UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Eduardo Vinicius Hahn

Dispositivo de Apoio a Localização para Pessoas Cegas

Araranguá
2022

Eduardo Vinicius Hahn

Dispositivo de Apoio a Localização para Pessoas Cegas

Trabalho de Conclusão de Curso submetida ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia de Computação.
Orientador: Prof. Fábio Rodrigues de la Rocha, Dr.

Araranguá
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Hahn, Eduardo
Dispositivo de Apoio a Localização para Pessoas Cegas / Eduardo Hahn ; orientador, Fábio Rodrigues de la Rocha, 2022.
25 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2022.

Inclui referências.

1. Engenharia de Computação. I. Rocha, Fábio Rodrigues de la . II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Eduardo Vinicius Hahn

Dispositivo de Apoio a Localização para Pessoas Cegas

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de bacharel e aprovado em sua forma final pelo Curso Engenharia de Computação

Araranguá, 25 de julho de 2022.

Documento assinado digitalmente
Analucia Schiaffino Morales
Data: 02/08/2022 10:43:59-0300
CPF: 622.256.420-87
Verifique as assinaturas em https://v.ufsc.br

Profª. Analúcia Schiaffino Morales, Drª.
Coordenadora de Curso

Banca Examinadora:

Documento assinado digitalmente
Fabio Rodrigues de La Rocha
Data: 30/07/2022 13:28:52-0300
CPF: 963.007.520-20
Verifique as assinaturas em https://v.ufsc.br

Prof. Fábio Rodrigues de la Rocha, Dr.
Orientador

Documento assinado digitalmente
Anderson Luiz Fernandes Perez
Data: 30/07/2022 10:45:01-0300
CPF: 020.803.459-58
Verifique as assinaturas em https://v.ufsc.br

Prof. Anderson Luiz Fernandes Perez, Dr.
Avaliador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Jim Lau
Data: 31/07/2022 16:03:25-0300
CPF: 613.464.702-00
Verifique as assinaturas em https://v.ufsc.br

Prof. Jim Lau, Dr.
Avaliador
Universidade Federal de Santa Catarina

Dispositivo de Apoio a Localização para Pessoas Cegas

LOCATION SUPPORT DEVICE FOR BLIND PEOPLE

Eduardo Vinicius Hahn * Fábio Rodrigues de la Rocha †

2022, Março

Resumo

Pessoas cegas possuem dificuldade de locomoção em novos locais, principalmente pela falta de informação sobre o ambiente, além disso há dificuldade de acesso a dispositivos e/ou cães guias que facilitem e permitam maior independência destas pessoas, devido ao problema de baixo orçamento comumente enfrentado por parte da população brasileira. Por isso este projeto tem como objetivo estudar a viabilidade do desenvolvimento de uma tecnologia assistiva através de um equipamento embarcado de baixo custo com a câmera acoplada a um acessório na cabeça capaz de capturar imagens e detectar *QR Codes*. Esses *QR Codes*, devem ser pré-instalados em locais adequados para identificar salas de aula, escritórios, banheiros, etc. A partir da leitura do *QR Code*, este será enviado a um servidor e o aplicativo criado irá buscar e reproduzir a informação ao usuário. Durante o estudo foram desenvolvidos 3 protótipos para validar a efetividade e viabilidade do sistema. Após a finalização dos protótipos foram realizados testes tanto de uso do sistema quanto sobre performance do mesmo.

Palavras-chaves: Tecnologia assistiva, *QR Code*, Aplicativo para *smartphone*, Sistema embarcado.

*eduardo.hahn@grad.ufsc.br
†fabio.rocha@ufsc.br

Dispositivo de Apoio a Localização para Pessoas Cegas

LOCATION SUPPORT DEVICE FOR BLIND PEOPLE

Eduardo Vinicius Hahn * Fábio Rodrigues de la Rocha †

2022, Março

Abstract

Blind people have difficulty moving in new places, mainly due to the lack of information about the environment, in addition, there is difficulty in accessing devices and/or guide dogs that facilitate and allow greater independence of these people, due to the low budget problem commonly faced by part of the Brazilian population. Therefore, this project aims to study the feasibility of developing an assistive technology through a low-cost embedded device with a camera attached to an accessory on the head capable of capturing images and detecting QR Codes. These QR Codes must be pre-installed in suitable locations to identify classrooms, offices, bathrooms, etc. After reading the QR Code, it will be sent to a server and the created application will search for and reproduce the information for the user. During the study, 3 prototypes were developed to validate the effectiveness and feasibility of the system. After the completion of the prototypes, tests were carried out both on the use of the system and on its performance.

Keywords: Assistive technology, QR Code, Smartphone application, Embedded system.

*eduardo.hahn@grad.ufsc.br
†fabio.rocha@ufsc.br

1 Introdução

Segundo o Censo Demográfico de 2010, estudo realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE, 2010) havia no Brasil, 45,6 milhões de pessoas com algum tipo de deficiência, o que equivale a 23,91% da população brasileira sendo que deste total, 12,7 milhões (6,7% da população total) possuem pelo menos um tipo de deficiência grave, neste grupo a deficiência visual é a mais presente, acometendo 3,5% da população do Brasil, o que equivale a 6.5 milhões de habitantes.

Para uma pessoa ser considerada cega é necessário se enquadrar nas diretrizes do decreto 5.296 de 2004, na qual cegueira é determinada pela acuidade visual ser menor ou igual a 0,05 no melhor olho. Da mesma forma que a baixa visão é definida pela acuidade visual ser entre 0,3 e 0,05 no melhor olho. A acuidade visual é a capacidade do olho para diferenciar os detalhes espaciais, de outra forma é a competência de identificar a forma e o contorno dos objetos e é conhecido pelas famosas imagens com letras de diversos tamanhos. Também pode ser definido como deficiente visual a pessoa que possuir uma somatória de campo visual em ambos os olhos menor ou igual que 60º.(BRASIL. . . , s.d.)

Pessoas com cegueira ou baixa visão enfrentam uma constante falta de informação sobre ambiente, principalmente em locais que nunca estiveram. Esta informação está associada a local, objetos e até pessoas. A carência de ajuda para suprir a falta de informação cria situações em que a pessoa com dificuldade só resolve recorrendo a auxílio de outras pessoas. Alguns exemplos são identificação de presença de pessoas, informações escritas e outros detalhes de rotina.(RODRIGUES; DUARTE; GUERREIRO, 2020)

Atualmente encontra-se algumas tecnologias assistivas como o (VEEVER. . . , s.d.) que funciona como um assistente em ambientes urbanos contendo descrições de local por áudio e um micro mapeamento que auxilia a pessoa cega e com baixa visão ter uma melhor vivência. Este sistema atua com *Beacons* instalados nos ambientes, assim que o usuário chega ao local recebe uma descrição do mesmo para facilitar sua mobilidade. Além disso, se o usuário apontar o *smartphone* para direção desejada o aplicativo irá informar quais os pontos de interesse estão mapeados ali. Uma barreira deste sistema é o custo de instalação, pois é necessário um *Beacon* para cada ambiente. Outros sistemas utilizam a tecnologia *RFID* para a mesma finalidade. Módulos *RFID* são instalados no ambiente, com descrições e orientações sobre o mesmo, o usuário possui um dispositivo leitor *RFID*, deste modo ao chegar ao ambiente recebe o sinal e a explicação por áudio.

Deste modo, sabendo que pessoas cegas e com baixa visão possuem dificuldade de locomoção em novos locais, principalmente pela falta de informação sobre o ambiente, além de que o baixo orçamento é um problema comum enfrentado por parte da população brasileira, dificultando o acesso a dispositivos e/ou cães guias que facilitem e permitam maior independência a pessoas cegas, este projeto tem como objetivo de oferecer um auxílio na obtenção de informações sobre o ambiente.

Este auxílio será oferecido na forma de um equipamento embarcado de baixo custo para ser utilizado em um óculos ou chapéu capaz de capturar imagens e detectar *QR Codes*. Esses *QR Codes* devem ser pré-instalados em locais chave para identificar salas de professor, banheiros, auditórios, elevadores, etc. O uso do *QR Code* é especialmente útil, pois diferentemente de outras etiquetas como *RFID* e *Beacons*, não é um equipamento que possui custo associado e pode ser simplesmente impresso em papel comum e afixado em local adequado rapidamente. A partir da leitura do *QR Code*, será enviado a decodificação ao aplicativo que irá reproduzir a informação ao usuário. Para isso foram desenvolvidos

protótipos como prova de conceito e sobre estes foram realizados testes funcionais para validar a proposta.

