UNIVERSIDADE ESTADUAL DO MARANHÃO – UEMA
CENTRO DE CIÊNCIAS TECNOLÓGICAS
CURSO DE ENGENHARIA DE COMPUTAÇÃO

CLAUDIO DIEGO SOUSA RAMOS

INTERFACE TÁTIL PARA AUXÍLIO NO DESLOCAMENTO DE PESSOAS COM DEFICIÊNCIA VISUAL

São Luís, MA
2016

CLAUDIO DIEGO SOUSA RAMOS

INTERFACE TÁTIL PARA AUXÍLIO NO DESLOCAMENTO DE PESSOAS COM DEFICIÊNCIA VISUAL

Monografia apresentada ao Curso de Engenharia de Computação da Universidade Estadual do Maranhão, como registro para obtenção parcial do grau de Bacharelado em Engenharia de Computação.

Orientador: MSc. Reinaldo de Jesus da Silva

Coorientador: MSc. Mario Renato Nunes Crispim da Silva Chaves Lima

São Luís, MA

2016

Ramos, Cláudio Diego Sousa.
Interface tátil para auxílio no deslocamento de pessoas com deficiência visual / Cláudio Diego Sousa Ramos.–São Luís, 2016.

77 f

Monografia (Graduação) – Curso de Engenharia de Computação, Universidade Estadual do Maranhão, 2016.

Orientador: Prof. Msc. Reinaldo de Jesus da Silva.
Co-orientador: Prof. Msc.Mário Renato Nunes C.da Silva Chaves Lima.

1.Interface tátil. 2.Deficiente visual. 3.Locomoção. I.Título

CDU: 004.5-056.262

CLAUDIO DIEGO SOUSA RAMOS

INTERFACE TÁTIL PARA AUXÍLIO NO DESLOCAMENTO DE PESSOAS COM DEFICIÊNCIA VISUAL

Monografia apresentada ao Curso de Engenharia de Computação da Universidade Estadual do Maranhão, como registro para obtenção parcial do grau de Bacharelado em Engenharia de Computação.

Trabalho aprovado. São Luís, MA, 12 de janeiro de 2015:

MSc. Reinaldo de Jesus da Silva
Orientador

Msc. Mario Renato Nunes Crispim da Silva Chaves Lima
Primeiro membro

Msc. Diógenes Carvalho Aquino
Segundo membro

Dedicatória

Ao Deus único e à minha família pelo amor e compreensão incondicionais.

Agradecimentos

Tenho muito a agradecer, pois há muitos sem os quais não teria chegado aonde cheguei.

Agradeço primeiramente a Deus por seu infinito amor que permitiu que eu desse mais esse passo na jornada da minha existência.

À minha esposa e filho por terem me recebido com amor, sorrisos e carinho todos esses dias, e me fortaleceram simplesmente pelo fato de existirem e persistirem em meu coração e mente, sendo uma família melhor do que eu mereço.

Aos meus pais e irmão pela paciência, compreensão e disponibilidade, me ajudando de forma que nunca conseguirei retribuir.

Aos meus amigos da igreja, em especial os integrantes projeto Radicais Livres, que entenderam ausências, atrasos, mal humor e outras falhas sem sequer reclamar.

Aos professores Reinaldo Silva e Mario Lima por compartilharem não apenas conhecimentos, mas também experiência, por serem não apenas mestres, mas também companheiros, animando nos momentos de desânimo e cobrando nos momentos necessários.

Aos colegas de curso que lutaram essa batalha comigo, enfrentaram o desconhecido sem pestanejar, o injusto sem fadigar e o impossível sem desacreditar.

Agradeço também à direção, meus professores e aos funcionários do curso de Engenharia de Computação da UEMA pela sua dedicação e zelo por essa honrosa instituição, quiçá pelo compromisso com a educação e crescimento do nosso Estado e do país, verificado na rotina diária.

