Internet das Coisas na integração de Aplicativo Móvel com Semáforo Eletrônico para Auxílio à Travessia de Deficiências Visuais

Luiz Henrique Naspolini dos Santos[1], Luciano Antunes[1]

[1]Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
Av. Universitária, 1105 - CEP: 88806-000 - Criciúma - SC- Brasil

luiznds62@gmail.com, luc@unesc.net

Abstract. *Technology can perform an important role in the life of persons who have visual impairment through solutions that raise the accessibility levels of urban displacement. This research has the objective to apply the Internet of Things and Cloud Computing concepts with a mobile application and a traffic light prototype with a color reader device to help blind persons to pass through streets with traffic lights. The application verifies the proximity of traffic lights based on the location of the mobile device constantly and when approaching it begins to perform vibration and sound alerts based on the current status. The results indicate that the application has good performance to alert the user in places that have access to Wi-Fi or 4G networks.*

Resumo. *A tecnologia pode ter um papel importante na vida de pessoas com deficiência visual por meio de soluções para o aumento da acessibilidade no deslocamento urbano. Esta pesquisa tem como objetivo aplicar o conceito de Internet das Coisas e Computação na Nuvem na integração de um aplicativo móvel, semáforo prototipado e protótipo de leitor de cores para auxiliar na travessia de deficientes visuais em ruas com semáforos. São verificados semáforos próximos ao smartphone constantemente e ao se aproximar, é iniciado o monitoramento das cores,realizando alertas sonoros e vibratórios baseados no status atual. Os resultados apontam que a aplicação tem um bom desempenho para alertar o usuário em locais com acesso a redes Wi-Fi ou 4G.*

1. Introdução

Existem dois tipos de deficiência visual, a cegueira e a baixa visão, sendo a primeira um tipo de limitação da forma de apreensão de informações, ou seja, a visão. É importante frisar que embora haja uma preocupação com a utilização de termos pejorativos existe diferença entre deficiente visual e a palavra cego em si, pois pode haver diferença em níveis de capacidade visual e o termo pode não englobar corretamente o que se quer descrever. Pode se dizer que a cegueira é uma privação da visão, que é um sentido importante porém não caracteriza a desabilitação do indivíduo e nem o torna menos capaz como ser humano, descrição que infelizmente ocorre em casos de preconceito [Nunes e Lomônaco 2014].

Pode se afirmar, segundo a OMS, que o indivíduo considerado portador de cegueira é aquele que possui uma acuidade visual de zero a 6 metros a frente, sendo que uma pessoa não portadora de deficiência visual pode enxergar estímulos em seu melhor olho de até 60 metros. Como a visão faz parte da integração de modalidades sensoriais diferentes, a compreensão das informações recebidas dos sentidos ocorre de forma unificada e relaciona um sentido com o outro em portadores de deficiências visuais, e como existe uma disfunção, podem ocorrer erros e cortes no processamento da

informação e conceitos dos indivíduos, logo, algumas barreiras e obstáculos são apresentados principalmente na interação e locomoção no ambiente em que se vive, assim como no transporte que é uma das etapas para se atingir locais básicos de ocupação humana como hospitais, clínicas, escolas e ambientes de lazer, que por direito previstos em lei estes cidadãos deveriam ter acesso [Fornaziero e Zulian 2010].

Com os aparatos tecnológicos à disposição, é possível desenvolver recursos e serviços para proporcionar ou ampliar condições mais favoráveis para promoção de uma vida mais independente e com mais inclusão para portadores de algum tipo de deficiência. A computadorização pode melhorar as habilidades funcionais de pessoas com deficiência por meio de produtos, recursos, práticas e estratégias para contemplar questões sobre acessibilidade, mobilidade, comunicação alternativa, acionamento de chaves, aparelhos para audição, dispositivos para auxílio visual e vários outros itens e segmentos do cotidiano [Sartoretto e Bersch 2020].

Segundo Galegale et al. (2016), a Internet das Coisas é uma das próximas tecnologias disruptivas e é esperado que em 2020 existam de cinquenta a cem bilhões de objetos conectados a internet, dentre eles alguns exemplos seriam uma casa sendo limpa por um aspirador de pó inteligente, um fogão inteligente preparando alimentos sozinho ou até mesmo sensores e câmeras mostrando o trânsito e suas ocorrências diversas em tempo real em centros de operações.

A computação na nuvem é uma tecnologia que vem rapidamente ganhando tração nos negócios, ela oferece serviços online comercializados sob demanda como Gmail, iCloud e Salesforce por exemplo, e também permite com que sejam cortados custos com hardware e suporte de Tecnologia da informação. Estes serviços têm como função processar, compartilhar e sincronizar dados, organizar serviços à clientes e gerenciar vendas além de um número cada vez maior de formas de utilização. Os recursos computacionais como software, espaço de armazenamento, poder de CPU e rotinas de manutenção são oferecidos em versões online escaláveis em grandes datacenters [Bruin e Florid 2019].

Como afirma Kerschbaumer (2018), os microcontroladores, também conhecidos pelo termo “computadores de um único chip”, são circuitos integrados, possuidores de um núcleo de processamento e memória, além de poderem ser acoplados com periféricos que podem assumir o papel captar a entrada e distribuir a saída de dados, sendo ligado em uma fonte de alimentação externa. São normalmente baratos de produzir e sua popularidade é proveniente da versatilidade a qual desempenham na prototipação de diversos dispositivos, podendo o mesmo circuito ser utilizado para gerar diferentes resultados variando somente o software.

Conforme o levantamento bibliográfico, foram encontradas poucas pesquisas com a finalidade de auxiliar a travessia de ruas em si, mas em contrapartida existem diversas sobre melhorias de acessibilidade por meio de dispositivos tecnológicos com o emprego de sensores ou aplicações móveis. O estudo realizado por Oliveira 2018 desenvolveu uma aplicação para ajudar a garantir a autonomia na travessia de ruas por deficiêntes visuais, com os dados extraídos diretamente de um semáforo prototipado em protoboard com microcontrolador e consumidos por um aplicativo móvel por meio de um webservice disposto na rede por um framework chamado ngrok, que realiza o

tunelamento de requisições HTTP para uma máquina local onde a autora da pesquisa executava seu servidor de aplicação. Os resultados foram bem sucedidos para alertar o usuário através do aplicativo de forma esperada apesar de ser relatado pontos de melhoria como a precisão da localização recebida no smartphone do usuário.