2 Revisão da literatura

Nesse capítulo serão apresentados os trabalhos relacionados ao tema, descrevendo as principais tecnologias utilizadas, desde *hardware* até *software*.

(DUARTE *et al.*, 2020) apresenta um sistema de navegação e informação em interiores de edifícios com a intenção de auxiliar a locomoção de pessoas cegas. O sistema utiliza emissores *Bluetooth* (instalados previamente no local) e técnicas de triangulação usando a força do sinal *Bluetooth*. Fundamentalmente utiliza ao menos três intensidades de sinal (que estão relacionadas à distância), assim a posição do usuário pode ser estimada. Outra parte importante do sistema é o módulo de comando de voz, este módulo transforma a voz em texto para enviar ao servidor e faz o caminho contrário para enviar a mensagem ao usuário. A API do *Google Voice Recognizer* é utilizada para receber o áudio e transformá-lo em comando para o aplicativo e na saída é utilizada a API *TextToSpeech* que reproduz alguma informação útil ao usuário.

(LÓPEZ-DE-IPIÑA; LORIDO; LÓPEZ, 2011) expõe um sistema de compras autônomas para pessoas cegas utilizando *QR Code* e *RFID*. Através do site é possível cadastrar a etiqueta *RFID* e a gondola do supermercado em que irá ser alocada, além de cadastrar os produtos e gerar o *QRCode* com as informações dos mesmos. O usuário possui um leitor *RFID* e ao chegar ao estabelecimento pode através de comandos de voz solicitar a localização de determinado produto, o aplicativo guiará o usuário até a gondola desejada. Após o usuário chegar ao local desejado é possível com que ele realize a leitura do *QR Code* do produto, assim sendo informado o preço, marca e outras informações relevantes.

(M.VARPE; WANKHADE, 2013) apresenta um sistema que utiliza etiquetas *RFID* implantadas no ambiente e módulos Zigbee para transmitir os dados entre servidor e o usuário. Previamente as etiquetas são cadastradas em um banco de dados com identificador e texto a ser informado, após isso são colocadas no ambiente. O usuário que utiliza uma bengala com leitor *RFID* recebe o sinal da etiqueta, através de um microcontrolador e do módulo *Zigbee* envia o identificador da etiqueta para o servidor e recebe a informação cadastrada. Após receber o texto, o microcontrolador utiliza a biblioteca de conversão de texto para voz (*TextToSpeech*) e reproduz a informação para o usuário, assim o usuário recebe uma descrição do local, facilitando sua locomoção.

A partir da análise dos resultados presentes na literatura é possível perceber uma diversidade de projetos com o mesmo foco, mas utilizando técnicas e materiais diferentes, assim como alguns pontos em comum como a utilização de *TextToSpeech* e o uso de *RFID*.

3 Metodologia

Inicialmente foi realizada uma revisão da literatura para identificar as alternativas para o desenvolvimento de um dispositivo de apoio para localização de pessoas com deficiência visual, seguido de um estudo amplo de tecnologias assistivas e uma proposta de solução para o problema de falta de informação sobre o ambiente que normalmente acomete pessoas cegas ou com baixa visão.

Após o estudo realizado foi modelado um dispositivo para auxiliar a localização

de pessoas cegas ou com baixa visão utilizando tecnologias já existentes, dentre elas uma câmera, um microprocessador embarcado e um *smartphone*, afim de fornecer mais informações sobre o ambiente as pessoas com estas necessidades. E após o desenvolvimento foram realizados testes de performance e viabilidade do protótipo, estas etapas serão descritas nos próximos capítulos.

4 Fundamentação Teórica

4.1 Tecnologia Assistiva

Tecnologia assistiva, segundo (CARVALHO *et al.*, 2016) é um termo utilizado para descrever algo que *assiste, ajuda, auxilia*, em termos práticos pode ser confecção de ajudas técnicas e a prestação de intervenção tecnológica para pessoas com deficiência.

No Brasil, o Comitê de Ajudas Técnicas definiu tecnologia assistiva como:

> uma área do conhecimento, de característica interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social.(TÉCNICAS, 2009)

Assim, de forma simples, é possível definir tecnologia assistiva como produtos e serviços que de alguma forma auxiliam pessoas com deficiência na melhora da sua qualidade de vida e independência.

4.1.1 Categorias de Tecnologia Assistiva

Segundo o documento apresentado pelo Comitê de Ajudas Técnicas, os recursos de Tecnologias Assistivas são classificados de acordo com os seus objetivos e/ou funcionamento a que se destinam. Dessa forma, foram definidas diversas classificações de Tecnologias Assistivas. Abaixo serão citadas as classificações que abrangem pessoas com baixa visão ou cegueira, que é o foco do trabalho.

- Recursos de Acessibilidade ao Computador

  Coleção de ferramentas de software e hardware que são produzidos para tornar o computador acessível a pessoas com privações sensoriais. Inclui dispositivos de entrada, como mouse, teclado ou outro dispositivo especifico, e dispositivos de saída , como sons, imagens e informações táteis. Na figura 1 é possível observar exemplos de recursos.

- Sistemas de Controle de Ambiente

  Através de um controle remoto as pessoas com limitações, podem ligar, desligar e ajustar aparelhos eletro-eletroeletrônicos. Este acionamento pode ser feito de forma direta ou indireta através de sensores presentes no ambiente. Atualmente existem sistemas com acionamento por voz, com o auxilio de uma assistente virtual como a desenvolvida pela Amazon. Outros sistemas podem ter acionamento por pressão, tração, sopro dependendo da limitação do usuário. A figura 2 apresenta exemplos de controle de ambiente.

Figura 1 – Exemplos de Recursos de Acesso ao Computador

Fonte: Autor, 2022.

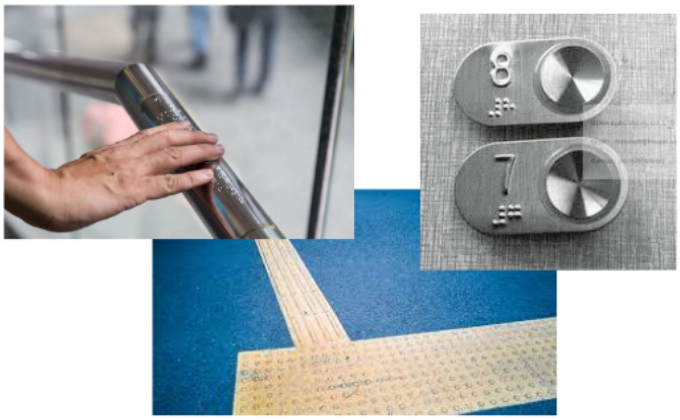
Figura 2 – Exemplos de Sistemas de Controle de Ambiente

Fonte: Autor, 2022.

- Projetos Arquitetônicos para Acessibilidade

  Projetos de edificação e urbanismo que garantem acesso e mobilidade a todas as pessoas, independente de sua condição física e sensorial. Ajustes estruturais e reformas na casa e/ou ambiente de trabalho, por meio de rampas, elevadores, adaptações em banheiros, mobiliário entre outras, que removem ou diminuem as barreiras físicas. Além destes projetos para a mobilidade de pessoas com deficiência física esta categoria engloba os projetos como calçadas com guias, auto relevos em braile em elevadores, corrimões, banheiros, entre outros. Na figura 3 pode-se observar exemplos projetos arquitetônicos para acessibilidade.(GALVÃO FILHO, 2009).

Figura 3 – Exemplos de Projetos Arquitetônicos para Acessibilidade



Fonte: Autor, 2022.

- Auxílio para Cegos ou para Pessoas com Baixa Visão

  Recursos projetados para auxiliar a visão como lupas e lentes, máquinas de Braile, softwares de leitura de tela e outros dispositivos que utilizam síntese de voz para a comunicação com o usuário. Inclui-se os animais adestrados para acompanhar as pessoas no dia a dia. A figura 4 mostra exemplos de auxílios

Figura 4 – Exemplos de Ferramentas de Auxílio para Cegos ou para Pessoas com Baixa Visão

Fonte: Autor, 2022.