"E o que importa não é o que você tem na vida,
mas quem você tem na vida.
E que bons amigos são
a família que nos permitiram escolher."
(William Shakespeare)

Resumo

É de grande relevância social desenvolver trabalhos que visam tornar o deficiente visual mais independente, principalmente em sua locomoção. Este trabalho tem como objetivo desenvolver uma interface tátil para o aplicativo “FollowVibe” com a função de auxílio no deslocamento em ambientes pré-mapeados, utilizando controlador arduino e atuadores do tipo motor de vibração. Utilizar uma solução tátil para deslocamento implica que o sentido da audição ficará mais disponível ao usuário e ao mesmo tempo não impedirá a utilização do tato. A solução tátil também permite uma interface mais precisa em menor tempo, dado que podem ser utilizados vários atuadores e que o mesmo tempo que leva o sinal que indica uma direção, leva o sinal de qualquer outra direção, alterando apenas os atuadores que deverão ser acionados. O usuário utilizará uma luva com os atuadores e que receberá comandos do aplicativo. Esses comandos são as direções que o usuário deve seguir. Para cada direção, uma combinação de atuadores é acionada na luva. Para cada movimentação do usuário, uma nova direção é mostrada até o usuário chegar ao destino desejado. Serão mostrados os testes realizados para verificar a eficiência do dispositivo.

Palavras-chaves: Interface. Tátil. Deficiente visual. Locomoção.

Abstract

It has a lot of social relevance, developing work aimed at making more independent visually impaired, especially in its locomotion. This work aims to develop an tactile interface for "FollowVibe"application with the aid of function in displacement pre-mapped environments, using Arduino controller and actuator-type motor vibracall. Using a tactile solution to displacement implies that sense of hearing will be more available to the user while not prevent use of touch. Tactile solution also enables a more accurate interface in a shorter time, since many actuators may be used and that the same time it takes signal that indicates a direction, takes signal from any other direction by changing only actuators to be driven. User will wear a glove with actuators and receive commands application. These commands are directions that user must follow. For each direction, a combination of actuators is triggered in glove. For each user movement, a new direction is setted until desired destination been reached. Tests will show performed to verify device's efficiency.

Key-words: Tatile Interface. Device to Deficients. Pre-mapping.

Lista de ilustrações

Lista de tabelas

Sumário

1 INTRODUÇÃO

Muitos brasileiros sofrem de deficiência visual. São cerca de trinta e cinco milhões de compatriotas que portam alguma limitação no aparelho visual de acordo com o censo de 2010 do IBGE (G1, 2012). Decorrentes da deficiência visual, outros problemas surgem, dentre os quais pode-se citar os associados a locomoção independente.

Os mais variados impedimentos podem ser listados no que tange empecilhos à locomoção do deficiente visual. Sejam as vias não adequadas às suas limitações, sejam as políticas públicas não inclusivas. Por fim, ainda pode-se verificar que o problema associado à localização geográfica também deve ser levantado.

Observa-se diversas opções que foram desenvolvidas no sentido de cooperar com a locomoção independente dos deficientes visuais. Podem-se citar os projetos ARGOS e VID. Ambos são tecnologias que permitem ao deficiente visual ser informado com relação a presença de obstáculos em seu caminho e com isso evitar incidentes. O primeiro resume-se a uma pulseira e o segundo um colete, os dois com atuadores. Em ambos os casos sensores identificam os obstáculos e informam ao usuário em tempo supostamente hábil(FILHO et al., 2011) (GLOBO, 2012).

Este trabalho propõe uma tecnologia do tipo vestível que, dados a posição atual e o destino desejado, informa qual direção o deficiente visual deve seguir.

O presente trabalho pretende descrever a criação de um sistema que ofereça funcionalidades de redes coorporativas, utilizando tecnologias *open source* e *hardware* de baixo custo.