A pesquisa de Santos, Correia e Silva (2018) propôs o desenvolvimento de um protótipo de sistema para auxiliar deficientes visuais a realizar a travessia de ruas com apenas uma faixa de pedestre contando com botão de acionamento para transição de status do sinal e alerta sonoro. Os resultados apontaram que o protótipo cumpriu seus objetivos após alguns ajustes, sendo principalmente a mudança de sinais sonoros para simbolizar o início da fase de travessia que ainda sim não ficou conforme o ideal descrito do autor e a correção dos valores demonstrados no display.

O estudo de Siddesh, Manjunath e Srinivasa (2016) desenvolveu uma aplicação onde o usuário pode traçar uma rota de deslocamento onde é guiado por instruções de onde andar e ao mesmo tempo compartilha a localização do mesmo para pessoas cuidadoras. Foram atingidos os objetivos com sucesso e planejado pelos autores pontos de melhoria para o futuro no sentido de alcançar maior eficiência no algoritmo de rotas do aplicativo.

A pesquisa de Perakovic, Perila, Cvitic e Brletic (2017) propõe aplicar conceitos de IoT para auxiliar deficientes visuais em ambientes externos por meio de um dispositivo que descobre informações relevantes do ambiente e comunica com um aplicativo móvel em tempo real. A pesquisa concluiu que foi possível entregar com segurança e confiança os dados em tempo real.

O trabalho desenvolvido por Ramadhan (2018) se propôs a criar um dispositivo vestível para notificar o usuário de obstáculos, emitir sinais sonoros para solicitar auxílio de pessoas próximas e notificar contatos de emergências em caso de acidente. O autor conclui que o dispositivo cumpriu seus objetivos com sucesso e que a notificação de acidentes se mostrou muito atrativa.

De acordo com os estudos realizados foram encontrados poucos trabalhos com o foco em auxiliar pessoas na travessia de ruas com semáforos. Portanto, esta pesquisa tem como objetivo aplicar o conceito de Internet das Coisas em uma aplicação móvel com semáforo eletrônico para auxiliar a travessia de deficientes visuais.

2. Materiais e Métodos

A presente pesquisa foi do tipo aplicada, explicativa e de base tecnológica. Foi desenvolvido uma aplicação que empregou conceitos de IoT e *Cloud Computing* com a finalidade de auxiliar indivíduos portadores de algum tipo de deficiência visual a realizar a travessia de ruas com semáforos por meio de alertas vibratórios e sonoros em um aplicativo móvel, baseados em dados coletados por um sensor de cor dispostos em um servidor de aplicação na nuvem. Esta aplicação é constituída por quatro módulos, sendo eles: prototipo de semáforo, sensor de cores de semáforo, servidor de aplicação e aplicativo móvel (Figura 1). Foi utilizado uma instância do banco de dados *Postgres*, fornecido gratuitamente pela plataforma *ElephantSQL* com capacidade máxima de 20 MB de dados para armazenar informações dos semáforos e de forma opcional também dos usuários.

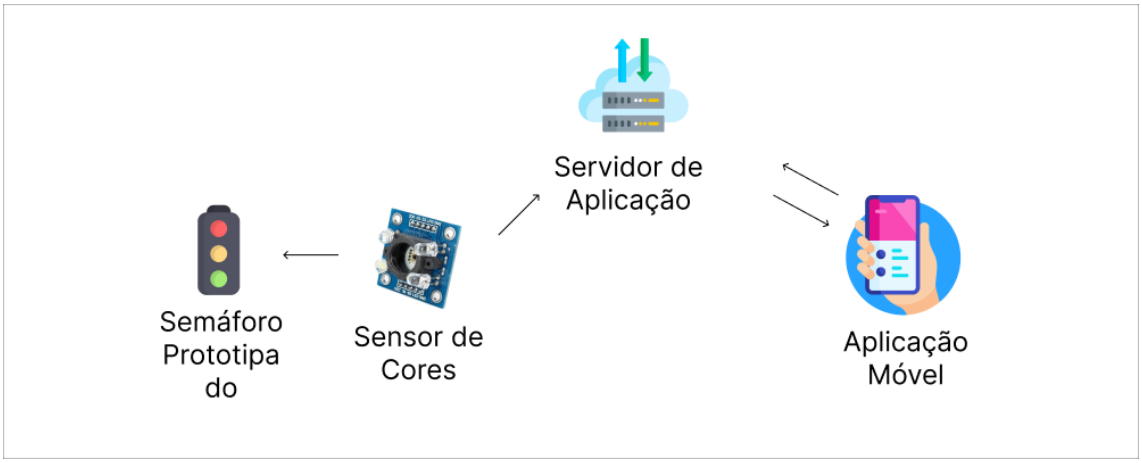

Figura 1. Módulos da aplicação

O protótipo de semáforo consiste em um anel de led RGB de 16x controlado por um microcontrolador que simula cores de um semáforo real com apenas uma lâmpada, sendo alternado em períodos fixos pré-definidos entre as cores vermelho, verde e amarelo. O módulo sensor de cores de semáforo é composto por um sensor que reconhece níveis de luz RGB e um microcontrolador, é responsável por realizar a leitura das cores emitidas pelo protótipo de semáforo de forma constante, onde comunica com o servidor de aplicação através da rede por meio de uma conexão sem fio Wi-Fi. O servidor de aplicação é disponibilizado na nuvem por uma PaaS *(Platform as a Service)* e é encarregado de armazenar dados de semáforos e usuários no banco de dados, além de dispor de uma API responsável por disponibilizar recursos para o aplicativo móvel. O aplicativo móvel realiza a interação com o usuário por meio de ferramentas como *Talkback* para dispositivos *Android* e *VoiceOver* em dispositivos *iOS*, também permite se identificar em uma etapa cadastral ou pular e seguir como convidado, monitora proximidade de semáforos à localização atual e dispõe alertas sonoros e vibratórios quando se aproxima a uma distância simulada de algum semáforo.

2.1 Desenvolvimento do protótipo de semáforo

No desenvolvimento do protótipo de semáforo, na parte de hardware, foi utilizado um anel com 16 LEDs RGB com um chip driver WS2812 embutido, um resistor de 330 Ohms (1 Watt) e um Arduino Uno R3 e uma fonte de alimentação de 9 Volts e um cabo USB compatível com Arduino. O custo total para montagem foi de R$137,00.

Para acoplar os componentes foi utilizada uma caixa de papelão com as seguintes dimensões 200x120x70mm. Os componentes foram fixados dentro da caixa por meio de fita adesiva.