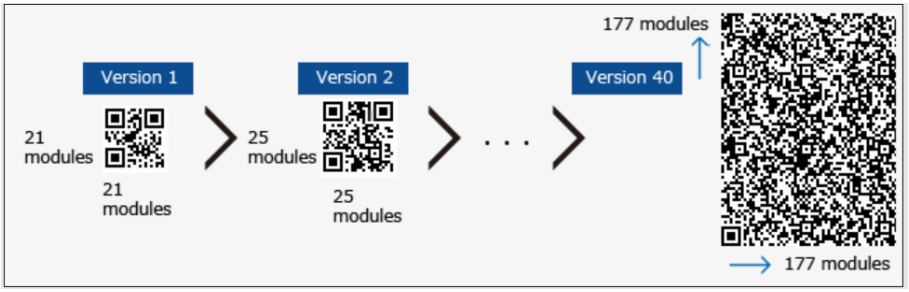
4.2 *QR Code*

QR Code, do inglês *Quick Response Code* (Código de resposta rápida) é um código em matriz bidimensional que foi projetado levando em consideração dois pontos principais, primeiro deve armazenar grande quantidade de dados em comparação com códigos de barras 1D e segundo deve ser decodificado em alta velocidade usando qualquer dispositivo portátil como telefones (TIWARI, 2016). Além disso, possui digitalização rápida, legibilidade omnidirecional, correção de erros para códigos danificados e diferentes versões.

4.2.1 Capacidade de Armazenamento

Segundo (QRCODE. . . , s.d.), a capacidade de armazenamento do *QR Code* varia com a versão, iniciando com 21 módulos na versão 1 até 177 módulos na versão 40. O módulo refere-se aos pontos pretos e brancos que compõem o *QR Code*. Configuração do módulo refere-se ao número de módulos contidos em um símbolo. Cada versão do *QR Code* tem uma capacidade de armazenamento máxima de dados, de acordo com a quantidade de módulos, tipo de caractere e nível de correção de erros. A figura 5 ilustra a configuração do módulo de algumas versões do *QR Code*.

Figura 5 – Configuração do módulo *QR Code*



Fonte: *Qrcode.com*, 2021

4.2.2 Correção de Erro

Conforme (TIWARI, 2016), o *QR Code* possui correção de erros que gera uma série de palavras de código de dados que permite que o símbolo seja lido mesmo sujo ou danificado. É aplicado um poderoso recurso de correção de erros usando códigos *Reed-Solomon*, um método matemático de correção de erros. O *QR Code* dispõe de 4 níveis de correção:

- Nível L : 7% de correção
- Nível M : 15% de correção
- Nível Q : 25% de correção
- Nível H : 30% de correção

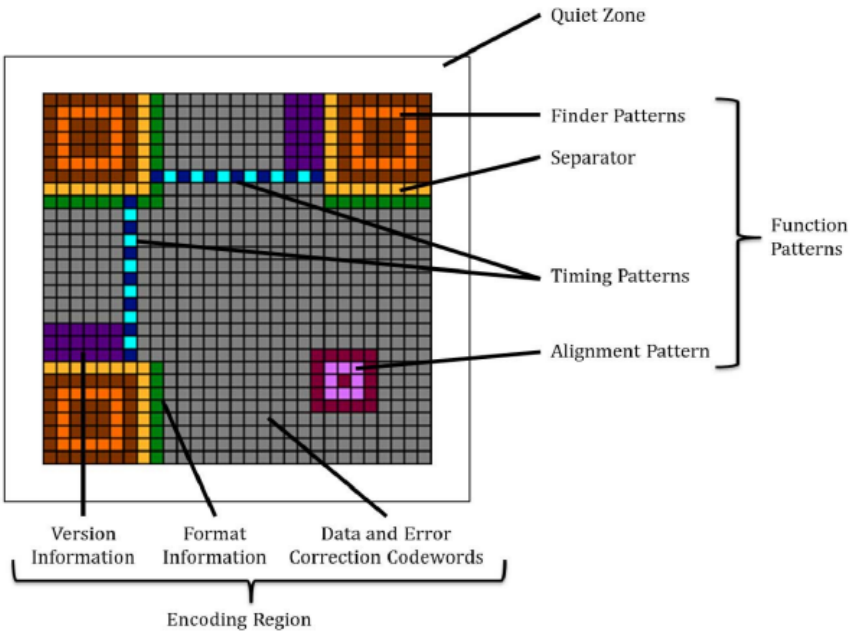
Os níveis Q e H são utilizado em fábricas ou outras aplicações em que o *QR Code* pode ficar sujo ou danificado. Para ambientes limpos o nível L é utilizado, mas no geral o

nível M é mais utilizado. É importante salientar que quanto maior o nível de correção de erros menor será o espaço para dados no *QR Code*.

4.2.3 Estrutura do *QR Code*

Segundo (TIWARI, 2016), cada símbolo *QR Code* é construído com módulos dispostos em uma matriz quadrada, que deve conter as funções padrões e região de codificação. Além disso, todo símbolo deve possuir uma borda de 4 módulos, chamada de zona silenciosa, esta área é usada para garantir que o texto ou marcações ao redor não desviem os dados. Outra parte importante é a região de codificação, que contém as informação de formato, versão, dados e correção de erros. As funções padrões são a forma que os módulos são colocados, assim garantindo que os *scanners* possam identificar e orientar corretamente o código para decodificação. Na figura 6 pode-se observar um exemplo.

Figura 6 – Padrões no *QR Code*



Fonte: (TIWARI, 2016)

Existem 4 funções padrões: localizadora, separadora, de temporização e de alinhamento, abaixo serão descritas suas principais caraterísticas:

- Função Padrão Localizadora: Este padrão possui a função de localizar e orientar corretamente o *QR Code* para decodificação, para isso utiliza alguns módulos em três cantos (superior esquerdo, superior direito e inferior esquerdo) em cada símbolo. Cada localizador consiste em um quadrado preto externo com 7×7 módulos, um quadrado branco interno com 5×5 módulos e um quadrado preto sólido no centro com 3×3 módulos.
- Função Padrão Separadora: Os separadores são as áreas entre cada padrão localizador e região de codificação.

- Função Padrão de Temporização: Há 2 padrões de temporização, o padrão de temporização horizontal e padrão de temporização vertical que se alternam em módulos pretos e brancos. Esses padrões são úteis para determinar a densidade do símbolo, as coordenadas do módulo e a área de informações da versão.

- Função Padrão de Alinhamento: É um padrão que possui 5 x 5 módulos pretos, 3 x 3 módulos brancos e um único módulo preto no centro. Os *QR Codes* da versão 2 e superior devem possuir padrões de alinhamento e a quantidade de padrões de alinhamento depende da versão do símbolo.

4.2.4 Codificação e Decodificação de *QR Code*

Para a codificação do *QR Code* são seguidos alguns passos que serão descritos abaixo:

- Análise dos Dados

  Para entender esta etapa é necessário saber que o padrão do *QR Code* possui quatro modos de codificação: numérico, alfanumérico, *byte* e Kanji. Cada modo codifica o texto para bits, mas cada um utiliza um método para a conversão. Deste modo, a análise dos dados vem para verificar se o texto pode ser codificado em numérico, alfanumérico, *byte* ou Kanji para obter a codificação mais otimizada possível.

- Codificação dos Dados

  Nesta etapa será criada uma sequência de bits que é dividida em palavras-chave *codewords* de dados, cada uma com 8 bits de comprimento. Para definir qual modo de codificação será utilizado se tem o indicador de modo *Mode Indicator*, que é uma sequência de 4 bits. O indicador de modo deve estar no inicio da codificação de dados. Além disso, é necessário saber a quantidade de caracteres que serão codificados e são representados pela sequência de *bits* conhecida como indicador de contagem de caracteres (*Character Count Indicator*). O indicador de contagem de caracteres é colocado após o indicador de modo e seu comprimento depende da versão.

- Código de Correção de Erros

  É gerada uma sequência de bits de dados que representam nosso texto, esses bits são usados para gerar palavras-chave de correção de erros usando um processo chamado *Reed-Solomon*. Os *scanners QR* leem as palavras-chave de dados e as palavras-chave de correção de erros. Ao comparar os dois, o *scanner* determina se os dados estão corretos ou não e, se não estiverem corretos, pode corrigir erros.

- Estrutura Final da Mensagem

  Os dados e as palavras-chave de correção de erros geradas nas etapas anteriores devem agora ser organizadas na ordem correta. Para *QR Codes* grandes, os dados e as palavras-chave de correção de erros são gerados em blocos, e esses blocos devem ser intercalados de acordo com a especificação do *QR Code*.

