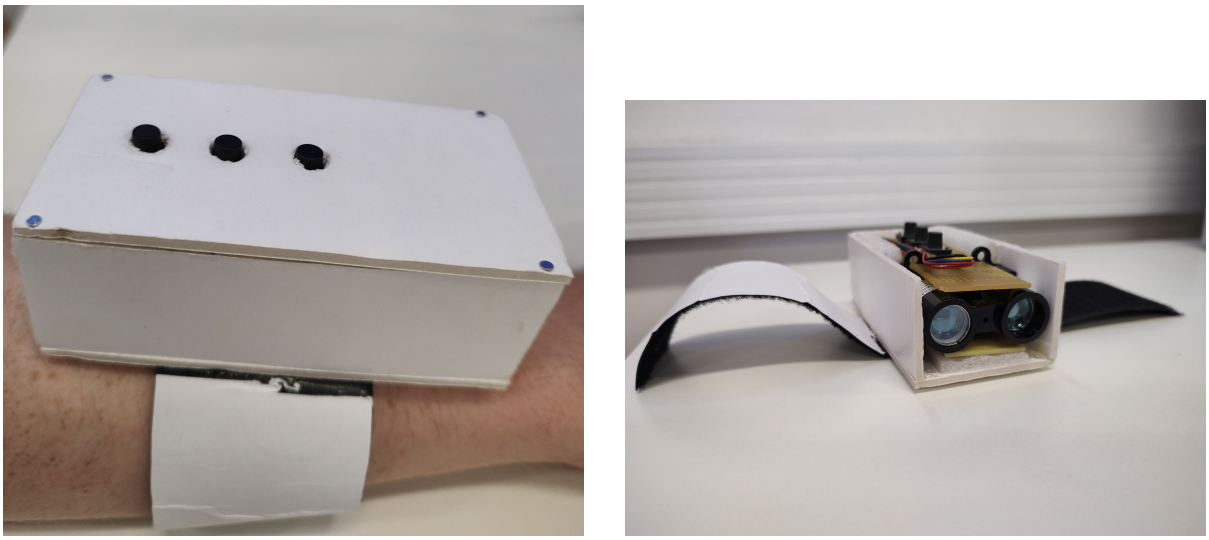
A respeito da construção da pulseira, foi decido montar um encapsulamento feito de *foam paper* em que os botões ficassem expostos e a parte da frente dele ficasse descoberto para não interferir na medição do sensor óptico. as dimensões do encapsulamento estão na Figura 12. Além disso, como a conexão com o microcontrolador é via cabo, foi necessário monta-lo soldando os conectores.

Figura 12 – Projeto do emcapsulamento da pulseira.



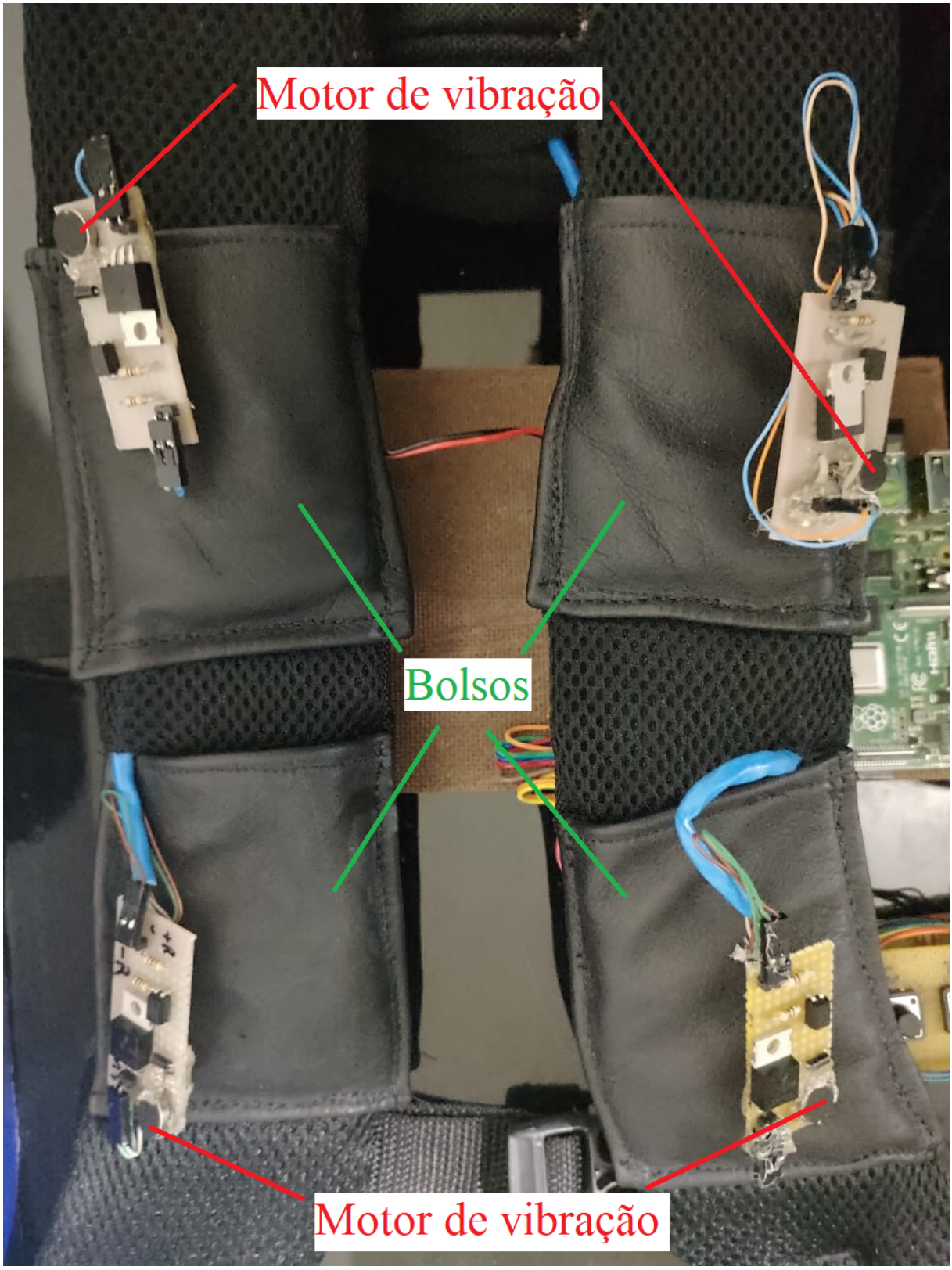
Fonte: Autoria Própria (2022)

Figura 13 – Emcapsulamento da pulseira.



Fonte: Autoria Própria (2022)

Figura 14 – Adaptação das alças da mochila para o suporte dos motores de vibração.



Fonte: Autoria Própria (2022)

Figura 15 – Adaptação da mochila para o suporte da câmera RGBD.

Fonte: Autoria Própria (2022)

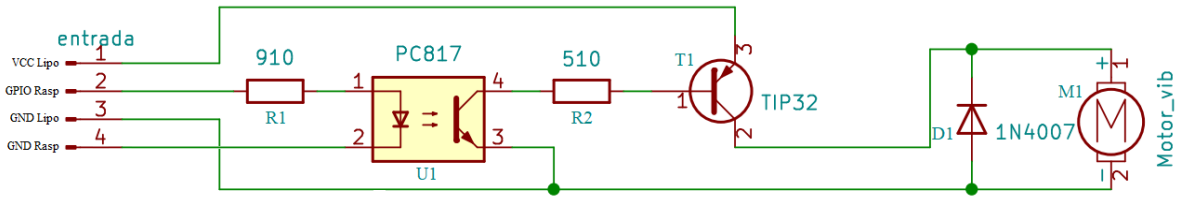
3.3 PROJETO DE HARDWARE

Todos os módulos de hardware estão ligados no microcontrolador por meio de portas GPIO no caso dos botões e dos motores de vibração, comunicação I2C no caso do sensor óptico, cabo P2 no caso do alto-falante e USB no caso da câmera RGBD, assim como ilustra o diagrama de blocos da Figura 9.

A alimentação do sistema é composta por duas baterias, responsáveis por alimentar partes distintas do sistema. Uma *power bank* alimenta o microcontrolador e alguns periféricos, enquanto que uma bateria LiPo de com o auxílio de um *power module* fornece 5 V para a alimentação exclusiva dos motores de vibração.