A codificação se deu através da Arduino IDE em sua versão 1.8.13, utilizando a linguagem de programação C++. Por conta de se tratar de uma funcionalidade mais simplificada, não foram utilizadas bibliotecas externas. A implementação do *sketch* realiza a inicialização das variáveis que controlam os pinos e o *delay* de troca de cor, seguido do método *setup* onde os mesmos são delimitados como pinos de saída. No loop principal são realizadas chamadas a um método que realiza a alteração da cor do

anel de LED intercaladas por um delay de 15 segundos para a cor vermelha, 5 segundos para a cor amarela e 10 segundos para cor verde.

O método de alteração de cores recebe como parâmetro 3 valores do tipo inteiro, que podem variar de 0 a 255, e são utilizados na chamada da função *analogWrite* que faz parte do conjunto de bibliotecas nativas da Arduino IDE e serve para enviar um valor analógico para um pino. Para gerar a cor verde são enviados por parâmetro os valores 0 para os pinos que irão ativar luz vermelha, 255 para os pinos que irão ativar luz verde e 0 para os que irão ativar luz amarela. Para cor vermelha são enviados 255, 0, 0 e para amarela 255, 255 e 0 na ordem vermelho, verde e amarelo dos pinos.

Figura. 2 Prototipo de semáforo

2.2 Sensor de cores de semáforo

No desenvolvimento do sensor de cores de semáforo, na parte de hardware foi estudado tecnologias associadas a montagem de circuito eletrônico e sobre os componentes utilizados. Sobre o software foi necessário descobrir bibliotecas hábeis a auxiliar na comunicação via *websockets* com microcontroladores, sendo essa a parte onde mais levou-se tempo no desenvolvimento do que o restante, que se tratou da utilização do sensor de cor *TCS3200* e da conexão com a internet via microcontrolador *ESP8266*. O conjunto de componentes selecionados resultou num custo total de R$113,00.

2.2.1 Montagem do dispositivo

No desenvolvimento do dispositivo, parte de hardware, foram utilizados um microcontrolador *Wemos D1 R1 Mini*, o qual vem integrado com um módulo ESP8266 que possui conexão com *Wi-Fi*, um cabo USB-C para conexão entre o *Wemos D1 R1 Mini* com o computador, um sensor de cores *GY-31 TCS3200* da *TAOS*, e um rolo de papelão de 100x50x50x mm e 7 jumpers e fita crepe.

Para dispor os jumpers de um modo mais maleável e fácil de reclinar de forma ereta em superfícies foram unidos com fita crepe, e assim mais facilmente colocado por cima de um rolo de papelão com a função de posicionar o sensor TCS3200 de uma forma onde o mesmo pode realizar a leitura das cores mais assertiva conforme a Figura 3.

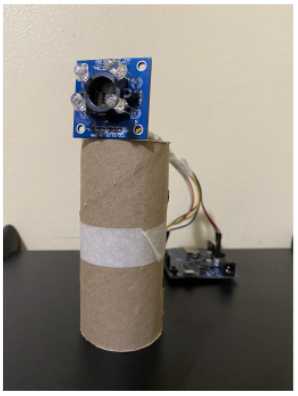

Figura. 3 Sensor de Cores de Semáforo

2.2.2 Montagem do circuito

A alimentação do circuito se deu pela conexão USB com cabo USB-C conectado ao *Wemos R1 D1 Mini*. O sensor de cor TCS3200 foi conectado ao 5V e GND diretamente da placa e não contou com auxílio de *protoboard*. Os pinos também foram conectados diretamente na placa sem ter sido necessário resistores, sendo o cabeamento feito por meio de jumpers conforme Figura 4.

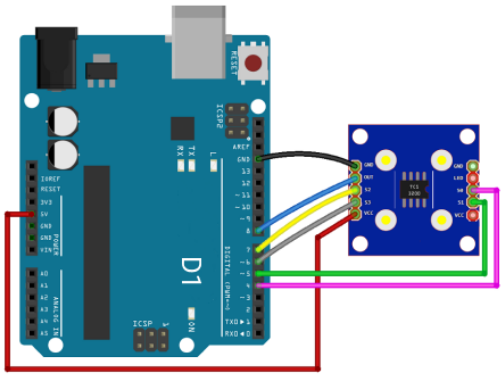

Figura. 4 Circuito eletrônico do sensor de cor de semáforos

2.2.3. Montagem do software

Foi utilizada a Arduino IDE (1.8.13) no desenvolvimento do software do sensor de cor de semáforos, rodando a linguagem C++. Foram utilizadas as bibliotecas ESP8266HTTPClient.h, WiFiClient.h e ESP8266WiFiMulti.h (ambas na versão 2.10), ESP8266WiFi.h na versão 2.70, disponibilizadas pela própria Arduino IDE e as bibliotecas ArduinoJson.h na versão 6.17, e WebSocketsClient.h na versão 0.50 obtidas gratuitamente através do Github dos autores, que respectivamente são: gilmaicon e bblanchon.

As bibliotecas ESP8266WiFi.h e ESP8266WiFiMulti.h foram utilizadas para estabelecer a conexão Wi-Fi com o microcontrolador através do módulo ESP8266 embarcado. Para realizar o consumo das funcionalidades dispostas no servidor de aplicação, foram utilizadas as bibliotecas ESP8266HTTPClient.h e WiFiClient.h, auxiliadas pela biblioteca ArduinoJson.h por se tratar do mesmo formato de conteúdo escolhido utilizado na comunicação. Para realizar a atualização da mudança de status do semáforo foi utilizada a biblioteca WebSocketsClient.h para auxiliar no estabelecimento da conexão via websocket com o servidor de aplicação.

O código do sensor de cores de semáforo foi dividido entre 4 classes auxiliares criadas para prover métodos utilitários e reutilizáveis, e o *sketch* principal onde existe o instanciamento das classes utilitárias, a criação do método callback do websocket, a configuração e o loop principal onde é verificado o resultado da leitura da cor em frente ao protótipo de semáforo.

No método de configuração (*setup*) é inicializado a conexão com Wi-Fi, requisitado ao servidor de aplicação via protocolo *HTTP*, a porta gerada na plataforma *Heroku* para conexão do websocket (por conta de se tratar de um serviço gratuito, ocorre o desligamento do servidor a cada 30 minutos de ociosidade, sendo inicializado novamente em outra porta escolhida de forma aleatória). Após isso, é estabelecida conexão via websocket com o servidor de aplicação, finalizando o método de configuração.