No decorrer do trabalho, foi feito uma fundamentação teórica sobre os elementos cruciais para o desenvolvimento do protótipo, mais adiante serão mostradas as características e a implementação do protótipo do projeto e, por fim, foi avaliada a eficácia das técnicas utilizadas e mostrada a conclusão que foi possível chegar com o desenvolvimento do projeto.

1.1 Problema

Como garantir locomoção independente ao deficiente visual sem utilizar a audição do mesmo?

1.2 Hipótese

O desenvolvimento de um sistema móvel que determine uma rota dados dois pontos e forneça direcionamento por interface tátil.

1.3 Objetivos

1.3.1 Objetivo Geral

Promover interface tátil de deslocamento ao usuário do aplicativo “FollowVibe”.

1.3.2 Objetivos Específicos

- Descrever as principais necessidades dos deficientes visuais quanto ao deslocamento;
- Projetar dispositivo baseado nas limitações sensoriais dos deficientes visuais.
- Desenvolver dispositivo em conformidade com respectivo aplicativo.
- Testar o dispositivo aliado ao aplicativo.

1.4 Justificativa

O que motivou o desenvolvimento desse trabalho foi a falta de opções no mercado de dispositivos para deficientes visuais que contemplassem a mobilidade com foco no georeferenciamento e a utilização de interface tátil.

A maioria das soluções oferecidas pelo mercado são focadas ou na utilização da audição como interface ou na detecção de obstáculos.

2 REFERENCIAL TEÓRICO

2.1 Deficiência Visual

A visão é o sentido de maior relevância na percepção do mundo. Estudos recentes apontam que os seres humanos dão maior atenção aos estímulos visuais e que em caso de conflito de informação, a percepção geral é determinada pela visão (SHAMS; KAMITANI; SHIMOJO, 2000).

Muitas das manifestações da espécie humana são voltadas a esse sentido. Sejam elas manifestações artísticas (escultura, pintura), sejam sociais (escrita, símbolos de trânsito), o público-alvo majoritariamente deve ser alcançado pelo contato visual. Automaticamente, ficam excluídos de tais representações aqueles que não possuem visão ou que apresentam comprometimento significativo da função visual. Os impactos no cotidiano são muitos.

O termo deficiência visual não se restringe apenas às pessoas cegas. Nele são inseridas também as pessoas com baixa visão. O termo cegueira é utilizado quando existe uma perda total da visão até a ausência de projeção de luz, enquanto a baixa visão refere-se a um tipo de alteração na visão que pode ser causada por diversos fatores como baixa acuidade visual significativa, redução importante do campo visual, alterações corticais e/ou de sensibilidade aos contrastes que interferem ou limitam o desempenho visual do indivíduo (BRASIL, 2001).

Do ponto de vista legal, no Brasil, o decreto N ° 3.298, de 20 de dezembro de 1999 em seu artigo 4 °, inciso III define deficiência visual estabelecendo seus dois aspectos da seguinte forma: “deficiência visual – cegueira, na qual a acuidade visual é igual ou menor que 0,05 no melhor olho, com a melhor correção óptica; a baixa visão, que significa acuidade visual entre 0,3 e 0,05 no melhor olho, com a melhor correção óptica; os casos nos quais a somatória da medida do campo visual em ambos os olhos for igual ou menor que 60 °; ou a ocorrência simultânea de quaisquer das condições anteriores;” (BRASIL, 1999).

Segundo a Organization (2014), estima-se que no mundo haja 285 milhões de deficientes visuais, sendo 39 milhões de cegos e 246 milhões com baixa visão. Cerca de 90% desses possuem baixa renda e 82% dos cegos possuem cinquenta anos ou mais. Os erros de refração não tratados são ainda a principal causa de deficiência visual moderada e grave. A quantidade de pessoas com deficiência visual com causa em doenças infecciosas caiu nos últimos 20 anos, de acordo com estimativas. Entretanto, 80% de todas as deficiências visuais poderiam ser prevenidas ou curadas.