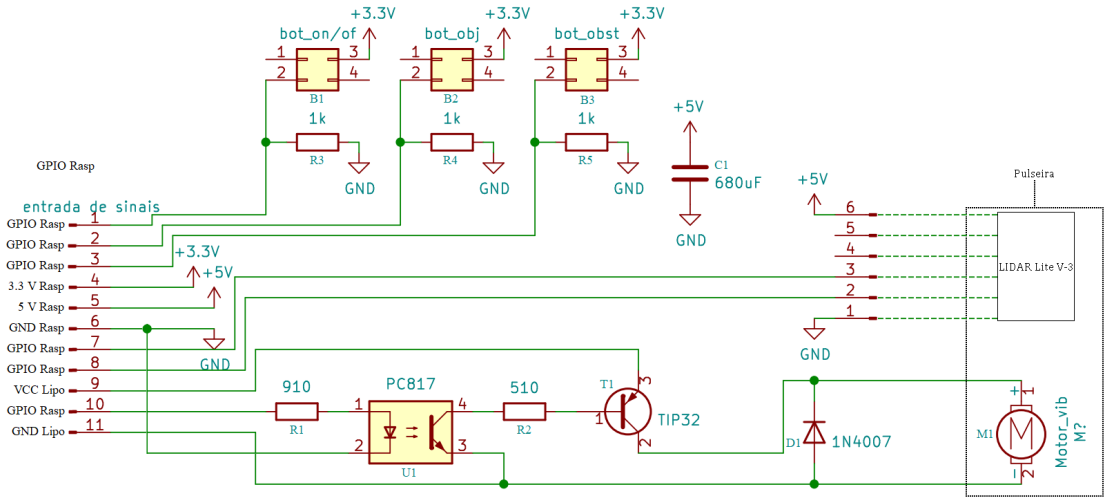
Todos os componentes apresentados no diagrama de blocos da Figura 9 foram ligados diretamente no microcontrolador, com excessão dos motores de vibração e dos botões, em que foi necessário desenvolver circuitos apropriados que estão representados nos esquemáticos da Figura 16 e Figura 17, para fazer a interface com o microcontrolador. No caso dos motores foi necessário desenvolver um circuito em que a alimentação dos mesmos seja feita com uma fonte externa, além de isolar eletricamente o microcontrolador por questão de segurança.

Figura 16 – Esquemático das placas dos motores de vibração



Fonte: Autoria Própria (2022)

Figura 17 – Esquemático da placa dos botões



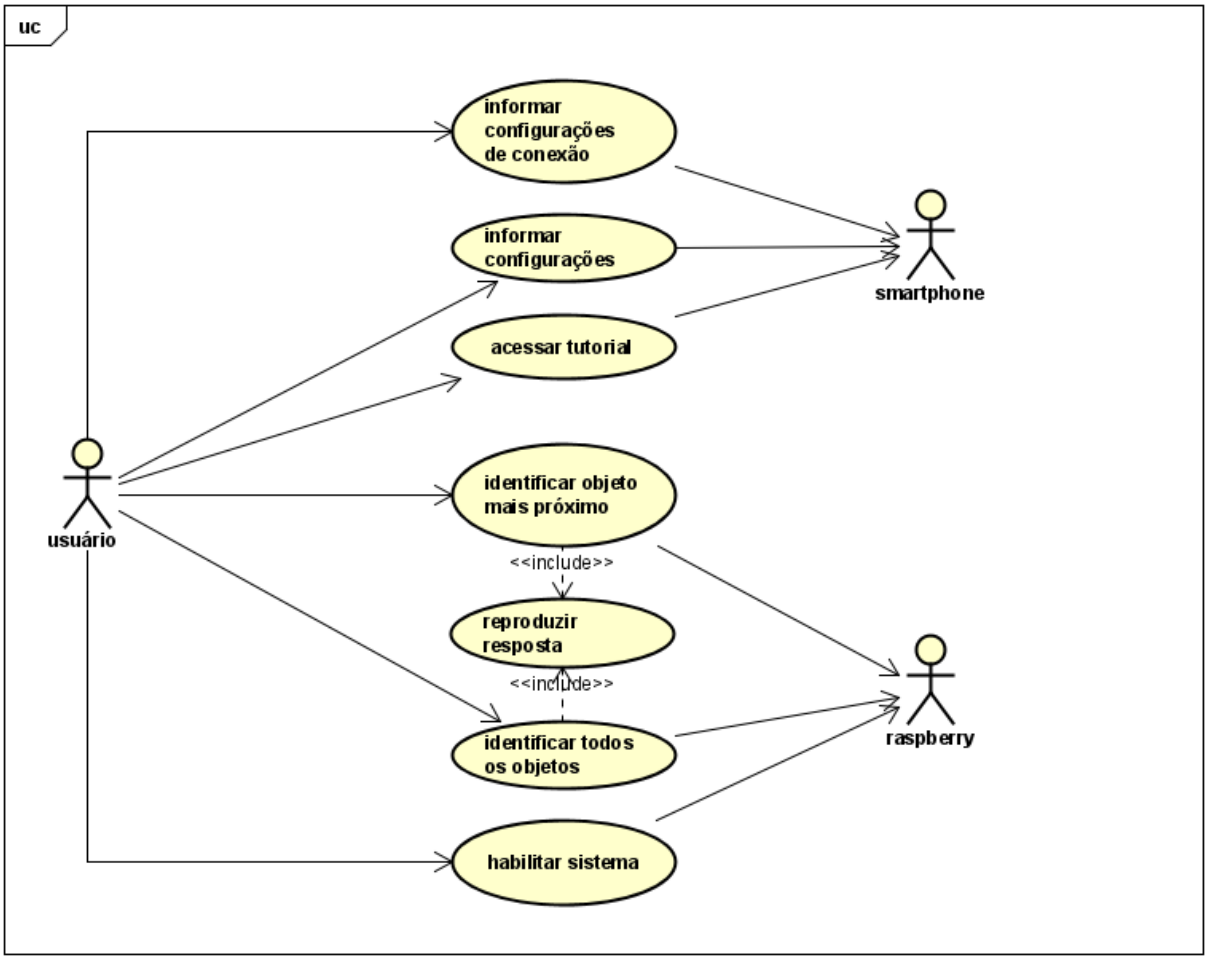
Fonte: Autoria Própria (2022)

3.4 PROJETO DE SOFTWARE

Primeiramente foi levantando os requisitos necessários para o funcionamento do projeto, que englobam de forma geral a configuração do sistema, detecção e identificação de objetos.

Desse modo, foi elaborado o diagrama de casos de uso, tomando os tópicos de configuração e identificação, que são as ações das quais o usuário (deficiente visual) pode interagir. Sendo assim, tendo o usuário para executar as configurações e decidir o momento e qual será o modo de identificação.

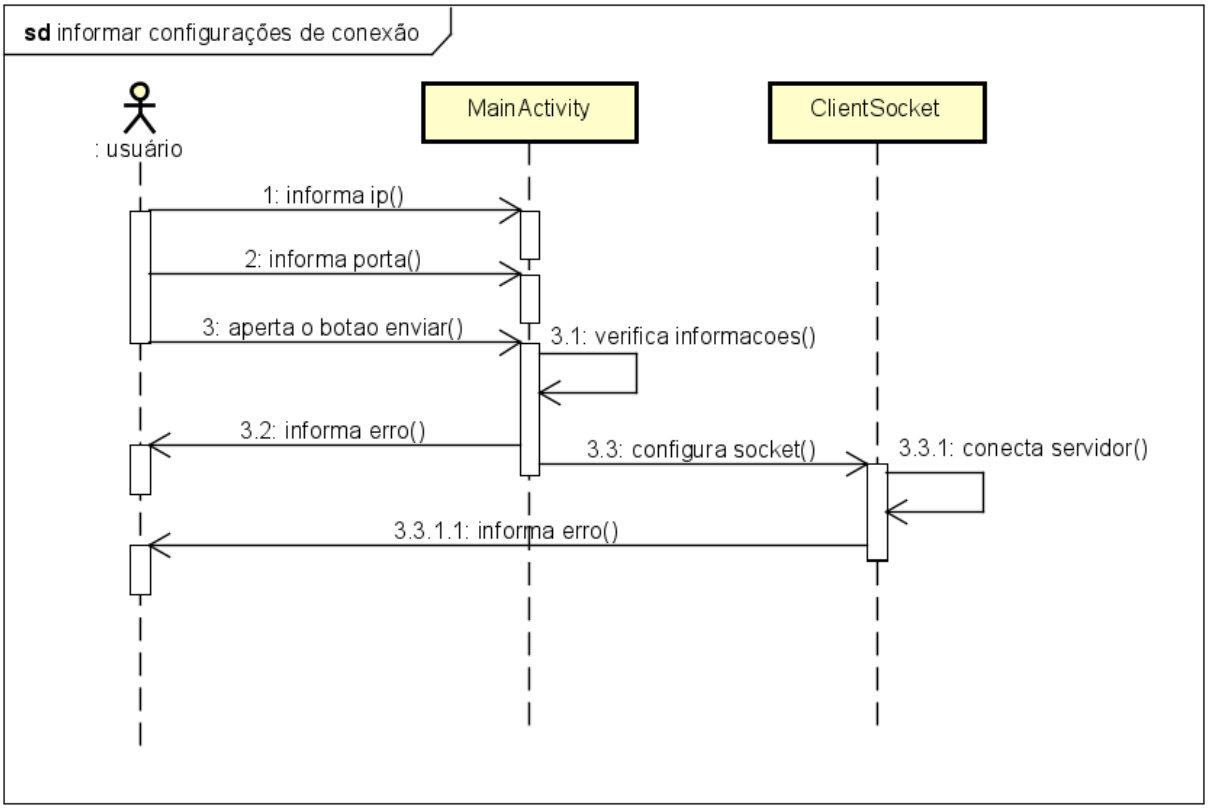
Figura 18 – Diagrama de casos de uso do projeto