O loop principal é responsável por manter em loop a escuta e envio de mensagens pelo objeto de manipulação do websocket e realizar a leitura da cor pelo sensor, evento que ocorre a cada 100 milisegundos. A leitura da cor se dá através da alteração de nível lógico (*LOW* igual a BAIXO e *HIGH* igual a ALTO) nos pinos do sensor TCS3200 dinamicamente, conforme a Tabela 1 abaixo. Foi operado com configuração de escalonamento de frequência em 20%, definidos através do sinal lógico 1 e 0 nos pinos S0 e S1 respectivamente, a qual foi indicada pelo fabricante para trabalhar com Arduino ou placas similares.

Tabela. 1 Níveis lógicos da leitura de frequência de cor do sensor TCS3200

Tipo do fotodiodo	S2	S3
Vermelho	LOW	LOW
Azul	LOW	HIGH
Sem filtro de cor	HIGH	LOW
Amarelo	HIGH	HIGH

Fonte: Texas Advanced Optoelectronic Solutions Inc, 2009

As frequências dos fotodiodos com filtro vermelho, verde e azul são coletadas respectivamente através da função pulseIn disponibilizada pela Arduino IDE que envia sinal lógico 0 para o pino OUT, retornando um número inteiro que é utilizado para calibragem do resultado onde é definido a percepção de uma das 3 cores características de um semáforo.

Para reconhecer a cor vermelha utilizada no protótipo de semáforo foram obtidos valores de 7 a 183 para frequência do fotodiodo vermelho e de 124 a 258 para frequência do fotodiodo verde. Para a cor verde foram mapeados valores de 60 a 164 para frequência do fotodiodo vermelho, 4 a 116 para frequência do fotodiodo verde e de 31 a 110 para frequência do fotodiodo azul. Por fim para o reconhecimento da cor amarela foram mapeados valores de 5 a 43 para frequência do fotodiodo vermelho, 7 a 65 para frequência do fotodiodo verde e de 21 a 133 para frequência do fotodiodo azul. Os valores mapeados foram balizados com testes do sensor exposto a iluminação exterior (luz do local de testes) acesa e apagada para garantir uma leitura mais apurada.

Após realizar uma leitura de cor, é enviado uma mensagem ao servidor de aplicação através do objeto de manipulação de websocket, com um *payload* composto por um atributo identificador de qual semáforo está sendo representado e atributo que representa a cor lida, após o envio é aguardado 100 milissegundos e reiniciado processo.

2.3 Servidor de aplicação

A implementação do servidor de aplicação se deu através da linguagem *open-source Typescript*(*TS*) na versão 4.2, codificação que foi transpilada no *build* de produção para *Javascript* versão *ES6*, sendo desenvolvido com o auxílio do editor de códigos *Visual Studio Code* na versão 1.49. O ambiente de execução do projeto foi o Nodejs na versão 14.16.1. O banco de dados escolhido para armazenamento das informações referentes aos usuários e aos semáforos foi o *PostgresSQL*, que foi consumido como um serviço disponibilizado pela empresa *ElephantSQL* de forma gratuita, sendo a instância localizada na região leste dos Estados Unidos da América, dispondo de 20 MB (*Megabytes*) como limite de armazenamento.

Foram utilizadas 22 bibliotecas como dependência de execução e 17 bibliotecas de definições de tipo para *Typescript* e desenvolvimento. As dependências de execução foram necessárias para abstrair e terceirizar a codificação dos seguintes tópicos: consumo de *API* do Google Maps para *Javascript*, gerenciador de processos de aplicações Nodejs para facilitar clusterização, gerenciamento de *logs*, *live reloading*, definições de consumo de memória entre outras funcionalidades, biblioteca para processamento de operações em fila de mensageria, melhorias para *logging*, biblioteca para criação de servidor *HTTP* e dependências similares, biblioteca template engine para montagem de e-mail, bibliotecas clientes de banco de dados (neste caso apenas para Postgres), biblioteca para facilitar o uso de recursos como anotações e injeção de dependências. Todas as bibliotecas supracitadas foram também instaladas com suas definições de tipos para facilitar a codificação com as mesmas pelo *Typescript*. As dependências de desenvolvimento foram utilizadas para execução de código *Typescript* sem transpilação em ambiente local e para execução de testes unitários.

A aplicação em ambiente local é iniciada na porta 5000 pelo *Nodejs* utilizando a biblioteca Express e também é iniciado a espera de conexões do tipo websocket com o auxílio da biblioteca *ws*. Foi utilizado a biblioteca dotenv para facilitar a inicialização de variáveis de ambiente utilizadas em toda a aplicação, como por exemplo chaves de acesso de APIs de terceiros, credenciais de servidor de e-mail, dados da conexão com o host do banco de dados e informações relacionadas a configuração de tokens de autenticação. Foi implementado autenticação nos endpoints relacionados a manutenção

de informações de semáforo (métodos *POST*, *PUT* e *DELETE*) e endpoints relacionados a todas as operações de usuários exceto criação (método *POST*), sendo utilizado a estratégia de segurança *Bearer Authentication*, consistindo no envio de token no padrão RFC 7519 contendo o segredo de montagem e o e-mail do usuário, ficando apenas disponíveis publicamente os endpoints de criação de usuários e de consulta de semáforos para habilitar o acesso de forma anônima ao conjunto desenvolvido.

Foram implementadas funcionalidades referente às operações básicas de *CRUD* de usuários e semáforos, sendo salvo respectivamente nome, e-mail e senha do primeiro e identificador, nome, apelido, coordenada de localização e último status sobre segundo. Foi implementado também uma funcionalidade que retorna o semáforo mais próximo a localização informada, sendo de grande utilidade para o módulo móvel identificar alvos de monitoramento. A atualização do status do semáforo é feita por meio de uma *layer* de serviço utilizada tanto por endpoint do servidor *HTTP* e também pelo servidor do *websocket*, a qual é feita através do informe do identificador (que é definido de forma estática no módulo sensor de cores de semáforo) e um *enum* composto pelos elementos *RED*, *GREEN*, *YELLOW* para representar as cores e definir como status *SAFE* (seguro) quando o valor do *enum* for *RED* ou como *WAIT* (espere) se forem as opções restantes, sendo que no final do processo é realizado um *broadcast* pelo servidor do *websocket*, enviando a todos os clientes conectados um objeto contendo os dados do semáforo e outro contendo o status anterior e o novo status e a data da atualização.