O noticiário online G1 (2012) citou o censo do IBGE de 2010 que estimou que no Brasil, mais de 6,5 milhões de habitantes possuem algum tipo de deficiência visual, dos quais mais de 500 mil são cegos.

De acordo com a OMS, a função visual possui quatro categorias, conforme tabela 1.

“A deficiência visual, em qualquer grau, compromete a capacidade da pessoa de se

Tabela 1: Tabela de valores de classificação de função visual.

Categoria	Menor que	Igual ou maior que
Deficiência Visual mínima ou nenhuma deficiência visual		6/18, 3/10(0,3), 20/70
Deficiência visual moderada	6/18, 3/10(0,3), 20/70	6/60, 1/10(0,1), 20/200
Deficiência visual severa	6/60, 1/10(0,1), 20/200	3/60, 1/20(0,05), 20/400
Cegueira	3/60, 1/20(0,05), 20/400	1/60, 1/50(0,02), 5/300 (20/1200)

FONTE:(ORGANIZATION, 2014)

orientar e de se movimentar no espaço com segurança e independência" (GIL, 2000). Sá (2005) afirma que, de acordo com suas pesquisas, a bengala é a principal suporte utilizados pelas pessoas cegas ou com baixa visão para se locomoverem. Os deficientes visuais dependem ainda de outras pessoas para identificar ruas, endereços, itinerários de ônibus, avisos, obstáculos, entre outros. Ao transitarem pelas vias públicas as situações de risco a que são expostas são constantes.

A compreensão social, a educação por meios não-visuais e a mobilidade independente são os três grandes problemas que atingem o indivíduo cego (TELFORD; SAWREY, 1972). O foco deste trabalho está na mobilidade independente.

Os centros urbanos apresentam barreiras em ruas e vias públicas que dificultam o trânsito de pedestre em geral e para aqueles que possuem dificuldade de locomoção ou mobilidade reduzida tornam-se inacessíveis. Segundo Sá (2005):

> A disposição desordenada e caótica do mobiliário urbano ganha realce e visibilidade através das pessoas cegas que deparam com barreiras tais como: a) cabines telefônicas ou orelhões e lixeiras sem sinalização; b) veículos estacionados irregularmente em passeios públicos; c) obras sem proteção ou cordão de isolamento, cuja maleabilidade e altura não são detectadas pela bengala; d) esgoto e bueiros abertos, dejetos, buracos, sacos de lixo, entulhos, pisos quebrados; e) cartazes, placas publicitárias, mesas e cadeiras nas calçadas; f) falta de sinais sonoros nas ruas; g) toldos baixos avançados nas calçadas e outros obstáculos aéreos; h) vegetação agressiva, vasos, canteiros, jardineiras e árvores com ramos baixos sem proteção; i) camelôs, bancas de frutas, carrinhos de pipoca e de *hot-dog*; j) pavimentação irregular, calçadas com aclives e declives; k) portões abertos ou que se abrem automaticamente; l) barras de ferro, postes metálicos finos e de difícil localização pela bengala; m) falta de alinhamento na construção dos edifícios; n) excesso de ruído próprio dos centros urbanos; o) elemento surpresa como andaimes nas calçadas; p) falta de sinais de trânsito nas ruas e avenidas mais movimentadas; q) falta de faixas de segurança com sinaleira para travessia de pedestre; r) inexistência de calçamento, degraus nas calçadas; s) semáforos com pouca luz; t) todo tipo de barreira arquitetônica e ideológica.

A importância da mobilidade é verificada cotidianamente. Muitas das tarefas do dia a dia incluem mover-se, fazer pequenas viagens. Dirigir, ir ao ponto de ônibus coletivo, ir ao trabalho,

até mesmo deslocar-se para lazer incluem essas pequenas viagens. A capacidade de deslocar-se pequenas e médias distâncias e chegar ao local desejado é importante para independência pessoal (HERSH; JOHNSON, 2008).