Fonte: Autoria Própria (2022)

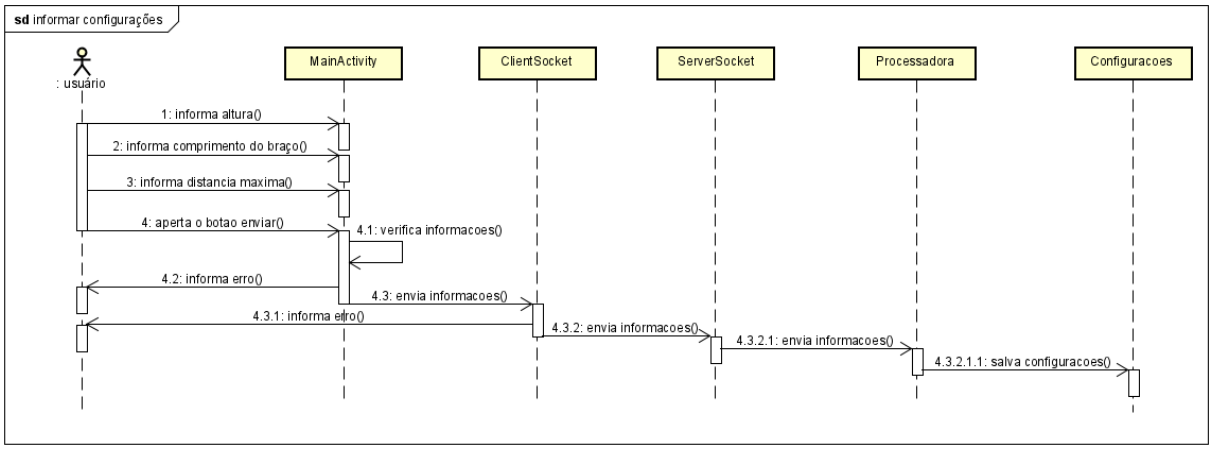
Em seguida, foram desenvolvidos os diagramas de sequência em cima do diagrama de casos de uso. No qual os dois primeiros diagramas demonstram as configurações que podem ser executadas pelo usuário, sendo o IP e porta da Raspberry Pi para efetuar a conexão e as demais para a calibração da detecção de objetos.

Figura 19 – Diagrama de sequência (Informações das configurações de conexão)



Fonte: Autoria Própria (2022)

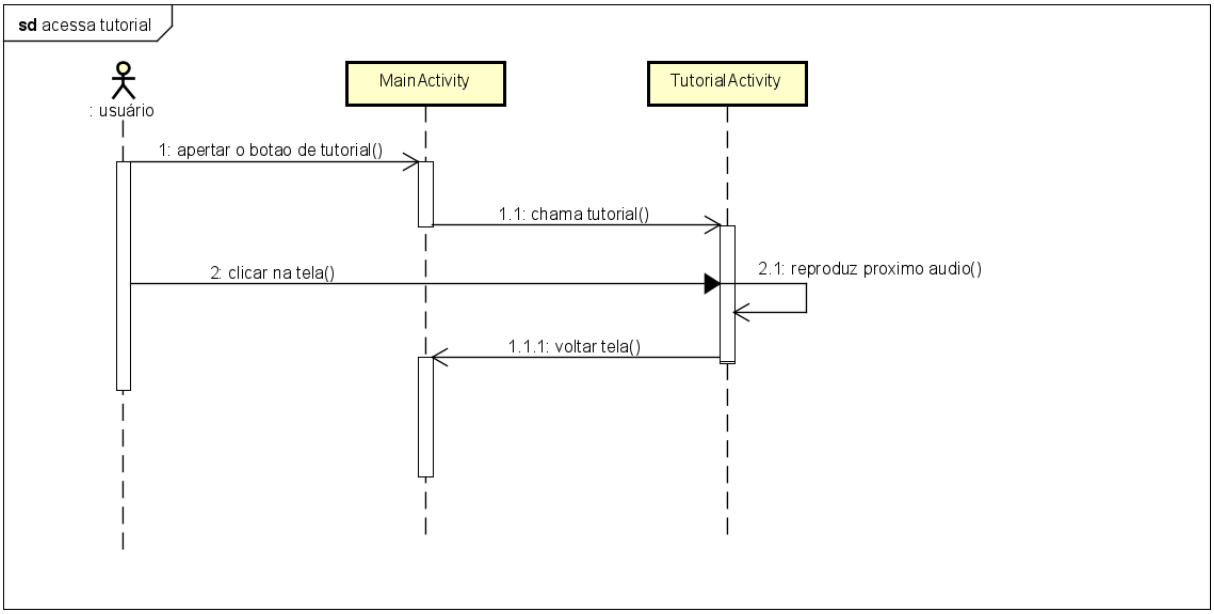
Figura 20 – Diagrama de sequência (Informações das configurações para calibração)



Fonte: Autoria Própria (2022)

Ainda sobre a parte do aplicativo, o mesmo possui um tutorial para auxiliar o usuário nas questões de configuração e como utilizar o *Talker V-EYE*, o diagrama de sequência do tutorial está a seguir:

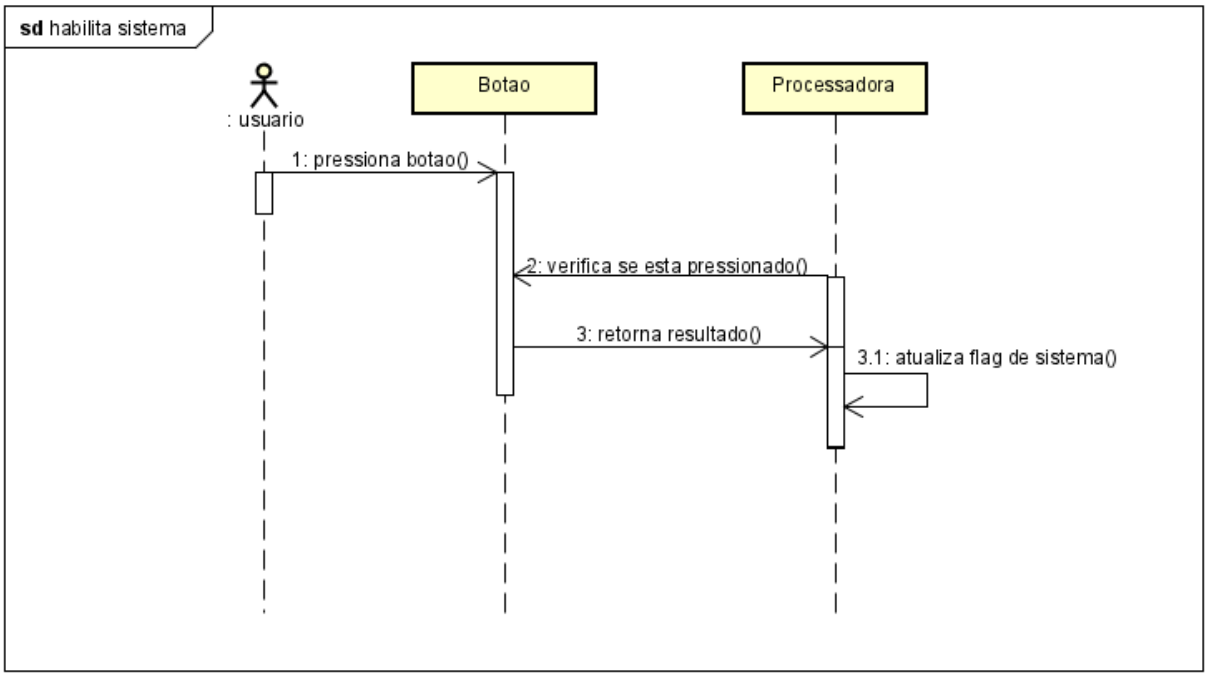
Figura 21 – Diagrama de sequência (Acessar tutorial)