- Colocação do Módulo na Matriz

  Após gerar as palavras-chave de dados e palavras-chave de correção de erros e organizá-las na ordem correta, você deve colocar os *bits* na matriz de *QR Code*. As palavras-chave são organizadas na matriz de uma maneira específica.

- Adicionar Máscara aos Dados

  Certos padrões na matriz de *QR Code* podem dificultar a leitura correta do código por *scanners*. Para neutralizar isso, a especificação do *QR Code* define oito padrões de máscara, que altera o código de acordo com o padrão.

- Adicionar Informações de Formato e Versão

  A última etapa é adicionar informações de formato e (se necessário) de versão ao *QR Code* adicionando *pixels* em áreas específicas do código que foram deixadas em branco nas etapas anteriores.

Ao realizar a decodificação de um *QR Code* é utilizado o processo reverso do processo de codificação.

4.3 Sintetização de Voz

Conforme afirmado por (SANTOS; DORNELLES, 2008), a síntese de voz é vital para as ferramentas de tecnologias assistivas permitindo que barreiras do ambiente sejam removidas para as pessoas com limitações visuais. A síntese de voz já é utilizada em aplicações de leitura de tela, como *JAWS* desde 1989, mas existem outras aplicações que aplicam a síntese de voz.

Síntese de voz é o técnica de produção artificial de voz humana. Segundo (LOPES, 2015) para produzir uma síntese eficiente é necessário conhecer as características do sinal de voz. Primeiramente, a voz possui uma limitação de frequência em torno de 10 kHz. Outra característica é que a energia do sinal da voz é focada na faixa de frequência entre 300 e 3400 Hz.

As principais características da síntese de voz são a naturalidade e a inteligibilidade. A naturalidade representa o quanto próximo o som da síntese é da voz humana, enquanto a inteligibilidade descreve o quanto o som é compreensível.

Para isso é necessário compreender que o sinal de voz dispõe de duas classificações, sonoros e surdos, de acordo com a vibração ou não das cordas vocais. Os sinais surdos são os fonemas com peculiaridade de ruído, como "S"e "CH". Em contra ponto, os sinais sonoros são as vogais e consoantes não ruidosas.

Nos sinais sonoros, a vibração das cordas vocais é baseado em uma frequência fundamental. As demais harmônicas determinam o timbre, modelando assim a onda periódica. Isso é o que permite o reconhecimento do fonema e também do interlocutor, já que o timbre é a principal característica para distinguir vozes de mesma frequência que sejam emitidas por pessoas diferentes. A amplitude do sinal de voz é o que define a intensidade do som e depende da força ou potência que a voz é produzida.

Utilizando essas características, surgiram ferramentas especializadas transformar texto em voz, conhecidos como *Text to Speech* (TTS). De acordo com (REN *et al.*, 2019), o TTS visa sintetizar a fala natural e inteligível a partir de um texto e tem sido um dos principais tópicos de pesquisa no campo da inteligência artificial. A pesquisa sobre TTS mudou de síntese paramétrica estatística para síntese paramétrica baseada em rede neural e modelos de ponta a ponta, e a qualidade da fala sintetizada por modelos de ponta a ponta está próxima da paridade humana.

Segundo (CARVALHO, 2021), estes são os principais processos que ocorrem nas ferramentas *Text to Speech*:

- Normalização de texto: Converter o texto de entrada em uma série de palavras.
- Análise linguística: Determinar a informação semântica como frases.
- Análise lexical: Definir a forma de pronunciar e identificar as sílabas.
- Geração prosódia: Determinar quando fazer pausas e a acentuação, duração e pronúncia das palavras
- Síntese de diálogo: Gerar a informação de áudio, geralmente concatenando unidades de diálogo

4.4 Acessibilidade em sites

Sabendo que o intuito do projeto é oferecer um auxílio na obtenção de informações sobre o ambiente para pessoas com deficiência visual, é necessário conhecer padrões de acessibilidade em sites/aplicativos utilizados no mundo todo. Há um documento internacional que contém as recomendações de acessibilidade para conteúdo *online*, chamado de Diretrizes de Acessibilidade para Conteúdo *Web* (*Web Content Accessibility Guidelines - (*WCAG. . . , *s.d.)*).

Segundo (SILVA, 2021) o WCAG 2.1 é estruturado em camadas de orientações, desde elementos gerais até componentes mais específicos como uma nova autenticação. Os requisitos de acessibilidade possuem três níveis desde **A** com menor nível, **AA** e **AAA** com maior nível. As diretrizes de acessibilidade são sustentadas em quatro princípios: perceptível, operável, compreensível e robusto.

- Perceptível: as informações e componentes da interface devem ser apresentados em mais de uma forma, para que os usuários percebam as informações apresentadas.
- Operável: os elementos da interface e a navegação devem ser operáveis, possibilitando os usuários de realizar diversas operações da interface.
- Compreensível: as informações nas páginas *web* devem aparecer e funcionar de modo que os usuários venham a compreender.
- Robusto: o conteúdo deve ser robusto para poder ser interpretado de forma confiável por uma variedade de mecanismos de usuário, incluindo tecnologias assistivas.

5 Proposta

5.1 Visão Geral

O projeto visa oferecer um auxílio na obtenção de informações sobre o ambiente para pessoas cegas ou com baixa visão, através de um dispositivo com uma câmera que captura imagens e detectará *QR Codes*. Estes *QR Codes* foram pré-colocados no ambiente com orientações/informações sobre o mesmo, assim que capturados pela câmera serão processados por um microcontrolador embarcado, assim que a informação for decodificada será reproduzida ao usuário através de um aplicativo que também contará com a criação dos *QR Codes*.

Conforme a classificação apresentada no capítulo 4.1, o projeto se encaixa em **Auxílio para Cegos ou para Pessoas com Baixa Visão** e também pode conter

elementos da categoria **Projetos Arquitetônicos para Acessibilidade**, pois conta com pequenas modificações do ambiente, com a colocação dos *QR Codes* pelo espeço físico.

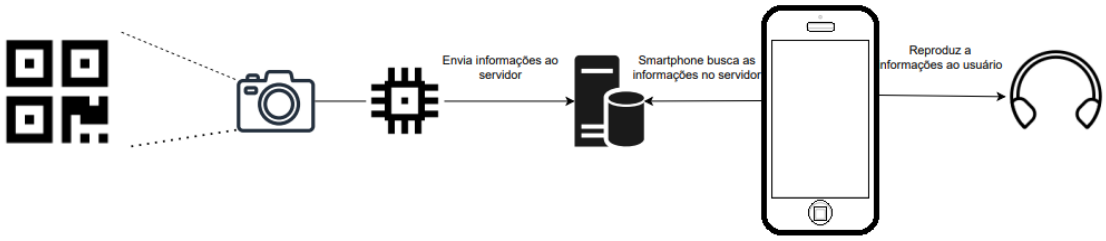
No esquema da figura 7 é possível observar como usuário irá interagir com o dispositivo. A câmera foi acoplada em um óculos e conectada diretamente com o microcontrolador. Ao encontrar um *QR Code*, o sistema irá decodifica-lo o e enviará a decodificação ao servidor, assim o aplicativo consegue buscar a informação no banco de dados e reproduzir ao usuário.

Figura 7 – Esquema do usuário com o dispositivo

Fonte: Autor, 2022.

Através do esquema da figura 8 é possível visualizar o fluxo de funcionamento do sistema, inicia-se com a leitura do *QR Code* pela câmera, o microprocessador atua na decodificação do mesmo e realiza o envio para o banco de dados. Então, o aplicativo executa a busca de informações no servidor, se houver dados a serem relatados ao usuário, o aplicativo reproduz por áudio ao usuário, através do *smartphone*.

Figura 8 – Esquema do funcionamento da câmera, microcontrolador e *smartphone*



Fonte: Autor, 2022.

5.2 Projeto do Hardware

No inicio do projeto elencou-se possibilidades de hardwares a serem trabalhados, assim foi realizado um estudo de quais alternativas eram viáveis e poderiam ser usadas para o desenvolvimento do projeto. Assim, utilizou-se no primeiro protótipo o componente chamado *ESP32-Cam*, definido devido ao baixo custo do dispositivo, em torno de R$ 70,00. Além disso, o *ESP32-Cam* possui as características necessárias para a realização do projeto

como conexão *wi-fi* integrada e câmera de fácil incorporação. As especificações detalhadas podem ser vistas na imagem 9.