O servidor de aplicação atingiu um percentual de 76,85% de cobertura de testes unitários, sendo a estatística monitorada gratuitamente por meio de uma aplicação chamada *Codecov.io*, onde foi integrada com o uso da plataforma *CircleCI* também com plano gratuito, a fim de realizar a integração e *delivery* contínuo da aplicação por meio da execução de *pipelines* que executam os testes unitários, que se executarem com sucesso, enviam métricas e permitem com que o código seja integrado a *branch* principal da aplicação hospedada no *Github*, que foi configurada para habilitar *pushes* somente com compilações bem sucedidas pelo *CircleCI* e uma taxa de cobertura superior a 70% verificado no *Codecov.io*. Após a tramitação ocorrer com sucesso e o *push* ser integrado a *branch* principal, é ativado um *webhook* da *PaaS Heroku*, que foi obtido por meio de plano gratuito de 550 horas por mês, e iniciado processo de *deploy* da aplicação, que após finalizar fica hospedada em ambiente *Cloud* para ser utilizada pelo aplicativo móvel e pelo sensor de cores de semáforo.

2.4 Desenvolvimento do aplicativo móvel

O aplicativo móvel foi desenvolvido na linguagem *Typescript* na versão 4.0.0 juntamente do framework *React Native* na *sdk* 40.0.1, sendo codificado com o auxílio do editor de código *Visual Studio Code* na versão 1.49. Por conta da utilização de um framework capaz de compilar código do tipo *Cross Platforms*, foi possível executar a aplicação em smartphones com sistema Android e iOS, que em fase de desenvolvimento se deu através do aplicativo *Expo*.

Foram utilizadas 7 bibliotecas de dependência de execução e 4 de definições de tipo para *Typescript*, sendo que as que não eram dependências do framework *React Native* e *Expo* (versão 40.0.0) ficaram responsáveis por abstrair e terceirizar os seguintes tópicos: Gerenciamento e persistência de estados previsíveis conteinerizados (*redux* na

versão 4.0.5 e *redux-persist* na versão 6.0.0), cliente *HTTP* (*axios* na versão 0.21.0), client de *websocket* (*ws* na versão 7.4.4). Foram adicionadas duas dependências extras junto com a biblioteca *Expo*, no intuito de facilitar tratativas utilizando localização do dispositivo (*expo-location* na versão 10.0.0) e emissão de áudios pela aplicação (*expo-speech* na versão 8.5.0).

No desenvolvimento do aplicativo foi criada uma tela inicial onde é executado um áudio onde é explicado sobre a funcionalidade do aplicativo e sobre os elementos presentes na tela. Foi criado também na sequência da navegação, uma tela de *login* opcional, onde é explicado via áudio a possibilidade de realizar um breve cadastro ou pular a etapa e utilizar diretamente como convidado, sendo essa funcionalidade útil para contato futuro com o usuário. A tela principal inicia um áudio explicando que a partir daquele momento é iniciado o monitoramento de semáforos próximos e que em sua presença serão emitidos alertas, que são enviados pelo funcionamento de dois componentes: gerenciador de semáforos próximos e painel de alertas. O gerenciador de semáforos próximos realiza a requisição da funcionalidade do servidor de aplicação de buscar o semáforo com menor distância da localização capturada pelo dispositivo com o auxílio da biblioteca *expo-location*, onde inicia o fluxo de processos do componente gerenciador de alertas por meio de uma estado compartilhado caso a distância entre os dois pontos seja menor que um parâmetro configurado globalmente na aplicação (definido como 10 metros em fase de desenvolvimento).

O componente painel de alertas quando condicionado a iniciar seu fluxo de processos pelo gerenciador de semáforos próximos, estabelece a conexão com o servidor de websocket localizado no servidor de aplicação, codificado para atuar somente no recebimento de mensagens de broadcast contendo no seu *payload* o identificador do semáforo definido como mais próximo. Cada mensagem recebida faz com que o painel receba um *refresh*, o qual verifica se atributo status do *payload* é igual a *SAFE* para realizar um alerta vibratório e sonoro com a mensagem "*Travessia segura"*, ou igual a *WAIT*, para realizar um alerta vibratório e disparar a mensagem sonora "Aguarde para atravessar". A tela principal permite a exibição dos componentes principais também em formato estendido, sendo que ambas áreas são tocáveis e é disparado uma mensagem sonora sobre a informação clicada tanto para o status do semáforo quanto a localização e distância até o semáforo mais próximo (salva de forma fictícia por meio de coordenadas coletadas no *Google Maps*), conforme ilustrado na Figura 5.

Referente a questões de usabilidade, foram aplicadas as estratégias de explanação via mensagens sonoras e também configurado em componentes onde não são compostos de texto, o mapeamento de títulos utilizados pelas ferramentas *Google Talkback* e *Apple VoiceOver* através das propriedades contidas nos elementos do tipo *JSX*. Foi utilizado a propriedade *accessible*, a qual atua representando um elemento que agrupa componentes filhos em um único elemento acessível rotulado através do texto contido na propriedade accessibilityLabel, a qual fora utilizada para explicar uma imagem na tela inicial e um botão que possui um ícone dentro. A navegação e interação com o aplicativo por meio de uma das ferramentas de acessibilidade se dá utilizando os gestos configurados nas próprias, tendo a aplicação apenas a responsabilidade pela simplificação do fluxo de utilização, a qual pode ser feito com o acesso de apenas 2

botões quando recém instalado, sendo eles o botão de continuar da tela inicial e o botão de pular a etapa cadastral. Após a primeira escolha da tela de *login*, é persistido pelo *store* de usuário do *redux* a escolha de continuar como convidado e caso encerrado o aplicativo é iniciado novamente é aberta diretamente a tela principal para simplificar a navegação.