Fonte: Autoria Própria (2022)

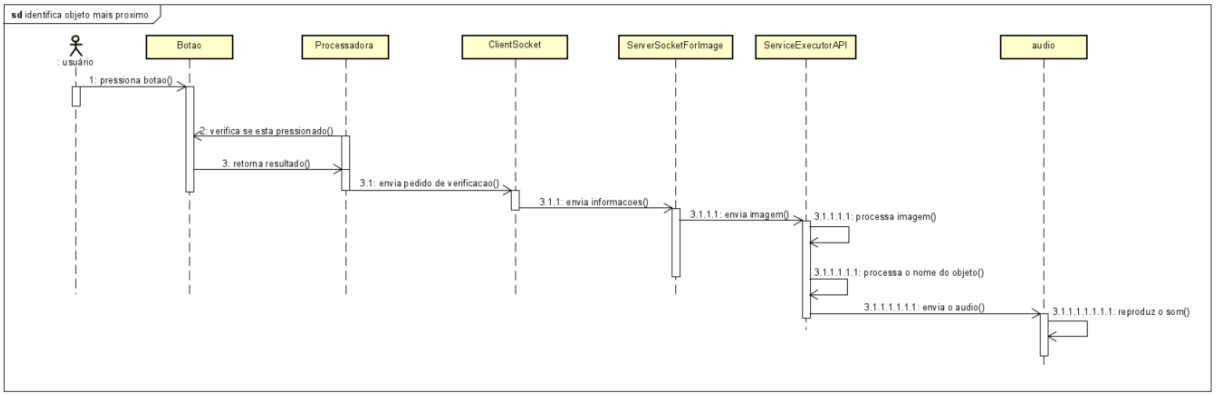
Através da pulseira do sistema, o usuário pode realizar algumas ações por meio de botões, como habilitar/desabilitar o sistema ou identificar um objeto (todos os objetos ou o objeto mais próximo); sendo esses casos mapeados nos diagramas de sequência a seguir respecitivamente:

Figura 22 – Diagrama de sequência (Habilitar Sistema)



Fonte: Autoria Própria (2022)

Figura 23 – Diagrama de sequência (Identificar objeto mais próximo)

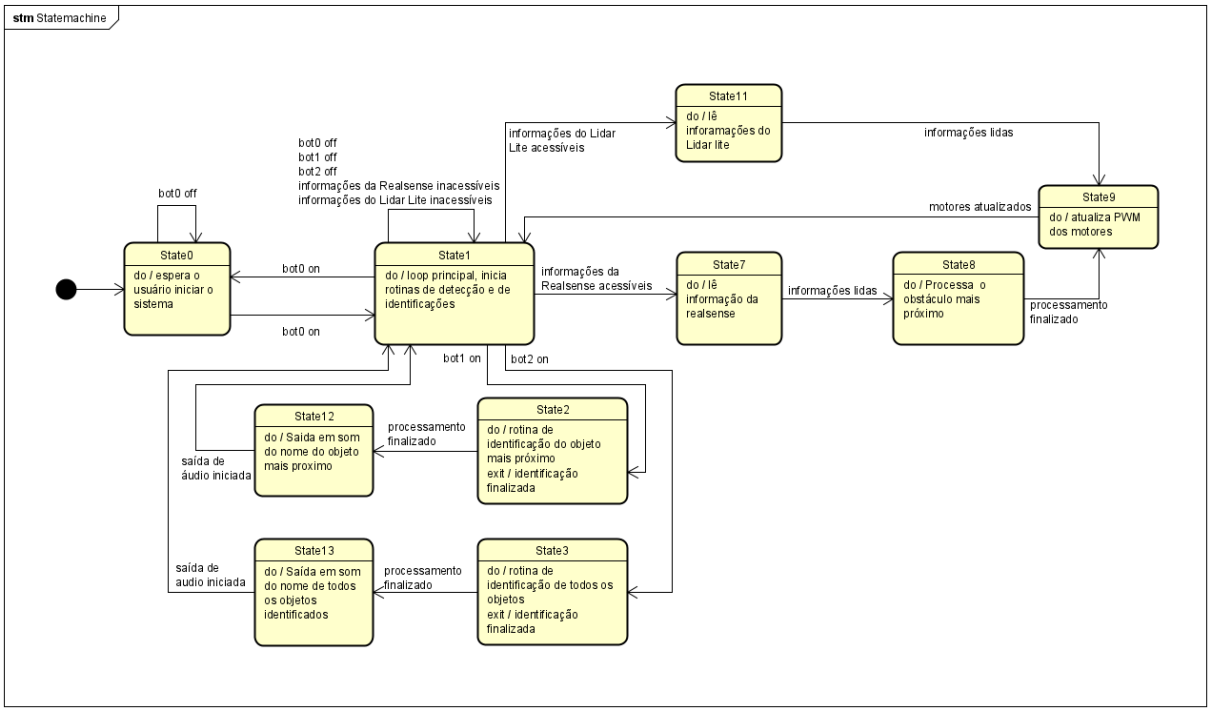


Fonte: Autoria Própria (2022)

Portanto para facilitar a inserção da configuração pelo usuário foi definido que seria implemenado através de uma aplicativo. Na questão da identificação e detecção de objetos, como também a detecção da interação com os botões presentes no sistema, foram feitas através de softwares contidos no micro-computador, que será mais detalhado posteriormente.

Abaixo está o diagrama de estados, o qual exemplifica o fluxo dos softwares do sistema.

Figura 24 – Diagrama de estados



Fonte: Autoria Própria (2022)

Após toda diagramação completa, foi iniciada a implementação dos softwares, como também as escolhas das tecnologias que seriam utilizadas. Primeiramente, no tocante ao aplicativo, o mesmo foi desenvolvido em Android nativo, devido a experiência prévia que um dos membros do grupo possui.

Para a detecção dos obstáculos, como foi utilizado a câmera RGBD, foi utilizada a biblioteca da realsense fornecida pela Intel para interfacear com o dispositivo, o qual a mesma deve ser executada em um software em C++. Portanto, outras partes que se relacionam com a detecção, como o interfaceamento com o sensor óptico a manipulação dos motores de vibração e utilização dos pinos GPIOs da Raspberry foram implementados dentro do mesmo software, sendo que a utilização do GPIO foi realizada por meio da biblioteca *pigpio*.

No tocante à identificação dos objetos, pelo conhecimento prévio de um dos integrantes sobre a API Cloud Vision do Google, a mesma pode ser utilizada de inúmeras formas de acordo com o Google. Contudo, é incentivada a utilização da biblioteca de cliente que facilita a utilização da API, e devido ao conhecimento do mesmo integrante em Java e Gradle, foi utilizado a biblioteca de cliente.

Entretanto, como a mesma API retorna apenas textos e o requisito da identificação de objetos é executar um áudio, foi utilizado a *API Text to Speech* da mesma produtora, visando unificar as configurações das API da qual está última retorna um arquivo .mp3 de acordo com o texto fornecido para a API.

Por fim, para efetuar a comunicação entre os softwares, foi utilizado *sockets* usando o protocolo TCP/IP, tendo utilizado duas instâncias, das quais uma é usada para o envio de configurações do aplicativo para o software de detecção de objetos. A outra, é utilizada quando o botão de identificação for pressionado , informando o software que contém as APIs para processar a imagem e fazer os devidos tratamentos.