Figura 9 – Especificações técnicas da placa *ESP32-Cam*.

- CPU: Xtensa® Dual-Core 32-bit LX6;
- 448 KBytes de memória ROM;
- 520 KBytes de memória RAM;
- 4 MBytes de memória Flash;
- Suporte a câmera OV2640 e OV7670;
- Clock máximo: 240 MHz;
- Conexão Wifi 2.4 GHz (máximo de 150 Mbps);
- Suporte para cartão SD;
- Conector micro-USB;
- 16 Portas GPIO;
- Tensão de operação: 5 V;
- Antena embutida;
- Conversor analógico digital;
- Distância entre os pinos: 2,54 mm;
- Tamanho: 64 mm X 22 mm X 5 mm (Largura X Profundidade X Altura);

Fonte: Autor, 2021.

A câmera utilizada neste protótipo é a *OV2640*, que possui 2 MP e conexão com o *socket* integrado da *ESP32-Cam*. Por fim, realizaram-se os testes do aplicativo através de um *smartphone* Motorola ®*One Fusion*, que possui 4 GB de memória RAM e processador *Qualcomm Snapdragon 710, (2.2 GHz Octa-Core)*.

Para o segundo protótipo foram alterados os recursos de hardware. Foram utilizados uma *Raspiberry Pi 3 Model B+* e a *HP Webcam HD-4110* para realizar a captura de imagens. Na figura 10 é possível visualizar as especificações da placa.

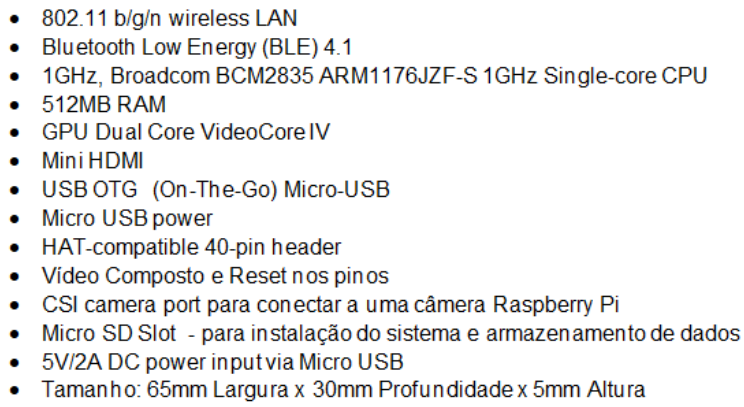
Figura 10 – Especificações técnicas da placa *Raspberry Pi 3 Model B+*.

- Processador Broadcom BCM2837B0 64bits ARM Cortex-A53 Quad-Core
- Clock 1.4 GHz
- Memória RAM: 1GB
- Adaptador Wifi 802.11 b/g/n/AC 2.4GHz e 5GHz integrado
- Bluetooth 4.2 BLE integrado
- Conector de vídeo HDMI
- 4 portas USB 2.0
- Conector Gigabit Ethernet over USB 2.0 (throughput máximo de 300 Mbps)
- Alimentação: recomendamos uma fonte DC chaveada 5V 3A
- Interface para câmera (CSI)
- Interface para display (DSI)
- Slot para cartão microSD
- Conector de áudio e vídeo
- GPIO de 40 pinos
- Dimensões: 85 x 56 x 17mm

Fonte: Autor, 2022.

Para o terceiro protótipo foram utilizados a placa *Raspberry Pi Zero W* e a câmera *OV5647* que possui a conexão por meio de um cabo *flat*. As especificações da placa podem ser visualizadas na imagem 11.

Figura 11 – Especificações técnicas da placa *Raspberry Pi Zero W*



Fonte: Autor, 2022.

5.3 Projeto do Software

Para o software do aplicativo, nomeado de *Be My Guide* que pode ser traduzido para *Seja Meu Guia*, foi definida a tecnologia *React Native* para a implementação, por se tratar de uma tecnologia atual com comunidade ativa e uma vasta documentação. Além disso, foi utilizado uma ferramenta de desenvolvimento *open-source* denominada *Expo* que disponibiliza *APIs* nativas do *smartphone* sem a necessidade de instalar qualquer dependência.

Além do *Expo*, utilizou-se a biblioteca *qrcode* do *JavaScript* para a geração das imagens. Foi empregada a biblioteca *axios* para realizar a requisição Http fazendo a conexão com o servidor *PHP*, por se tratar de algo difundido na comunidade.

Para o desenvolvimento do software embarcado, de leitura do *QR Code*, no primeiro protótipo utilizou-se a *Arduino IDE*, pois possuí suporte ao dispositivo *ESP32-Cam*. Para isso foi necessário instalar as bibliotecas de conexão com a placa modelo *ESP32-Cam*. Além disso, utilizou-se de bibliotecas de conexão *wi-fi* e de detecção e decodificação de *QR Code*.

No segundo e no terceiro protótipo foi utilizado um *script* em linguagem *Python* com base na biblioteca *OpenCV* para a captura das imagens e decodificação do *QR Code*.

A fim de unir as informações do aplicativo e do dispositivo embarcado criou-se um servidor em *PHP*. O servidor faz a conexão entre o dispositivo embarcado e o banco de dados, que será explicado abaixo, e também a conexão entre o aplicativo e o banco de dados. Foi utilizado a linguagem de programação *PHP*, pela rapidez de prototipação e facilidade de conexão com o banco de dados *MySQL*.

Foi empregado o banco de dados *MySQL* e estruturado em duas tabelas, que foram nomeadas **log** e **dispositivo**. A tabela log contém 5 colunas (idLog, texto, idDispositivo, verificado, data_inclusao). A coluna “idLog” é o índice único da tabela, o campo “texto” contém as informações do *QR Code* lido, a coluna “idDispositivo” define qual dispositivo

incluiu o valor no banco de dados, assim somente o aplicativo deste dispositivo poderá acessar este valor. O campo “verificado” informa se este log já foi executado ao usuário. A coluna “data_inclusao” informa a data que o log foi incluído. Já, a tabela dispositivo contém 2 colunas “idDispositivo” e “nome”, somente para identificar o dispositivo. Na figura 12 é possível observar o diagrama do banco de dados.

Figura 12 – Diagrama lógico do banco de dados do projeto.

Fonte: Autor, 2022.

A cada leitura de *QR Code* o protótipo acessa o servidor *PHP* enviando as informações de texto (dado decodificado do *QR Code* lido) e idDispositivo para a inserção na tabela log. Assim, o aplicativo faz uma busca filtrando pelo idDispositivo e verificado igual a zero, reproduz a mensagem ao usuário e atualiza o campo verificado para "1", encerrando o ciclo deste *QR Code*.

6 Implementação

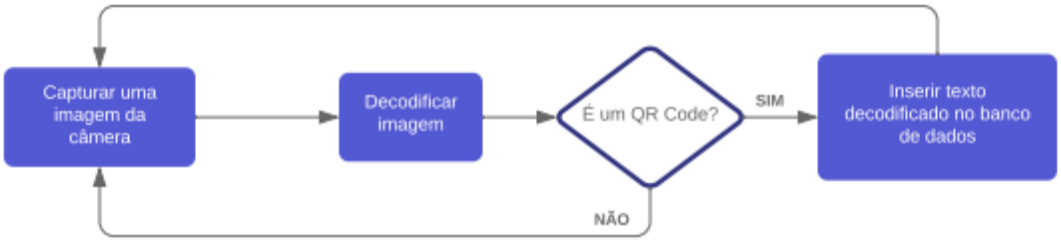
Nesta seção serão abordados as ferramentas utilizadas para a criação dos protótipos e suas particularidades, tanto de hardware quanto de software. Mesmo utilizando tecnologias diferentes o principio de execução do algoritmo de reconhecimento de um *QR Code* é o mesmo, na figura 13 é possível observar o funcionamento do software de decodificação do *QR Code* através de um fluxograma.

O sistema capta uma imagem da câmera, como se fosse um *screenshot* do vídeo que a câmera esta visualizando, envia esta imagem para uma função de detecção e decodificação de *QR Code*, assim se o algoritmo conseguir decodificar a imagem, grava a informação decodificada no banco de dados para ser transmitido ao usuário. Caso não conseguir decodificar a imagem, o sistema recebe uma resposta falsa. Nos dois casos, o sistema retorna ao inicio do *loop* para capturar outra imagem e repetir o processo.