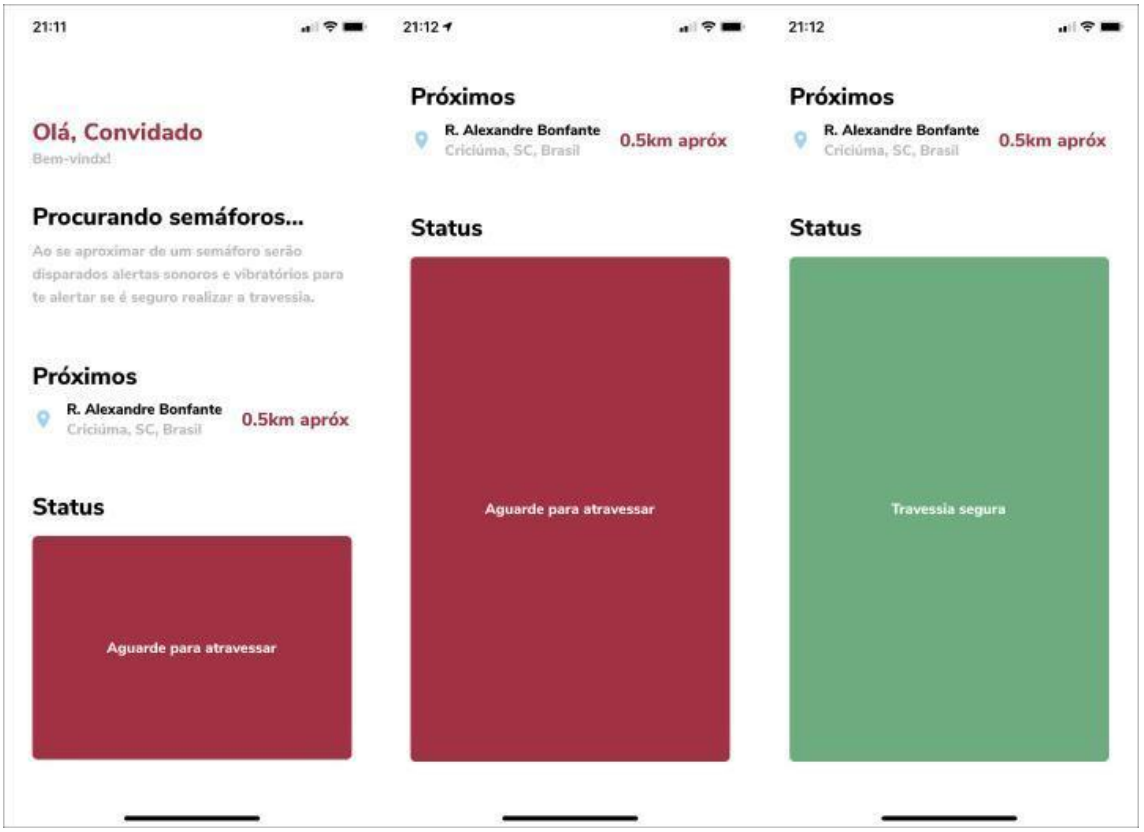

Figura. 5 Componentes gerenciador de localização e painel de alertas

No dispositivo iOS (*iPhone 11*) onde foi realizado grande parte dos testes durante a implementação, foi verificado a necessidade de realizar a ativação da chave de Toque/Silencioso para a audição dos alertas sonoros executados pelo aplicativo.

3 Resultados e discussões

A integração entre os módulos da aplicação se mostrou funcional, sendo eficaz para alertar o usuário sobre o status atual do semáforo prototipado contendo apenas uma lâmpada. A *PaaS* utilizada atendeu as necessidades do projeto, sendo verificado aspectos positivos no quesito disponibilidade, performance do container e facilidade de *deploy*, sendo o tempo de resposta das requisições contemplado dentro de um intervalo pequeno de tempo (em média de 500 milisegundos a no máximo 2 segundos), porém com algumas limitações no quesito de monitoramento da aplicação, pois no plano gratuito não é disponibilizado o painel de métricas, que tornaria mais difícil a análise de custos para migração para plano com maior desempenho. Outra limitação encontrada na *PaaS* foi o tempo de inicialização do *dyno* (como é chamado o container da aplicação no *Heroku*), a qual levou cerca de 5 a 10 segundos nos testes realizados durante a

implementação do módulo servidor de aplicação, sendo essa também um ônus do plano gratuito conforme verificado na documentação da plataforma, ocorrendo a cada 30 minutos de ociosidade e que dificultaria a utilização do conjunto em testes com cenários mais realistas por conta da demora da realização da primeira requisição, a qual caso o usuário já esteja na tela principal por conta de autenticação prévia, pode ser relevante como a busca de semáforos próximos. A instância do banco de dados *PostgresSQL* fornecido como serviço gratuitamente pela *ElephantSQL* atendeu as expectativas durante o desenvolvimento provendo uma conexão estável, sendo apenas o tamanho liberado para armazenamento uma limitação para ambientes de produção.

O módulo sensor de cores do semáforo foi implementado visando reconhecer o status de semáforos com apenas uma lâmpada, sendo necessário mais sensores de cor *TCS3200* e ajustes na biblioteca criada para o *handle* da leitura de cor para cenários com 3 lâmpadas. Foi necessário realizar um mapeamento de valores (Tabela 2) para cada uma das cores reconhecidas e apesar de contar com 4 LEDs embutidos para clarear a superfície de sensoriamento, foi notado durante o mapeamento uma maior estabilização da frequência lida em ambiente com pouca iluminação, contudo, no decorrer do mapeamento foi contornado via código e a diferença deixou de ser demonstrada. Outro fator importante notado durante o mapeamento das frequências foi a espaço entre o sensor e o objeto a ser captado a cor, sendo que distâncias acima de 4 centímetros retornavam valores menos precisos e a 6 cm não foi mais possível aferir as cores emitidas pelo anel de LED presente no semáforo prototipado. Por fim, a latência da leitura em uma distância menor que 4 centímetros foi imperceptível, sendo estipulado um valor via código de delay para obter aferições em um ritmo fixado em 100 milissegundos.

Tabela. 2 Mapeamento dos resultados captados para reconhecimento das cores

Cor mapeada	R	B	G
Vermelho	> 7 && < 183	-	>124 && < 255
Verde	> 60 && < 164	> 31 && < 110	> 4 && < 116
Amarelo	> 5 && < 43	> 21 && < 133	> 7 && < 65

Fonte: Dados da pesquisa, 2021.

Foi verificado nos testes de implementação o bom funcionamento do conjunto de módulos, sendo que na distância menor de 4 centímetros entre o módulo sensor de cor e o protótipo de semáforo e com conexão Wi-Fi e 4G no smartphone, não foram demonstrados falhas nos módulos, um ponto positivo também por conta do desempenho satisfatório do servidor de aplicação, que realiza procedimentos simples e com *payloads* com tamanhos pequenos (em média 126 bytes para o *JSON* com maior conteúdo). Em um teste de implementação onde a rede estava com acesso somente a 3G foram verificadas algumas ocorrências do erro *Gateway Timeout* (código 504 do protocolo *HTTP*) e o fechamento de conexão com o *websocket* no sensor de cores do semáforo e no aplicativo móvel ao tentar requisitar o servidor de aplicação, o que poderia manter o status de travessia segura de forma incorreta expressando assim uma informação incorreta ao usuário. Um tratamento realizado para contornar essa situação de maneira

paliativa foi a captura de exceções pelo aplicativo móvel sinalizando ao usuário a ocorrência de problemas de conexão e indicando a redobrar a atenção na travessia.