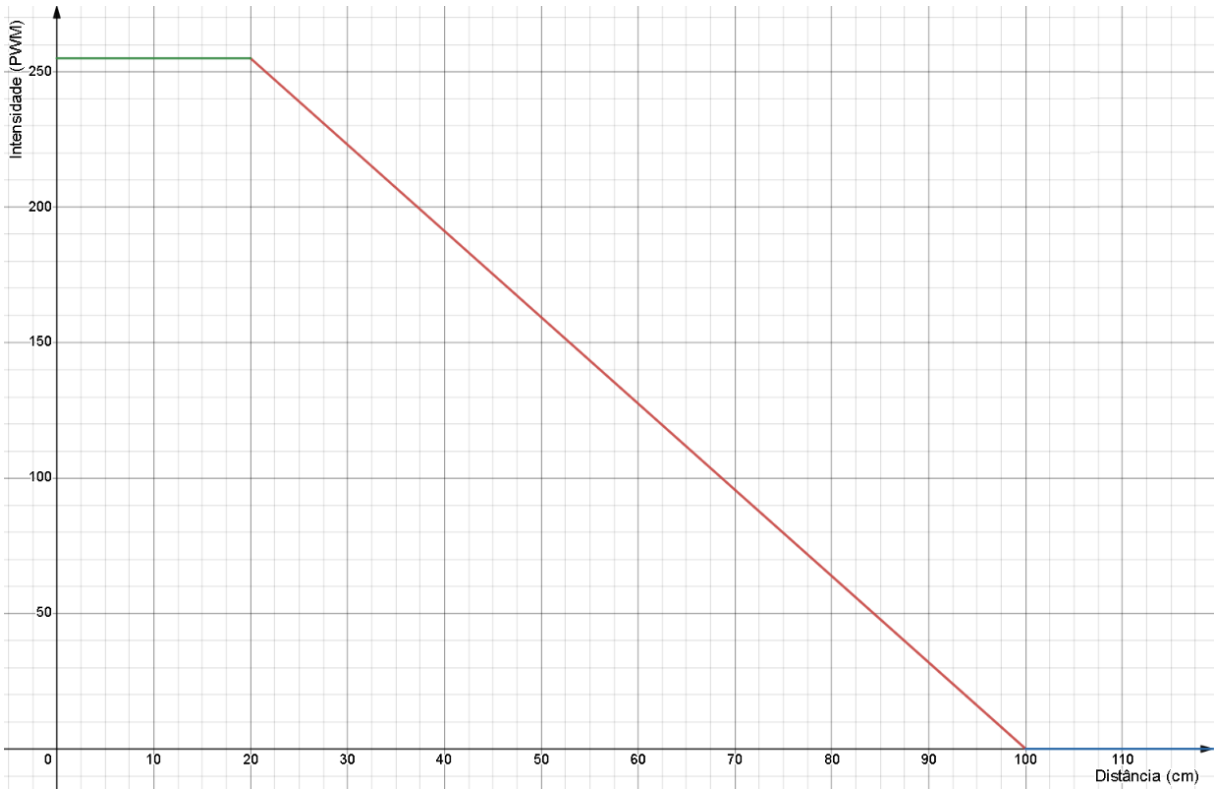
3.4.1 ALGORITMOS DE DETECÇÃO

3.4.1.1 ALGORITMO DE DETECÇÃO COM BASE NO SENSOR ÓPTICO

No caso da pulseira, a intensidade de vibração deve variar de acordo com distância retornada pelo LIDAR, que depende da região em que o usuário está apontando o sensor. Logo, o algoritmo de detecção de obstáculo do LIDAR se resume a traduzir as distâncias retornadas pelo sensor em intensidade de vibração no motor da pulseira.

A Figura 25 demonstra a relação entre intensidade de vibração (PWM) em função da distância, considerando somente o intervalo de distâncias de 20 cm à 1m. Assim, quando a distância retornada pelo sensor for de 20 cm ou menos, então a intensidade de vibração deve ser máxima (255), enquanto que, quando a distância retornada pelo sensor for de 1m ou mais, então a intensidade de vibração deve ser mínima (0).

Figura 25 – Intensidade em função da distância para a vibração do motor da pulseira.



Fonte: Autoria Própria (2022)

Logo, com base na função ilustrada na Figura 25, é possível estabelecer o algoritmo de detecção de obstáculos do sensor óptico, que está ilustrado abaixo. No algoritmo, a função converteDistanciaIntensidade() representa a função da Figura 25.

Algoritmo 1: Detecção de Obstáculos LIDAR

Input: Distancia d do LIDAR

1 $intensidadeVibracao \leftarrow$ converteDistanciaIntensidade(d)

2 vibra o motor relativo a pulseira do LIDAR com $intensidadeVibracao$

Algoritmo 2: Detecção de Obstáculos REALSENSE

```
Input: Nuvem de pontos P
distanciaMinima ← ∞
for pi ∈ P do
    distanciaAtual ← calculaDistancia(pi)
    if distanciaAtual < distanciaMinima and
      comprimentoBraco < distanciaAtual then
        if ponto não pertencer ao chão then
            distanciaMinima ← distanciaAtual
        end
    end
end
for pi ∈ P do
    distanciaAtual ← calculaDistancia(pi)
    if |distanciaAtual − distanciaMinima| < IntervaloObstaculo then
        if ponto não pertencer ao chão then
            quadrantePonto ← convertePontoQuadrante(pi.x, pi.y)
            incrementa em +1 o número de pontos de obstáculo no quadrante
              indicado por quadrantePonto
        end
    end
end
intensidadeVibracao ← converteDistanciaIntensidade(distanciaMinima)
if quadrante 1 tem um número mínimo de pontos de obstaculo then
    vibra o motor relativo ao quadrante 1 com intensidadeVibracao
end
if quadrante 2 tem um número mínimo de pontos de obstaculo then
    vibra o motor relativo ao quadrante 2 com intensidadeVibracao
end
if quadrante 3 tem um número mínimo de pontos de obstaculo then
    vibra o motor relativo ao quadrante 3 com intensidadeVibracao
end
if quadrante 4 tem um número mínimo de pontos de obstaculo then
    vibra o motor relativo ao quadrante 4 com intensidadeVibracao
end
```

3.4.1.2 ALGORITMO DE DETECÇÃO COM BASE NA CÂMERA RGBD

No caso da câmera RGBD, o algoritmo pode ser separado em três principais partes:

- A primeira varredura sobre a núvem de pontos para a identificação do ponto de menor distância (linhas 2-9);
- A segunda varredura para encontrar pontos de obstáculo próximos ao ponto de menor distância (linhas 10 - 18);
- A definição a intensidade de vibração nos motores dos quadrantes em que o obstáculo mais próximo pertence (19 - 31).

A varredura da núvem de pontos da primeira parte tem por objetivo somente coletar a distância mínima, ou seja, a distância do ponto mais próximo da câmera, sabendo que a câmera é a origem do sistema de coordenadas da núvem de pontos. Para isso, a cada ponto verificado, é utilizada a função calculaDistancia(), que através do Teorema de Pitágoras calcula a distância do ponto. Vale ressaltar que pontos que pertencem ao chão não são considerados (linha 5), além disso, todo ponto deve ter uma distância maior que a distância mínima limite, normalmente considerada como o comprimento do braço do usuário (linha 4).

A varredura da núvem de pontos da segunda parte seleciona pontos com distâncias parecidas com a distância do ponto mais próximo, os considerando assim como parte de um mesmo obstáculo (o obstáculo mais próximo ao usuário). Além disso, ao calcular um ponto do obstáculo mais próximo, ainda é indetificado o quadrante ao qual o ponto pertence (linha 14), incrementando assim mais um ponto no contador de pontos de obstáculo do respectivo quadrante.

Vale ressaltar, ainda na varredura de pontos da segunda parte, que os pontos coletados não são necessariamente pontos de um mesmo obstáculo, mas sim pontos que contém como característica em comum as distâncias mais próximas ao usuário.

Por último, na terceira parte do algortimo é calculada a intensidade na qual os motores de vibração da mochila irão agir, por intermédio da função converteDistanciaIntensidade() que possui a mesma lógica da função apresentada na Figura 25, porém com distâncias mínimas e máximas que dependem do valor de entrada do usuário por meio do aplicativo. Ao calcular a intensidade de vibração, é verificado quais quadrantes tem pontos suficientes de obstáculo para assim poder fazer o respectivo motor de vibração agir (linhas 20 - 31), essa filtragem é interessante para evitar pontos falsos

3.5 INTEGRAÇÃO

A equipe sempre teve como foco testar os módulos individualmente (principalmente hardware e software), para que durante o momento da integração, ocorre-se menores quantidades de imprevistos e erros.

Sendo assim, a integração foi divida em partes e na maioria das vezes foram incluindos testes básicos antes de testar a integração do software para eliminar alguns erros, como por exemplo no uso do sockets, constituindo testes de mandar qualquer informação, para verificar se era recebido por outra aplicação.

Portanto, a primeira parte realizada foi integrar o aplicativo com o microcontrolador, enviando novos dados de calibração e atualizar o algoritmo de detecção. Prosseguindo, a próxima parte foi integrar o funcionamento dos motores de vibração de acordo com a detecção feita pela câmera RGBD ou pelo sensor óptico.

A próxima parte a qual a equipe focou, foi integrar o software de identificação com a câmera RGBD, tomando uma foto advinda da câmera RGBD para ser utilizada na API Cloud Vision. Por fim, fazendo com que a execução da API seja feita após um botão contido na pulseira ser pressionado.

4 EXPERIMENTOS E RESULTADOS

Os experimentos para este projeto estiveram presentes em todo momento, sendo que a equipe sempre visou testar separadamente os módulos implementados (mecânica/*hardware*/*software*) antes que ocorre-se a implementação de fato.