6.1 Primeiro Protótipo

Na imagem 14 é possível observar como ficou disposto o dispositivo *ESP32-Cam* juntamente com a câmera *OV2640* para utilização no projeto. Como utilizou-se a interface *Arduino* para o desenvolvimento desta parte do sistema é necessário explicar que por padrão há duas funções principais neste tipo de arquivo, estas funções são denominadas *setup* e *loop*. Na primeira, ocorre a configuração das variáveis e bibliotecas que serão utilizadas

Figura 13 – Fluxograma de Funcionamento do Software de Detecção de *QR Codes*



Fonte: Autor, 2022.

no projeto, como a conexão *wi-fi* e configuração dos pinos da câmera. Já na função *loop*, como o próprio nome já diz, ela se repete até o componente ser desligado ou ocorrer um erro inesperado.

Após as configurações dos pinos da câmera e *wi-fi*, o sistema entra na função *loop*, captura uma imagem da câmera e envia para a função *recognition* da biblioteca *ESPino32QRCode*. Se o algoritmo encontrar uma decodificação para a imagem envia a resposta ao servidor *PHP* para gravação no banco de dados, se não encontrar um *QR Code* o algoritmo volta ao início da função *loop* para capturar uma nova imagem.

Figura 14 – Imagens do protótipo com o *ESP32-Cam*

Fonte: Autor, 2022.

6.2 Segundo Protótipo

Buscando comparar os resultados obtidos no primeiro protótipo, para esta etapa foi utilizado a placa *RaspBerry Pi 3 Model B+*. Para o desenvolvimento do software do sistema embarcado, utilizou-se a linguagem *Python* devido a ampla documentação e pela diversidade de bibliotecas disponíveis. Na figura 15 é possível verificar como o protótipo foi disposto para uso.

Na construção do algoritmo foi utilizada a biblioteca *OpenCV* que realiza a captura das imagens e também possui um módulo que procura e decodifica um *QR Code* em uma imagem denominado *QRCodeDetector*.

Figura 15 – Imagens da Placa *RaspBerry Pi 3 Model B+* e câmera HP

Fonte: Autor, 2022.

6.3 Terceiro Protótipo

Por questões de espaço físico ocupado pelo protótipo 2, fez-se necessário um terceiro protótipo para verificar o desempenho de um modelo de *Raspberry* mais leve e que ocupasse menos espaço, neste caso optou-se pelo modelo *Raspberry Pi Zero W*. Além de alterar a placa, foi trocada a câmera utilizada para uma que fosse compatível com o soquete presente na placa modelo Zero. Na figura 16 pode-se visualizar como ficou organizado o protótipo 3 para o usuário.

O algoritmo utilizado foi o mesmo que no protótipo 2, pois havia compatibilidade de linguagem e de sistema operacional.

6.4 Aplicativo

No desenvolvimento do aplicativo utilizou-se a tecnologia *React Native*, o mesmo ficou responsável por gerar *QR Codes* e reproduzir ao usuário as informações dos *QR Codes* encontrados pela câmera do sistema embarcado. No primeiro protótipo utilizava-se a biblioteca *react-native-qrcode-svg* para a geração do *QR Code* que foi substituída pela biblioteca *qrcode* no protótipo final, por possuir uma forma mais eficiente de geração e *download* da imagem do *QR Code*. A qualidade da imagem baixada é *High Definition*(1280 *pixels* X 1280 *pixels*), assim é possível imprimir em uma folha A4 sem perder os detalhes.

Figura 16 – Imagens da Placa *RaspBerry Pi Zero W* e a câmera integrada

Fonte: Autor, 2022.

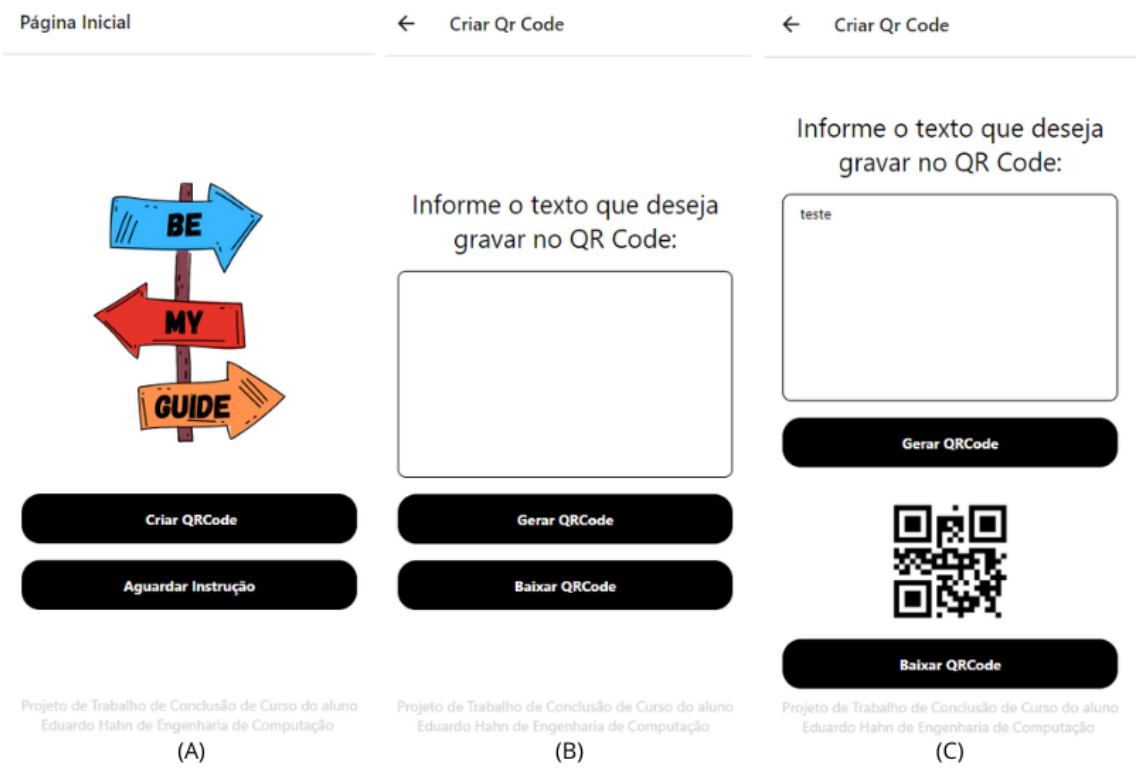
Na figura 17 é possível observar a página inicial do aplicativo (A), a tela de geração do *QR Code* antes do usuário digitar o texto desejado (B) e a tela de geração de QR Code após a geração do mesmo (C), quando já é possível visualizar o *QR Code* gerado pelo sistema. Utilizou-se de algumas diretrizes de acessibilidade para conteúdo *web* para definir o *design* do aplicativo, como o tamanho da fonte e contraste entre os botões.

Para transmitir a informação ao usuário foi criada uma tela denominada **Aguardar Instruções**, que basicamente aguarda uma leitura de *QR Code* para reproduzir ao usuário a mensagem. Nesta tela foi empregado um *TextToSpeech* da biblioteca *Speech* da ferramenta *Expo*. A biblioteca possui uma função chamada *speak* que recebe alguns parâmetros para reproduzir o texto desejado. No projeto utilizou-se a configuração padrão alterando apenas o parâmetro *language* que por definição é o inglês e foi substituído pelo português pt-BR.

6.5 Servidor

No servidor foram criados 4 arquivos denominados *conexao*, *gravar*, *atualizar* e *logs*. O arquivo *conexao* configura a conexão com o banco de dados MySQL e é importado nos demais arquivos. O arquivo *gravar* recebe os dados do sistema embarcado e insere os textos no banco de dados. O arquivo *logs* realiza uma busca dos logs seguindo as condições de verificado = 0 e idDispositivo = 1, no protótipo atual temos apenas 1 dispositivo cadastrado na rede. Já o arquivo *atualizar* modifica o campo verificado na tabela log para 1, assim encerrando o ciclo do log no sistema, pois este processo é feito após o texto ser reproduzido ao usuário.

Figura 17 – Telas do Aplicativo



Fonte: Autor, 2022.