Essas barreiras produzem um forte impacto na vida das pessoas com dificuldade de locomoção ou mobilidade reduzida, refletindo a falta de organização dos centros urbanos. Os ambientes restritivos, os espaços inacessíveis e as estruturas que os exclui, contribuem para uma relação distante entre o cidadão com deficiência e o meio que o cerca, onde o grande número de obstáculos reflete a ideia de um espaço projetado para o "homem-padrão", sem levar em consideração as limitações e diferenças entre os pedestres (Sá, 2005).

Diversas subtarefas estão incluídas em um pequeno deslocamento. Desviar-se de obstáculos na pavimentação, andar na direção certa, atravessar a via com segurança, chegar ao ponto de ônibus correto, reconhecer o ônibus correto são exemplos dessas subtarefas (HARPER; GREEN, 2002).

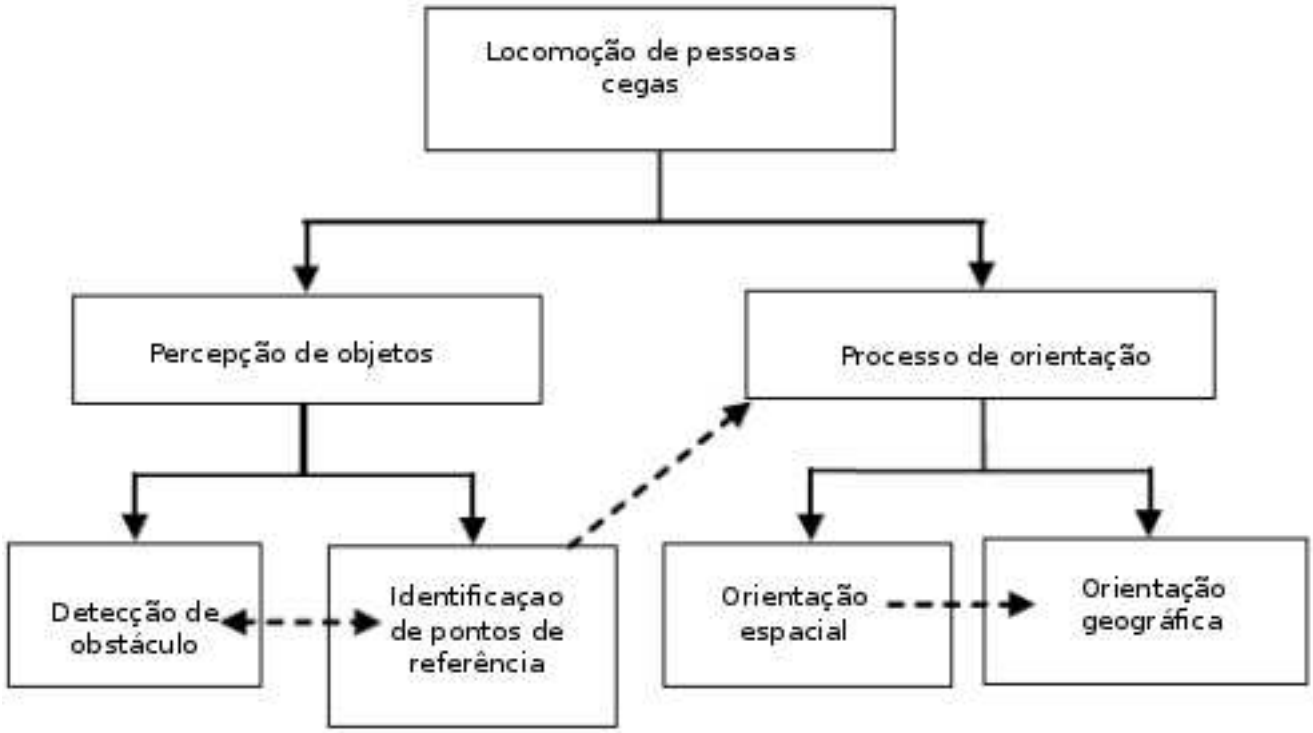
Essas subtarefas podem ser classificadas em duas categorias: mobilidade e acesso ao ambiente. Mobilidade aqui, ainda pode ser dividida em dois subpontos, a saber, desvio de obstáculos e orientação e navegação. Acesso ao ambiente pode, por sua vez, ser pensado a partir da minimização de perigos e de informação e sinais (HERSH; JOHNSON, 2008).