4.1 TESTES DAS APIs

A implementação e testes para APIs não apresentaram dificuldades para os integrantes, seguindo a documentação, foi testado com a Figura 26, obtendo as informações desejadas que pode ser vistas na Figura 27. A API poderia ser utilizada tanto como API Restful, pela biblioteca do cliente ou por chamadas via Spring Boot. E estudando as três formas, foi escolhida a biblioteca do cliente pela facilidade de implementação.

Figura 26 – Imagem utilizada para o teste da API Cloud Vison

Fonte: Google (2022)

Figura 27 – Imagem do retorno da API Cloud Vison

```
Object name: Cat
Confidence: 0.84174824
Normalized Vertices:
- (0.2858942, 0.17550413)
- (0.7382641, 0.17550413)
- (0.7382641, 0.980575)
- (0.2858942, 0.980575)
Object name: Furniture
Confidence: 0.58846515
Normalized Vertices:
- (0.8298903, 0.08395647)
- (0.9973958, 0.08395647)
- (0.9973958, 0.98583275)
- (0.8298903, 0.98583275)
Object name: Window
Confidence: 0.5883046
Normalized Vertices:
- (0.0019963116, 0.015424866)
- (0.21847336, 0.015424866)
- (0.21847336, 0.7845455)
- (0.0019963116, 0.7845455)
```

Fonte: Autoria Própria (2022)

4.2 TESTES DA ESTRUTURA

Tanto a mochila, com as devidas alterações citadas anteriormente, quanto a pulseira foram testadas em questão de conforto e funcionalidade, para confirmar se todos os componentes permaneceram acoplados corretamente. Além disso, ela foi testada também juntos aos outros módulos do sistema no teste final do protótipo, onde não se verificou nenhum problema mecânico.

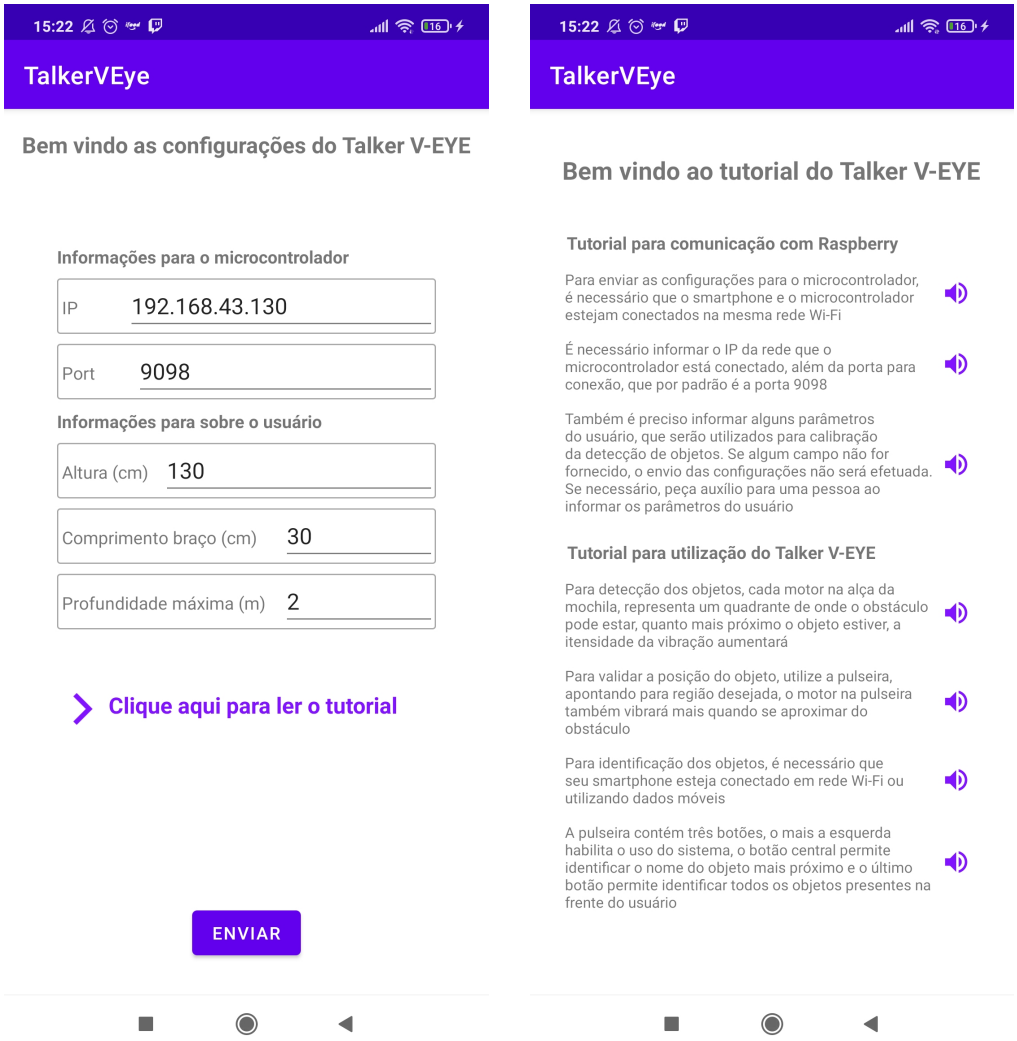
4.3 TESTES DOS MOTORES DE VIBRAÇÃO

Em relação aos atuadores do sistema (motores de vibração), a fabricação das placas dos motores foi feita por somente um dos integrantes. Para o desenvolvimento das placas, foram utilizadas placas de circuito perfurada para soldar os compontentes com base nos esquemáticos da Figura 16 e Figura 17. Para testar a variação da intensidade de vibração dos motores e validar o funcionamento do circuito, foi utilizado um Arduino Uno com um programa simples que varia a intensidade de vibração do motor por meio de um pino de PWM.

4.4 TESTES DO APLICATIVO

Na parte da configuração do sistema pelo aplicativo, não houveram problemas na criação da interface do usuário,devido a simplicidade da interface. Contudo, o que diferiu da projeção inicial era o funcionamento do tutorial, o qual a execução do áudio, que a princípio era realizada pelo botão contido na tela à esquerda de cada texto, também ocorre agora de forma iterativa ao clicar em qualquer ponto da tela, visando melhorar o uso para o usuário alvo (deficiente visual). A figura Figura 28 contém o resultado da tela do tutorial do aplicativo e a tela de configurações.

Figura 28 – Tela de tutorial e configurações do aplicativo



Fonte: Autoria Própria (2022)

4.5 TESTES DE SOCKETS

Para a questão do transporte de dados entre o aplicativo e o microcontrolador, iniciou-se com testes locais, sendo uma execução em Java e outra em C++ (contida em uma máquina virtual utilizando Ubuntu). Após ter criado aplicações de testes com *sockets* e ter efetuado a conexão e comunicação entre eles enviando textos simples e com a aplicação já criada, foi inserida a parte de *sockets* e novamente testado, para ver se ainda ocorria a comunicação, a qual foi bem sucedida.

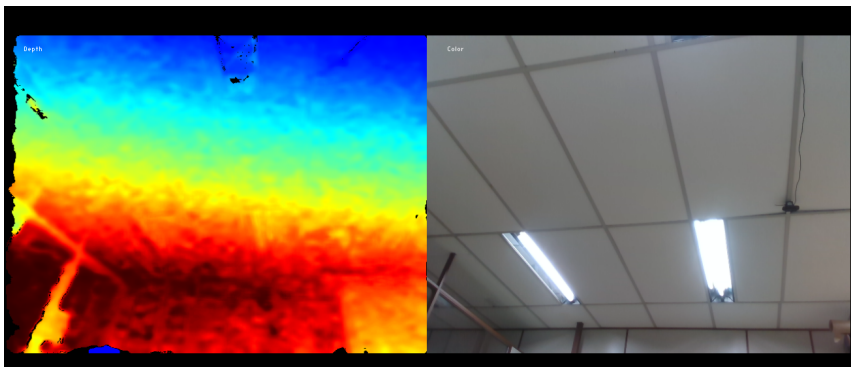
4.6 TESTES DE REPRODUÇÃO DO ÁUDIO

Uma das últimas partes implementadas foi a execução automática do áudio retornado pela API Text To Speech, para isso foi utilizada uma biblioteca em Java que tem como função a execução de vários tipos de arquivos de áudio. Em testes em uma máquina local, a bliblioteca apresentou ser promissora, executando sem nenhum problema os áudios gerados. Contudo foram detectados problemas no momento da integração com o microcontrolador. Portanto, foi necessário que o software em Java executasse comandos pelo terminal, rodando o áudio recebido pela API Text To Specch.

4.7 TESTES DA REALSENSE

Os testes iniciais da Realsense foram focados principalmente no uso da biblioteca, lendo imagens RGB , medindo de distâncias, como visto no resultado apresentado na Figura 29, além de leituras do *point-clound*, em que foram obtidos resultados positivos.