7 Testes e Resultados

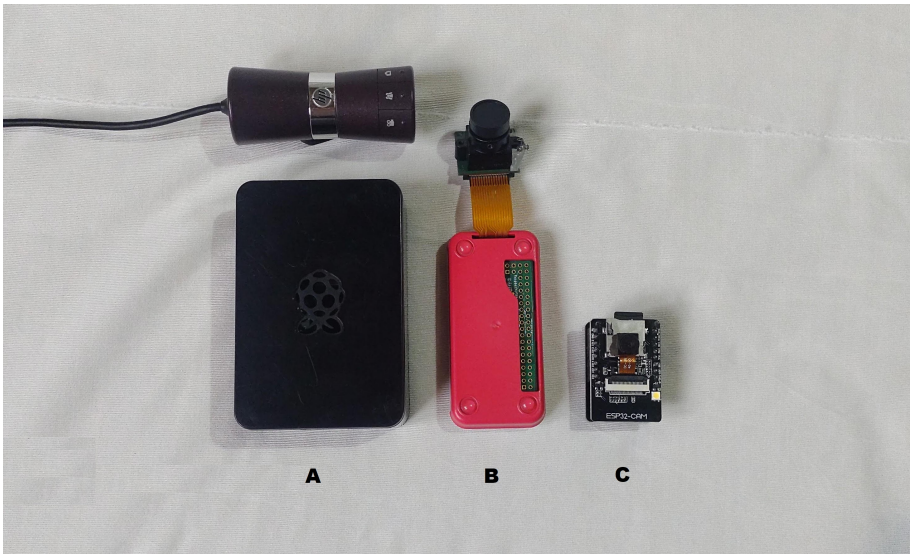
Na figura 18 é possível observar a comparação de tamanho das três placas utilizadas e suas respectivas câmeras. Primeiro protótipo, *ESP-Cam*, placa C, segundo protótipo composto pela *Raspberry Pi 3 Model B+* placa A e a câmera *HP Webcam HD-4110* e o terceiro protótipo constituído pela *Raspberry Pi Zero W* e câmera *OV5647*, placa B.

Como pode ser visto anteriormente, seção 6, as câmeras dos protótipos 1 e 3 foram colocadas na lente de um óculos e a câmera do protótipo 2 foi montada na parte lateral do óculos. Em todos os protótipos foram utilizadas presilhas plásticas para melhor fixação dos componentes ao óculos. O cabo de alimentação foi conectado diretamente ao *smartphone*, assim servindo de fonte de alimentação para os protótipos e ocorreram com 100% de efetividade. A fonte de alimentação pode ser substituída por uma bateria.

É importante ressaltar que os testes foram realizados em ambiente de laboratório, sem aplicação em ambiente real como a universidade por exemplo. Foi utilizada a casa do autor para testes realistas, verificando a efetividade e aumentando o embasamento sobre possíveis melhorias, mas entende-se que é necessário testar em um ambiente real para validação do sistema. O tamanho da imagem do *QR Code* utilizado para os testes foi 15 cm X 15 cm, ocupando aproximadamente metade de uma folha de papel A4.

Realizaram-se testes de performance tanto da leitura do *QR Code* quanto do aplicativo. A figura 19 é possível observar um gráfico que compara os três protótipos

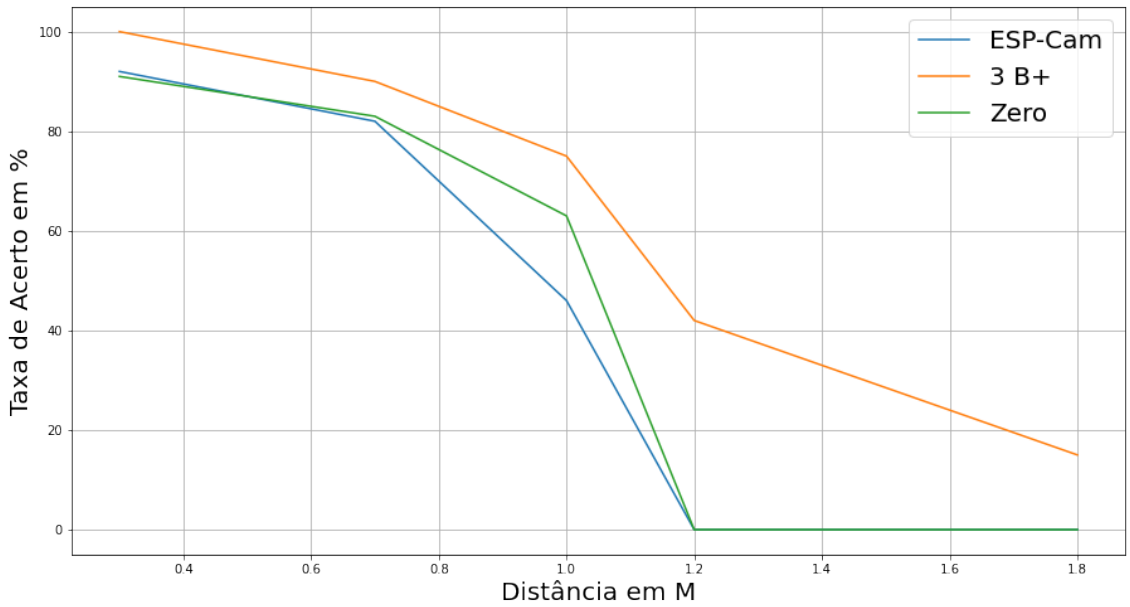
Figura 18 – Comparação dos três protótipos utilizados



Fonte: Autor, 2022.

considerando a taxa de acerto (porcentagem de resultados corretos ao ler uma imagem que possuía um *QR Code*) versus a distância que o *QR Code* estava da câmera.

Figura 19 – Gráfico com a Taxa de Acerto X Distância de cada protótipo

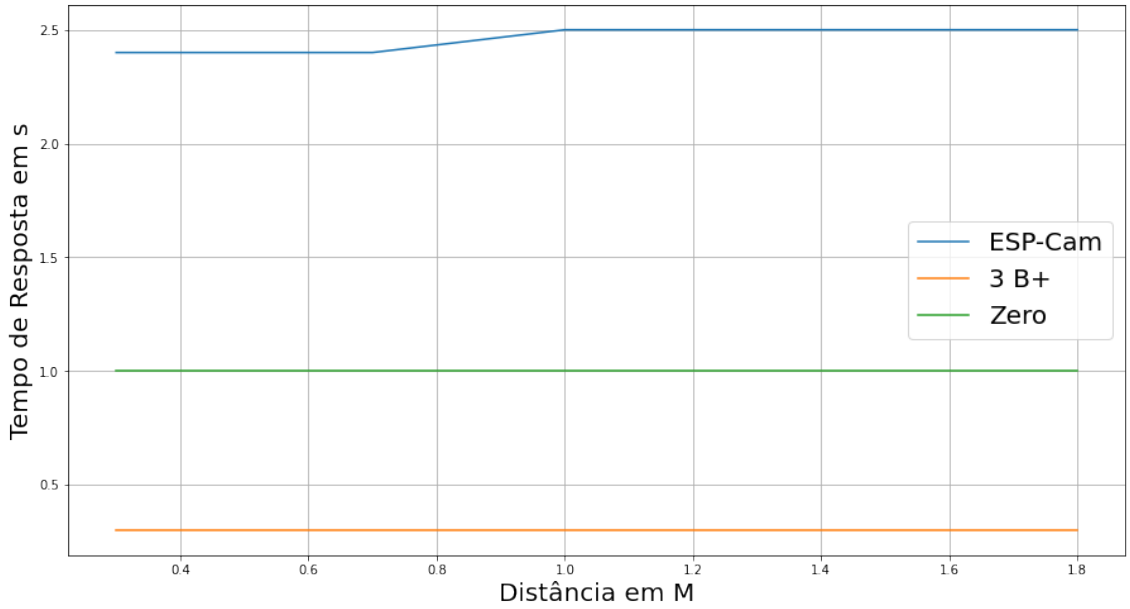


Fonte: Autor, 2022.

Outra informação relevante é o tempo de resposta do algoritmo de decodificação de um *QR Code*, pois quanto mais rápido o algoritmo for mais imagens podem ser processadas,

assim aumentando a chance de encontrar e decodificar um *QR Code* com informações para o usuário. Na figura 20 pode-se observar a comparação entre os três protótipos, inicialmente acreditava-se que aumentando a distância entre a câmera e o *QR Code* o tempo de resposta se alteraria, mas após os testes pode-se perceber que a diferença é miníma e ocorre realmente apenas em um dos protótipos.