Figura 1 – Esquema de mobilidade

FONTE: (HERSH; JOHNSON, 2008)

Fazer um deslocamento no mundo real para o deficiente visual é ainda mais complexo que movimenta-se em um ambiente mapeado, conhecido. Brambring (1985) desenvolveu um modelo de deslocamento que é frequentemente citado, como mostrado na figura a seguir.

Figura 2 – Esquema de mobilidade de Brambring



FONTE: (HERSH; JOHNSON, 2008)

Para ele, a locomoção do deficiente visual compreende dois tipos de processos, um relacionado a percepção e o outro a orientação. Perceber refere-se a obtenção de informações do ambiente com os demais órgãos dos sentidos e orientar-se refere-se ao conhecimento de uma dada posição em relação à jornada.

Existem muitas coisas a serem realizadas para aumentar a qualidade de vida dos deficientes visuais, sendo papel da sociedade oferecer condições para que eles possam desenvolver suas capacidades físicas e mentais (GIL, 2000).

Interessa a este trabalho entender as tarefas de mobilidade a partir da orientação geográfica. Também chamada de navegação, diz respeito a determinar a posição do espaço geográfico de um percurso, como já fora adiantado.

2.2 Tecnologia Assistiva

Em um mundo globalizado onde as pessoas e ambientes estão em constantes mudanças, faz-se necessário viabilizar as formas de comunicação, locomoção, interação e participação social das pessoas que encontram-se impossibilitadas de realizar determinadas tarefas. “Para as pessoas sem deficiência, a tecnologia torna as coisas mais fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis” (RADABAUGH, 1993). Nesse cenário surgem os recursos

e serviços da tecnologia assistiva.

A tecnologia assistiva pode ser compreendida como um suporte que torna possível a ampliação de alguma habilidade que encontra-se limitada ou que desempenha algo inviável de ser realizado devido a circunstâncias como deficiência e envelhecimento. Dessa forma, é possível afirmar que a tecnologia assistiva busca possibilitar ao deficiente um elevado grau de independência e qualidade de vida, contribuindo ainda para a sua inclusão social, ao promover uma melhor comunicação, mobilidade e controle do ambiente em qual está inserido (BERSCH, 2008).

Para elaboração de um conceito brasileiro de tecnologia assistiva, foi instituído o Comitê de Ajudas Técnicas – CAT, pela Secretaria Especial de Direitos Humanos da Presidência da República através da Portaria nº 142 de 16 de novembro de 2006. Em 14 de dezembro de 2007 foi aprovado o conceito a seguir, elaborado pelo CAT:

> Tecnologia Assistiva é 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 (BRASIL, 2009).

"Os recursos de tecnologia assistiva são organizados ou classificados de acordo com objetivos funcionais a que se destinam" (BERSCH, 2008), no entanto, cada país estabelece a sua forma de apresentar essa classificação. No Brasil, para facilitar a compreensão dos recursos e serviços de tecnologia assistiva, Sartoretto e Bersch (2015) disponibilizaram, com objetivo apenas didático, a classificação em categorias que estão descritas a seguir:

1. Auxílios para a vida diária e vida prática: possibilitam um desenvolvimento com maior autonomia e independência em tarefas comuns, vividas diariamente, como por exemplo, alimentar-se, cozinhar, vestir-se, tomar banho e realizar necessidades pessoais.

2. Comunicação aumentativa e alternativa: oferece um suporte às pessoas que não falam ou não podem fazer uso da escrita funcional ou pessoas que possuem uma defasagem entre a necessidade de comunicar-se e sua fala e/ou escrita. As pranchas de comunicação, letras ou palavras escritas, são exemplos de recursos dessa categoria.