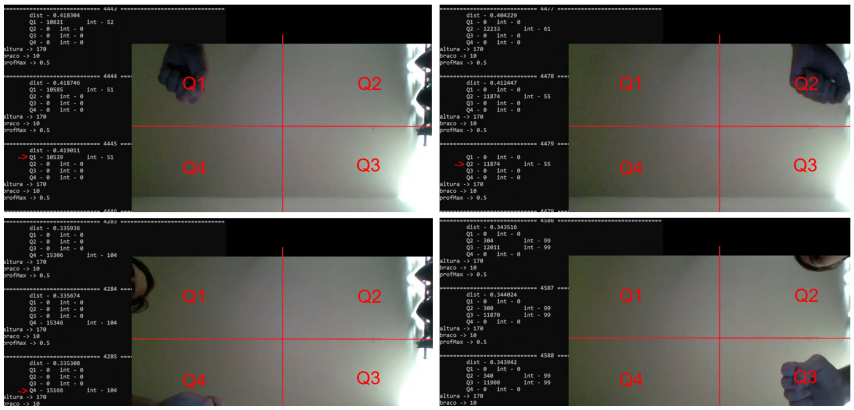
Figura 29 – Imagem RGB e medição de distância da Realsense



Fonte: Autoria Própria (2022)

Depois disso foi testado o algoritmo de detecção de obstáculos, colocando a câmera em um ambiente neutro e então posicionado um obstâculo em cada quadrante, um a um, indicado Figura 30. Esse teste obteve resultados positivos possibilitando a continuidade do desenvolvimento do projeto.

Figura 30 – Detecção de obstáculos por quadrante



Fonte: Autoria Própria (2022)

4.8 TESTES DO LIDAR-Lite v3

Inicialmente foi testado apenas a comunicação pelo i2c com o sensor assim como a medição de distâncias, o que deu certo sem grandes problemas. Depois disso fizemos alguns testes do valor retornado para definir o erro médio do sensor, resultou que para distâncias curtas, menos que 10cm, possui um erro significativo, porém esse erro não se mostrou problemático por conta do valor minimo de distância que foi definido inicialmente.

4.9 TESTES DE INTEGRAÇÃO

Na questão da integração, foram necessários inúmeros testes que não foram possíveis de executar nos testes individuais, como a integração da câmera RGBD com o software de identificação; e a integração dos dados fornecidos pela câmera RGBD e pelo sensor óptico com os motores de vibração.

Sendo assim, no quesito da integração entre a câmera RGBD e o software de identificação de objetos, inicialmente o projeto consistia em enviar a imagem por *socket*, entretando, nos testes realizados pela equipe, foi verificado que a imagem não era enviada completamente. Desse modo, foi escolhido gravar a imagem como arquivo no diretório do software em Java. Caso a imagem seja gravada com sucesso, é enviada uma mensagem por *sockets* com algumas informações, sendo o tipo de identificação, tamanho da imagem e posição de um dos pixels do objeto mais próximo. Desse modo, após a alteração do envio das mensagens no *socket*, foram aplicados novos testes, os quais tivemos sucesso nas respostas. Contudo, é importante ressaltar que as respostas da identificação estão alinhadas com o que era esperado pela equipe e pontuado no plano de projeto, pois em alguns casos não foram identificados todos os objetos presentes ou não teve objeto identificado, sendo esses resultados principalmente em áreas abertas. Além do mais, na identificação do objeto mais próximo há casos de não ter sido possível fazer a relação com o objeto mais próximo detectado pela câmera RGBD.

A integração entre o aplicativo e o software de detecção de objetos apresentou nenhum problema, conforme foi analisado em testes prévios em uma máquina local e posteriormente entre o aplicativo e osoftware presente no microcontrolador.

5 CRONOGRAMA E CUSTOS DO PROJETO

5.1 CRONOGRAMA

Na figuras abaixo, estão apresentados por completo o cronograma do projeto, sendo as imagens divididas por entregáveis, onde o cronograma foi seguido e atualizado no decorrer do desenvolvimento do *Talker V-EYE*. Pelas imagens, é possível verificar informações como as tarefas, data estipulada para término da tarefa, os integrantes participantes da tarefa e com qual requisito funcional e não funcional a tarefa está relacionada.

Figura 31 – Cronograma - Entregável 1 (Planejamento)

NÚMERO DA EAP	REQUISITOS	TÍTULO DA TAREFA	PROPRIETÁRIO DA TAREFA	DATA DE INÍCIO	DATA DE CONCLUSÃO	DURAÇÃO (HORAS)	DURAÇÃO (DIAS)	% DA TAREFA CONCLUÍDA
1		**Definição e plano do projeto**						
1.1	-	Definição do projeto	Todos	09/03/22	23/03/22	12	14	100 %
1.2	-	Definição dos requisitos	Todos	16/03/22	30/03/22	8	14	100 %
1.3	-	Elaboração do plano de projeto	João e Camilo	23/03/22	31/03/22	10	8	100 %
1.4	-	Testes iniciais com a realsense	Luciano	23/03/22	28/03/22	3	5	100 %
1.5	-	Elaboração do UML	Todos	30/03/22	01/04/22	2	1	100 %
2		**Elaboração e início do projeto**						
PP: Plano de Projeto								

Fonte: Autoria Própria (2022)

Figura 32 – Cronograma - Entregável 2 (Plano de Projeto)

2.1	RF003	Estudo do controle da Realsense	Luciano	30/03/22	13/04/22	8	13	100 %
2.1.1	RF003	Teste do protótipo do software que utiliza a Realsense	Luciano	10/04/22	13/04/22	6	3	100 %
2.2	RF004	Estudo do controle do LIDAR-Lite	Luciano	06/04/22	19/04/22	2	13	100 %
2.2.1	RF004	Teste do protótipo do software que utiliza o LIDAR-Lite	Luciano	06/04/22	19/04/22	2	13	100 %
2.3	-	Produção do blog	João	13/04/22	19/04/22	3	6	100 %
2.4	RF003/RF004	Teste do motor Vibracall	João	30/03/22	06/04/22	2	6	100 %
2.4.1	RF003/RF004	Produção de placas e PCBs	Todos	06/04/22	19/04/22	6	13	100 %
2.4.2	RF003/RF004	Estudo da biblioteca wiringpi e elaboração do software dos motores	João	20/04/22	28/04/22	8	8	100 %
2.4.3	RF003/RF008	Teste do programa feito para os motores	João	28/04/22	30/04/22	2	2	100 %
2.5	RF006	Estudo do uso da API Vision	Camilo	30/03/22	13/04/22	4	13	100 %
2.5	RF006	Testes do software da API Vision	Camilo	30/03/22	13/04/22	4	13	100 %
2.6	RF006	Estudo do uso da API Talk to Speech	Camilo	06/04/22	19/04/22	4	13	100 %
2.6	RF006	Testes do software da API Talk to Speech	Camilo	06/04/22	19/04/22	4	13	100 %
E1: Site/blog de acompanhamento								

Fonte: Autoria Própria (2022)

Figura 33 – Cronograma - Entregável 3 (Mecânica)

2.7	RF001	Adaptação da estrutura da mochila	Todos	13/04/22	19/04/22	1	6	100 %
2.8	-	Elaboração e/ou modificação da estrutura para a pulseira	Todos	13/04/22	19/04/22	1	6	100 %
2.9	RF001	Mochila (Integração Componentes)	Todos	18/04/22	19/04/22	2	1	100 %
2.9.1	RF001	Testes de uso (conforto e usabilidade da mochila e pulseira)	Todos	18/04/22	19/04/22	1	1	100%
E2: Projeto/design e montagem da estrutura mecânica								

Fonte: Autoria Própria (2022)

Figura 34 – Cronograma - Entregável 4 (Hardware)

E2: Projeto/design e montagem da estrutura mecânica								
2.10	-	Placa de conecçõese da placa dos botões	Luciano	20/04/22	27/04/22	7	7	100 %
2.13	RF002/RNF007	Elaboração das conexões (Sockets TCP/IP)	Camilo	19/04/22	26/04/22	3	7	100 %
2.13.1	RF006/RNF007	Elaborar o envio de foto do C++ para o Java por Sockets	Camilo	19/04/22	26/04/22	4	7	100 %
2.13.2	RF002/RNF007	Elaborar o envio das configurações do aplicativo para o C++ por Sockets	Camilo	19/04/22	26/04/22	4	7	100 %
E3: Projeto eletrônico e PCB. Testes de módulos isolados de hardware								

Fonte: Autoria Própria (2022)

Figura 35 – Cronograma - Entregável 5 (Software)