Figura 20 – Gráfico com a Tempo de Resposta X Distância de cada protótipo



Fonte: Autor, 2022.

No teste com o aplicativo obteve-se uma boa performance, dentre os 20 testes realizados na criação e *download* do *QR Code*, nenhum retornou erro ou falha, gerando uma imagem pronta para ser impressa e colocada no ambiente. Com relação a transmitir a informação ao usuário obteve-se uma ótima performance, pois dentre os 20 testes realizados na inclusão de dados no banco o aplicativo reproduziu a mensagem por áudio, utilizando o *TextToSpeech*, corretamente, com tempo de resposta menor que 0,5 segundos.

Sobre a acessibilidade no aplicativo, foi aplicado os princípios de acessibilidade **compreensível** e **operável**, destacando ao contraste de cores empregado e a objetividade da interface de navegação. O nível de requisitos de acessibilidade efetuado foi de nível **A**, contendo poucos elementos de acessibilidade.

Durante os testes o sistema todo apresentou consistência de operação concluindo todo o fluxo apresentado durante o trabalho. O sistema conseguiu detectar e decodificar o *QR Code* com sucesso, enviar a decodificação ao servidor que gravou no banco de dados, assim o aplicativo buscou a informação e reproduziu ao usuário utilizando uma função *TextToSpeech*. O TTS utilizado conseguiu apresentar todas as informações de teste de forma clara, mas como foi empregada a configuração padrão a voz transmitida parecia robotizada.

Adiciona-se algumas ressalvas a conexão do *ESP32-Cam* com o servidor, que em 20% dos testes gerou um erro ao conectar ao servidor *PHP*, cancelando a operação de gravação e consequentemente de aviso ao usuário. Nos outros protótipos (desenvolvidos

com a linguagem de programação *Python*) estes erros de conexão com o servidor não ocorreram, pois neles foram utilizadas outra forma de requisição com o servidor.

8 Conclusão e Trabalhos Futuros

Durante este trabalho foram construídos três protótipos que fazem a leitura de uma imagem de câmera, encontram e decodificam o *QR Code*. Quando o sistema embarcado decodifica a mensagem do *QR Code* gerado pelo aplicativo, reproduz a mesma no *smartphone* do usuário utilizando um sintetizador de voz.

Como visto na seção 7, quanto maior a distância menor é a taxa de acerto e a partir de 1,2 metros essa porcentagem decai para menos de 50%, isto ocorreu devido a dificuldade de encontrar o *QR Code* em imagens mais distantes. Também é possível observar que o tempo de resposta não se altera com o aumento da distância, assim possuindo um processamento mais rápido é possível compensar a taxa de acerto, já que mais imagens por segundo serão verificadas. Portanto com base nos testes realizados é necessário um processamento rápido, igual ou superior ao protótipo 2, para que o sistema seja viável.

Comparando os protótipos pode-se observar que todos cumpriram o ciclo esperado, realizaram a leitura do *QR Code*, gravaram a informação no banco de dados e o aplicativo reproduziu a mensagem ao usuário através de um *TextToSpeech*, tendo ressalvas para o tempo de resposta do primeiro protótipo. Mas utilizando o preço das placas e a eficiência em consideração, temos uma equivalência, pois a placa do protótipo 2 possui um valor de mercado próximo a 10 vezes do valor da placa do protótipo 1 e o tempo de resposta é aproximadamente 10 vezes menor.

Na interface do aplicativo foram empregados dois princípios de acessibilidade, **compreensível** e **operável**, de nível **A**, entretanto sabe-se que é necessário expandir este nível desenvolvendo recursos de comandos por voz, para facilitar a operação do público alvo. Outro ponto de destaque na acessibilidade do aplicativo é a utilização do *TextToSpeech* que apesar de apresentar resultados satisfatórios ao transmitir a informação de modo claro, ainda há a necessidade de buscar que a voz se apresente com mais naturalidade.

Analisando o desenvolvimento e desempenho deste projeto, pode-se sugerir para trabalhos futuros realizar um pré-processamento na imagem, antes que a mesma seja enviada ao decodificador de *QR Code*, pois este é o principal custo na execução do sistema. O pré-processamento poderia ser realizado com tratamentos na qualidade da imagem ou dividindo a imagem em alguns elementos menores e submetendo uma parte de cada vez para o algoritmo de decodificação.

Outro acréscimo a este trabalho que trará benefícios em relação ao aplicativo é a criação um serviço que verifica atualizações na tabela *logs*, assim informando o usuário sobre novos *QR Codes* sem a necessidade do usuários estar com o aplicativo aberto em primeiro plano.

Além disso, é possível criar uma rede de informações pertinentes ao público alvo e adicionar novas ferramentas ao sistema, utilizando a câmera em conjunto com outros sensores e recursos do *smartphone* do usuário permitindo que seja possível identificar possíveis obstáculos, por exemplo, assim tornando-o mais robusto e com maiores benefícios a pessoas cegas.

Referências

BRASIL, 2004, P.2. [S.l.: s.n.]. <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/decreto/d5296.htm>. Accessed: 04/02/2022.

CARVALHO, João Miguel Fonseca de. **Modelo de interação flexıével num Editor de Modelos operado por voz e gestos**. 2021. Tese (Doutorado).

CARVALHO, Vanessa Fernandes *et al.* Tecnologias assistivas aplicadas a deficiência visual: recursos presentes no cotidiano escolar e na vida diária e prática. **Educere-Revista da Educação da UNIPAR**, v. 16, n. 1, 2016.

DUARTE, Karen *et al.* Information and Assisted Navigation System for Blind People. **International Journal on Smart Sensing and Intelligent Systems**, Exeley, Inc., v. 7, n. 5, p. 1–4, 2020. DOI: <10.21307/ijssis-2019-062>.

GALVÃO FILHO, Teófilo Alves. A Tecnologia Assistiva: de que se trata. **Conexões: educação, comunicação, inclusão e interculturalidade**, Redes Porto Alegre, v. 1, p. 207–235, 2009.

IBGE, Instituto Brasileiro de Geografia e Estatística. **Censo Demográfico 2010**. 2010. Disponível em: <<http://www.ibge.gov.br>>.

LOPES, Lidiane Furlan. **Aplicativo autolocalizador para pessoas com deficiência visual**. 2015. B.S. thesis – Universidade Tecnológica Federal do Paraná.

LÓPEZ-DE-IPIÑA, Diego; LORIDO, Tania; LÓPEZ, Unai. Indoor Navigation and Product Recognition for Blind People Assisted Shopping. In: AMBIENT Assisted Living. [S.l.]: Springer Berlin Heidelberg, 2011. P. 33–40. DOI: <10.1007/978-3-642-21303-8_5>.

M.VARPE, Kanchan; WANKHADE, M. P. Visually Impaired Assistive System. **International Journal of Computer Applications**, Foundation of Computer Science, v. 77, n. 16, p. 5–10, set. 2013. DOI: <10.5120/13565-1324>.

QRCODE.COM. [S.l.: s.n.]. <https://www.qrcode.com/en/about/version.html>. Accessed: 03/03/2022.

REN, Yi *et al.* Fastspeech: Fast, robust and controllable text to speech. **Advances in Neural Information Processing Systems**, v. 32, 2019.

RODRIGUES, Ivo; DUARTE, Lués; GUERREIRO, Lués Carriço Tiago. Ferramentas Contextuais para Pessoas Cegas. The Eurographics Association, 2020.

SANTOS, Calebe Augusto dos; DORNELLES, Gino. Text-to-speech: Sintetizador de voz para documentos como ferramenta de apoio a deficientes visuais, 2008.

SILVA, Danilo Faustino da. Acessibilidade Web: avaliando os portais dos Institutos Federais da região Nordeste, 2021.

TÉCNICAS, COMITÊ DE AJUDAS. **Tecnologia Assistiva**. 2009. Disponível em: <<http://galvaofilho.net/livro-tecnologia-assistiva_CAT.pdf>>.

TIWARI, Sumit. An Introduction to QR Code Technology. In: 2016 International Conference on Information Technology (ICIT). [S.l.: s.n.], 2016. P. 39–44. DOI: <10.1109/ICIT.2016.021>.

VEEVER. [S.l.: s.n.]. <https://veever.global/>. Accessed: 16/02/2022.

WCAG, WEB CONTENT ACCESSIBILITY GUIDELINES 2.1. [S.l.: s.n.]. <https://www.w3.org/TR/WCAG21/>. Accessed: 05/04/2022.