No estudo realizado por Oliveira (2018), foi desenvolvido um protótipo de semáforo eletrônico em conjunto com um aplicativo móvel para dispositivos Android para assim como a presente pesquisa, alertar usuários sobre o status do semáforo a fim de garantir mais segurança na travessia, sendo composto por somente os 2 componentes. A presente pesquisa difere da de Oliveira (2018) no quesito de execução em ambos sistemas operacionais para dispositivo móvel (iOS e Android) e também no formato de captação de dados do semáforo, onde na referida foi coletado os dados gerados no *sketch* do microcontrolador e enviados a uma *API* exposta na rede por meio de uma biblioteca para *Javascript* chamada *Ngrok*, cuja função é de criar túneis entre portas de uma máquina local e servidores do mesmo. Os pontos onde a presente pesquisa pode se destacar perante a de Oliveira (2018) e que não estavam presentes nas melhorias futuras foram a nível de infraestrutura: aplicativo móvel *cross-platform* e *API* disponível na nuvem em tempo integral, a nível de domínio de aplicação: cenário de captação de dados mais realista por conta da coleta via sensor de cores (*IoT)* e localização mais apurada com uso de tecnologias de mercado (*Google Maps Javascript API* e *Expo Location)*, a nível de implementação: aplicação contendo fluxos de integração/*delivery* contínuos com o *CircleCI, Github, Heroku* e cobertura de testes unitários de 76,85%, sendo essas melhorias relevantes no quesito de boas práticas de desenvolvimento de software, utilizadas comumente em aplicações Cloud. Um dos pontos presentes como possíveis melhorias na pesquisa de Oliveira (2018) foi a precisão da localização do smartphone em relação ao semáforo, a qual nesta pesquisa foi implementado com o serviço *Maps Javascript API* com plano gratuito de 90 dias disponibilizado pela *Google* e demonstrou acertar no cálculo da distância entre a localização semáforo e a localização do smartphone obtida com o auxílio da biblioteca *Expo Location*.

A pesquisa de Elia et al. (2018), utilizou um microcontrolador para simulação de semáforo eletrônico que envia informações a um banco de dados na nuvem referentes ao status, coordenadas e situação de manutenção do mesmo, onde também foi implementado uma página web funcionando de forma local, onde um técnico poderia indicar a sinalização de uma manutenção em determinado semáforo dando um update no mesmo banco de dados onde o microcontrolador trabalha e desencadeando notificações de *push* para usuários cadastrados, a fim de notificar para escolha de trajetos alternativos. Diferentemente desta pesquisa, foi utilizado somente rede 3G para a conexão com a internet e o banco de dados, mas foi encontrada similaridade na comparação de desempenho informado, sendo que o funcionamento se provou adequado por conta da baixa complexidade e tamanho das mensagens transportadas entre os componentes da aplicação assim como na aplicação desta pesquisa.

O trabalho de Sarrab, Pulparambil e Awadalla (2020) apresentou similaridade com a presente pesquisa no quesito de propor um modelo de aplicação *IoT* para coletar, processar e armazenar dados de semáforos em tempo real, com o objetivo de verificar congestionamentos no trânsito e disparar mensagens aos cidadãos para ajudar na escolha de rotas em horários de pico. A conexão com a internet do protótipo de Sarrab, Pulparambil e Awadalla (2020) foi feita também com o módulo *ESP8266* mas diferentemente da presente pesquisa foi utilizado um serviço *Cloud* para a transmissão

dos dados a um servidor onde ocorria o armazenamento, sendo utilizado a plataforma *Thinger.io* que proveu análise dos dados, gerenciamento e visualização além de possibilitar comunicação em tempo real bidirecional onde no contexto da aplicação era possível controlar o tempo de duração dos status do semáforo, sendo feito a interação com auxílio de biblioteca que integra com a *Arduino IDE*. Ambas pesquisas utilizaram *Wi-Fi* para acesso a Internet, e não foi considerado questões energéticas ou de carregamento sendo que foi realizado alimentação dos módulos por meio de conexão USB com *notebook*.Não houveram relatos de ocorrências de erros de perda de pacotes ou de tempo de resposta do servidor como fora verificado na conexão 3G da presente pesquisa.

O artigo publicado por Danielsson, Postema e Munir (2021) também aplicou o uso da *PaaS Heroku*, voltado para desenvolvimento de uma plataforma inovadora de *deployment* de aplicações *web* de produtos em desenvolvimento para a empresa *Axis Communications*.O protótipo desenvolvido pela pesquisa se tratou de um *website* de gerenciamento interno de uma mercearia, sendo atingido os objetivos de facilitar o *deploy* da aplicação prototipada, reduzir custos, diminuir o tempo de desenvolvimento, e reduzir a complexibilidade do processo. Como verificado durante o desenvolvimento da presente pesquisa, também foi visto que a plataforma *Heroku* demonstrou atingir as expectativas oferecendo segurança, escalabilidade e simplicidade nos processos, sendo o autogerenciado o número de *dynos* conforme o uso podendo escalar para cima ou para baixo. O trabalho de Danielsson, Postema e Munir (2021) também envolveu uma pesquisa com seis profissionais especializados em *Cloud Computing*, sendo todos intitulados no artigo como Engenheiros de Software de nível pleno e sênior e cerca de 4 dos participantes apontaram o *Heroku* e seus *add-ons* como soluções praticadas que seriam utilizadas pelos mesmos no futuro.

A pesquisa de Leporini, Buzzi e Buzzi (2012) tem como propósito analisar a interação do uso da aplicação nativa *iOS VoiceOver* por portadores de deficiência visual com foco na funcionalidade de gestos, presente nos dispositivos *iPhone e iPad*. Na presente pesquisa foi verificado que a utilização do *iOS VoiceOver* assim como o *Google Talkback*, pode facilitar a interação por meio de gestos e sons tornando as aplicações acessadas com seu uso paralelo mais acessíveis, porém conforme a pesquisa de Leporini, Buzzi e Buzzi (2012), também foi verificado nos testes de implementação uma dificuldade na utilização de campos de texto, tornando o preenchimento de formulários uma tarefa que pode consumir uma maior quantidade de tempo e adaptação. Outro ponto levantado na referida pesquisa, foi a dificuldade de identificação da navegação entre os elementos e a falta de clareza dos detalhes de elementos presentes, sendo na presente pesquisa realizada uma implementação que pode atenuar estes pontos: a explicação breve da tela e sua funcionalidade no momento da inicialização da mesma.