3. Recursos de acessibilidade ao computador: Hardware e software desenvolvidos estrategicamente de forma a tornar o computador acessível a pessoas com deficiência visual, auditiva, intelectual ou motora. São exemplos desses recursos: mouses, teclados e acionadores diferenciados; sons, imagens e informações táteis.

4. Sistemas de controle de ambiente: um controle remoto que possibilita às pessoas com limitações motoras ações como ligar, desligar e ajustar aparelhos eletroeletrônicos como a luz, o som, televisores, ventiladores, executar a abertura e fechamento de portas e janelas, receber e fazer chamadas telefônicas, acionar sistemas de segurança, entre outros.

5. Projetos arquitetônicos para acessibilidade: construções que garantem acesso, funcionalidade e mobilidade a todas as pessoas, independente de sua condição física e sensorial.

6. Órteses e próteses: Peças artificiais que substituem partes ausentes do corpo e peças colocadas junto a uma região do corpo, garantindo-lhe um melhor posicionamento, estabilização e/ou função. Auxiliam na escrita, digitação, utilização de talheres, manejo de objetos para higiene pessoal, correção postural, entre outros.

7. Adequação postural: recursos que buscam oferecer posturas alinhadas, estáveis, confortáveis e com boa distribuição do peso corporal. A sua utilização desde os primeiros momentos em que a necessidade se apresenta ajuda na prevenção de deformidades corporais.

8. Auxílios de mobilidade: são recursos como bengalas, muletas, andadores, carrinhos, cadeiras de rodas manuais ou elétricas, equipamento ou estratégia utilizada na melhoria da mobilidade pessoal.

9. Auxílios para qualificação da habilidade visual e recursos que ampliam a informação a pessoas com baixa visão ou cegas: são recursos como auxílios ópticos, lentes, lupas manuais e lupas eletrônicas, softwares ampliadores de tela, material gráfico com texturas e relevos, mapas e gráficos táteis, etc.

10. Auxílios para pessoas com surdez ou com deficit auditivo: são recursos como aparelhos para surdez, telefones com teclado-teletipo (TTY), sistemas com alerta táctil-visual, celular com mensagens escritas e chamadas por vibração, software que favorece a comunicação ao telefone celular transformando em voz o texto digitado no celular e em texto a mensagem falada, livros, textos e dicionários digitais em língua de sinais, etc.

11. Mobilidade em veículos: dão suporte para que uma pessoa com deficiência física possa dirigir um automóvel, facilitadores de embarque e desembarque como elevadores para cadeiras de rodas, rampas para cadeiras de rodas, serviços de autoescola para pessoas com deficiência, etc.

12. Esporte e lazer: recursos que auxiliam na prática de esportes e atividades de lazer, como por exemplo, cadeira de rodas/basquete, bola sonora, auxílio para segurar cartas e prótese para escalada no gelo (BERSCH, 2008).

É importante não confundir a tecnologia assistiva com outras tecnologias. As tecnologias utilizadas na área médica e na área de reabilitação, por exemplo, buscam facilitar e dar mais qualidade às atividades desenvolvidas pelos profissionais na avaliação e intervenção terapêutica. São considerados recursos utilizados pelos profissionais e não pelo “usuários”, caracterizando-se como tenologia médica ou de reabilitação (BERSCH, 2008).

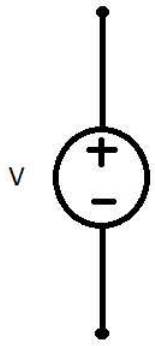
O debate em torno aos direitos da pessoa com deficiência é frequente. Busca-se o pleno exercício da cidadania, e acesso a outros direitos fundamentais como: aprender, comunicar-se, trabalhar, divertir-se, entre outros. As políticas públicas são voltadas para algumas áreas específicas da Tecnologia Assistiva, como concessão gratuita de próteses, por exemplo, mas os recursos precisam ainda ser destinados a outras categorias (FILHO, 2010). A exemplo das categorias da tecnologia assistiva que precisam de maior investimento e políticas públicas que as apoiem, podem-se citar os auxílios de mobilidade, categoria na qual se encaixa o projeto descrito neste trabalho.