2.16	RF003/RF005	Integração Realsense/vibração	Luciano e João	01/05/22	20/05/22	20	19	100 %
2.17	RF004/RF005	Integração Lidar/Vibração	Luciano e João	01/05/22	20/05/22	20	19	100 %
2.17.1	RF003/RF008	Testes do software criado para Detecção de objetos pela realsense e a vibração dos motores	Luciano e João	18/05/22	20/05/22	4	2	100 %
2.17.2	RF004	Testes do software criado para Detecção de objetos pelo Lidar e a vibração dos motores	Luciano e João	18/05/22	20/05/22	4	2	100 %
2.17.3	RF005	Teste da integração da detecção de objetos e a vibração	Luciano	18/05/22	20/05/22	4	2	100 %
2.18	RF002	Desenvolvimento do aplicativo do usuário	Camilo	27/04/22	04/05/22	8	7	100 %
2.18.1	RF002	Testes da calibração da câmera feita pelo aplicativo	Camilo	27/04/22	04/05/22	2	7	100 %
2.18.2	RF002	Testes da comunicação entre aplicativo e software na raspberry pi para calibração da câmera	Camilo	27/04/22	04/05/22	2	7	100 %
2.19	-	Elaboração do UML (Finalização)	Todos	01/05/22	04/05/22	8	3	100%
E4: Projeto do software com UML. Testes de módulos individuais de software								

Fonte: Autoria Própria (2022)

Figura 36 – Cronograma - Entregável 6 (Integração)

2.20	RF009/RF007	Programação para reprodução de áudio (alto-falante)	João	20/05/22	24/05/22	4	4	100 %
2.20.1	RF006/009	Testes do retorno da API, vinculando com a utilização do auto-falante conectado no microcontrolador	Camilo	18/05/22	24/05/22	3	6	100 %
2.20.2	RF007	Testes no projeto para ver se segue detectando os objetos sem estar conectado a internet	Todos	22/05/22	24/05/22	2	2	100%
2.21	RF006	Programação para o uso dos botões	Luciano	20/05/22	24/05/22	4	4	100 %
2.21.1	RF006	Testes da programação para utilização dos botões e funcionalidade dos botões	Luciano	20/05/22	24/05/22	4	4	100 %

Fonte: Autoria Própria (2022)

Figura 37 – Cronograma - Entregável 7 (Relatório Técnico Final)

3		**Finalização**						
3.1	-	Relatório Técnico	Joao e camilo	25/05/22	08/06/22	13	13	100 %
3.2	-	Revisão e Correção	Todos	25/05/22	08/06/22	13	13	100 %
DP: Demonstração do funcionamento do protótipo								
3.3	-	Relatório Técnico	Joao e camilo	09/06/22	14/06/22	5	5	100 %
3.4	-	Produção do vídeo	Todos	09/06/22	14/06/22	5	5	100 %
Prévia vídeo e Prévia do relatório final								
3.5	-	Relatório Técnico	Todos	15/05/22	21/06/22	36	36	100 %
3.6	-	Produção do vídeo	Todos	15/06/22	21/06/22	6	6	100 %
Vídeo e Relatório final								

Fonte: Autoria Própria (2022)

5.2 CUSTOS DO PROJETO

Na Tabela 1, estão apresentados o custos acatados no projeto, sendo os custos divididos igualmente pelos membros da equipe. Vale ressaltar, que a maioria dos equipamentos conseguimos emprestado sem nenhum custo, logo, o custo real do projeto foi muito inferior ao que está contido na tabela.

Tabela 1 – Custos do projeto.

Item	Quantidade	Preço Total (R$)
Intel Realsense d435i	1	2700,00
Lidar Lite V3	1	1253,00
Raspberry pi 4B+	1	1500,00
Bateria Lipo	1	220,00
Motor de Vibração	5	41,80
Mochila com modificações	1	70,00
Power Bank	1	60,00
Power Module	1	96,00
Optoaclopador	5	7,20
Diodo N4007	5	1,15
Chave táctil	3	0,36
Barra de pinos macho	1	1,50
Capacitor 680uF	1	1,50
Resistor 1K	3	0,45
Resistor 510	5	0,75
Resistor 910	5	0,75
Papel Foam Board	1	29,90
Total		5.984,36

Fonte: Autoria prória

6 CONCLUSÃO

Portanto, para este projeto, a equipe pode colocar em prática conceitos vistos em inúmeras matérias cursadas durante o curso, como também foi estudado novos conceitos para seguir com a conclusão do *Talker V-EYE*. Além do mais, foi possível de forma prática lidar com conceitos mais genéricos, como adminstração das tarefas e projeção do cronograma, inserindo a dificuldade e tempo gasto para cada tarefa e observando as distinções entre o que foi planificado e os reais resultados.

Outro fator importante vivenciado durante a matéria de Oficinas de Integração 2, foi a percepção das variadas soluções para um mesmo problema, tomando a projeção e o custo benefício de cada solução em consideração para a continuação do desenvolvimento.

Sendo assim, de acordo com os resultados do *Talker V-EYE* apresentados anteriormente, é possível concluir que o projeto teve sucesso no desenvolvimento e finalização.Visto que, os requisitos funcionais e não funcionais propostos no plano de projeto foram cumpridos, além do satisfatório funcionamento do *Talker V-EYE* como um protótipo, tomando o tempo de desenvolvimento e a expectativa sobre as funcionalidades do projeto.

6.1 TRABALHOS FUTUROS

No decorrer da elaboração do projeto, foram observadas inúmeras partes que podem ser melhoradas, como por exemplo aumentar a precisão da detecção de obstáculos, a experiência do usuário com a utilização do *Talker V-EYE* e o custo benefício para a confeccção do projeto.

Sobre a detecção, foi discutido pela equipe a mudança do algoritmo de detecção, primeiramente estudando algoritmos mais elaborados e robustos, os quais desconsideram o chão e teto como obstáculos. Conseguinte, é desejo da equipe aprimorar as respostas para o usuário, modificando a estrutura para trazer mais clareza espacial por meio das vibrações dos motores e traduzir o áudio da identificação dos objetos para português.

Além disso, foi pontuado pela equipe modificações na comunicação entre a pulseira e o microcontrolador, para que seja feita por meio de comunicação sem fio (wi-fi ou blueto-oth).Por fim, a equipe planeja modificar a utilização da alimetação do projeto, produzindo uma alimentação com maior capacidade para aumentar a autonomia do *Talker V-EYE*.

Referências

CANELLE, G. K.; ARAúJO, E. C. de. **Do C/C++ para o Java: Conheça as diferenças e principais características**. 2012. Disponível em: <https://www.devmedia.com.br>. Acesso em: 13 de junho de 2022. Citado na página 7.

FILIPEFLOP. **Motor de Vibração 1027**. 2018. Disponível em: <https://www.filipeflop.com/produto/motor-de-vibracao-1027/>. Acesso em: 11 de junho de 2022. Citado na página 5.

GARMIN. **Lidar Lite v3 Operation Manual and Technical Specifications**. 2016. Disponível em: <https://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf>. Acesso em: 11 de junho de 2022. Citado na página 4.

GUIMARAES, F. **I2C**. 2018. Disponível em: <https://mundoprojetado.com.br/i2c/>. Acesso em: 11 de junho de 2022. Citado 3 vezes nas páginas 5, 6 e 7.

INTEL. **Depth Camera D435**. 2018. Disponível em: <https://www.intelrealsense.com/depth-camera-d435/>. Acesso em: 11 de junho de 2022. Citado na página 4.

KHAN, A. **Raspberry Pi 4 GPIO Pinout**. 2022. Disponível em: <https://linuxhint.com/gpio-pinout-raspberry-pi/>. Acesso em: 12 de junho de 2022. Citado na página 8.

OLIVEIRA, L. B. **Cartilha Do Censo 2010**: Pessoas com deficiência. 2012. Disponível em: <https://inclusao.enap.gov.br/wp-content/uploads/2018/05/cartilha-censo-2010-pessoas-com-deficienciareduzido-original-eleitoral.pdf>. Acesso em: 07 de junho de 2022. Citado na página 1.

PANTUZA, G. **O QUE SÃO E COMO FUNCIONAM OS SOCKETS**. 2017. Disponível em: <https://linuxhint.com/gpio-pinout-raspberry-pi/>. Acesso em: 12 de junho de 2022. Citado na página 8.

VENTURA, L. **Fila para obter cão-guia no Brasil tem mais de 500 pessoas**. 2012. Disponível em: <https://brasil.estadao.com.br/blogs/vencer-limites/fila-para-obter-cao-guia-no-brasil-tem-mais-de-500-pessoas/>. Acesso em: 07 de junho de 2022. Citado na página 1.