4. Conclusão

Este trabalho aplicou conceitos de Internet das Coisas em uma aplicação móvel com semáforo eletrônico para auxiliar a travessia de deficientes visuais. Os alertas sonoros e vibratórios gerados em tempo real de acordo com a localização e status do semáforo pode resultar em uma travessia de rua com maior segurança e independência, podendo

também reduzir o risco de acidentes. O conjunto de protótipos implementados demonstraram funcionar de acordo com as expectativas, integrando os componentes por meio de conexão *Wi-Fi* ou 4G apresentando baixa complexidade nas rotinas computacionais desenvolvidas. De acordo com os resultados foi verificada eficácia no alertamento via aplicativo móvel, assim como na integração dos dados pelo módulo sensor de cores de semáforo e *API*, mas também, foram verificados problemas relacionados a instabilidade no acesso via rede 3G e lentidão em requisições *HTTP* após trinta minutos de ociosidade do servidor por conta de tempo de inicialização de container do plano gratuito do *Heroku*.

O protótipo de semáforo demonstrou ser capaz de gerar dados para simulação corretamente e o sensor de cores do semáforo pode ler com precisão as cores em uma distância menor que quatro centímetros. Porém, o dispositivo não está preparado para ambientes externos por conta de não ser acoplado fisicamente no semáforo, onde o contato com vento poderia deslocar o sensor a uma distância maior que quatro centímetros e interferir na coleta dos dados.

Apesar dos resultados obtidos serem positivos em alertar o usuário para auxílio na travessia de ruas, seria necessário disponibilizar o servidor de aplicação em um plano de *PaaS* sem tempo de inicialização por conta de ociosidade e verificar maneiras de acoplagem do sensor de cores no semáforo para garantir a precisão da leitura dos dados.

Baseado nos conhecimentos adquiridos e também nos resultados obtidos, esta pesquisa propõe para trabalhos futuros a validação se o conjunto desenvolvido atende de maneira funcional o público alvo, a utilização outras *PaaS* que não possuam limitações de planos para facilitar a implementação de testes; desenvolvimento de protótipo de semáforos com três lâmpadas de LED para abranger mais cenários juntamente com sensor de cores que seja capaz de coletar seus dados; utilização do protocolo *MQTT* para integração entre os dados coletados do sensor de cores com o servidor de aplicação.

Referências

Danielsson, P., Postema, T., & Munir, H. (2021). Heroku-Based Innovative Platform for Web-Based Deployment in Product Development at Axis. IEEE Access, 9, 10805–10819. https://doi.org/10.1109/ACCESS.2021.3050255

Elias, A. H., Kalid, K. S., Ahmad, W. F. W., Mahamad, S., & Sarlan, A. (2018). TrafficSys: Traffic Light Fault Notification Mobile Application. Advanced Science Letters, 24(2), 1493–1496. https://doi.org/10.1166/asl.2018.10777

Galegale, G. P., Siqueira, E., Souza, C. A. de, & Silva, C. B. H. (2016). Internet das Coisas aplicada a negócios: Um estudo bibliométrico. Journal of Information Systems and Technology Management, 13(3). https://doi.org/10.4301/S1807-17752016000300004

Ricardo Kerschbaumer. (2018). Microcontroladores (p. 181). Secretaria de Educação Profissional e Tecnológica Instituto Federal de Educação, Ciência e Tecnologia Catarinense. https://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/wp-content/uploads/sites/43/2018/02/Apostila-Microcontroladores.pdf

Leporini, B., Buzzi, M. C., & Buzzi, M. (2012). Interacting with mobile devices via VoiceOver. Proceedings of the 24th Australian Computer-Human Interaction Conference on - OzCHI '12, 339–348. https://doi.org/10.1145/2414536.2414591

Nunes, S. da S., & Lomônaco, J. F. B. (2008). Desenvolvimento de conceitos em cegos congênitos: caminhos de aquisição do conhecimento. Psicologia Escolar e Educacional, 12(1), 119–138. https://doi.org/10.1590/S1413-85572008000100009

Oliveira, M. de F. (2018). PROTOTIPAÇÃO DE APLICATIVO MÓVEL PARA ASSISTÊNCIA AO DEFICIENTE VISUAL COM SEMÁFORO AUTOMATIZADO. Universidade do Extremo Sul Catarinense.

Perakovic, D., Perisa, M., Cvitic, I., & Brletic, L. (2018). Internet of things concept for informing visually impaired persons in indoor environments. Proceedings of the 2nd EAI International Conference on Management of Manufacturing Systems. https://doi.org/10.4108/eai.22-11-2017.2274670

Ramadhan, A. (2018). Wearable Smart System for Visually Impaired People. Sensors, 18(3), 843. https://doi.org/10.3390/s18030843

Santos, D. de S., Silva, L. V., & Oliveira, L. C. (2018). Um Protótipo de Sistema para Auxílio a Deficientes Visuais em Semáforos de Trânsito. 2018, 7.

Sarrab, M., Pulparambil, S., & Awadalla, M. (2020). Development of an IoT based real-time traffic monitoring system for city governance. Global Transitions, 2, 230–245. https://doi.org/10.1016/j.glt.2020.09.004

Bersh, R., & Sartoretto, M. L. (2020). O que é Tecnologia Assistiva. https://www.assistiva.com.br/tassistiva.html

Siddesh, G. M., Manjunath, S., & Srinivasa, K. G. (2016). Application for assisting mobility for the visually impaired using IoT infrastructure. 2016 International Conference on Computing, Communication and Automation (ICCCA), 1244–1249. https://doi.org/10.1109/CCAA.2016.7813907

Fornaziero, S., & Zulian, M. (2010). ESTUDO DAS DIFICULDADES ENCONTRADAS PELAS PESSOAS COM DEFICIÊNCIA VISUAL NO USO DO TRANSPORTE COLETIVO. XIV Encontro Latino Americano de Iniciação Científica e X Encontro Latino Americano de Pós-Graduação, 4. http://www.inicepg.univap.br/cd/INIC_2010/anais/arquivos/RE_0308_0861_01.pdf

INC, T. A. O. S. (2009). TCS3200, TCS3210 PROGRAMMABLE COLOR LIGHT-TO-FREQUENCY CONVERTER (Patent No. 1). https://www.mouser.com/catalog/specsheets/tcs3200-e11.pdf