2.3 Fundamentos de Resistência Elétrica

2.3.1 Tensão

Energia é a capacidade de realizar um trabalho. Se há troca de energia no deslocamento de uma carga, há também diferença de potencial. Essa diferença de potencial é também chamada de tensão e é medida entre dois pontos de um sistema qualquer (BOYLESTAD, 2004). A imagem a seguir figura uma representação típica de fonte de tensão em circuitos elétricos.

Figura 3 – Representação de Tensão



FONTE: DO AUTOR, 2015

Há ainda outra forma de definir a tensão. A tensão é energia por unidade de carga criada pela separação entre as cargas (NILSSON; RIEDEL, 2008). Ela pode ser medida, quantificada. Assim, a tensão, também chamada de potencial elétrico, em qualquer ponto de um campo elétrico com uma determinada energia potencial e uma carga de teste associada a este ponto é definida pela fórmula seguinte:

$$V = U/q \tag{2.1}$$

Onde V é a tensão, U é a energia potencial e q é a carga de teste.

2.3.2 Corrente

O afastamento entre cargas gera uma força elétrica que chamamos tensão. Entretanto, outro fenômeno ocorre e é justamente o movimento destas cargas. Este fluxo elétrico chamamos corrente.

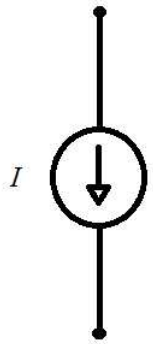
Outras considerações podem ser feitas quanto a corrente. O que chamamos de corrente é a variação temporal da carga necessária à movimentação de cargas ocasionando efeitos elétricos (NILSSON; RIEDEL, 2008). Por conta da última afirmação podemos expressar a corrente na seguinte forma diferencial:

$$I = dq/dt \tag{2.2}$$

Onde I é a corrente e dq/dt é a variação de carga no tempo.

Frise-se que a corrente sempre flui do ponto com maior potencial para o de menor potencial. A próxima imagem figura uma representação típica de corrente em circuitos elétricos.

Figura 4 – Representação de Corrente



FONTE: DO AUTOR, 2015

No mais, é importante frisar que não é qualquer fluxo que é denominado corrente. Para ser corrente deve haver mudança no valor da carga (YOUNG; FREEDMAN, 2009). Assim, onde for verificado equilíbrio eletrostático, não há corrente. Dentro do material os elétrons podem mover-se livremente sem, entretanto, escapar do corpo onde encontram-se.

2.3.3 Resistência Elétrica

Dentro de cada material, átomos e elétrons colidem entre si. Estas colisões provocam uma força de oposição ao fluxo da carga muito semelhante ao atrito mecânico, convertendo energia elétrica em alguma outra forma de energia, como a térmica. Chamamos essa oposição de resistência elétrica (BOYLESTAD, 2004).

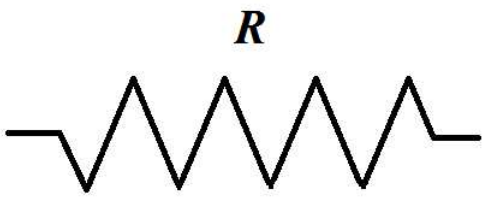
O valor da corrente pode ser relacionado com a diferença de potencial nas extremidades do condutor (YOUNG; FREEDMAN, 2009). A razão entre a tensão e a corrente, para um dado condutor, é o que chamamos de resistência e assinalamos na seguinte fórmula:

$$R = V/I \tag{2.3}$$

Onde R é a resistência, V é a tensão e I é a corrente.

Frise-se que a resistência, entre outras coisas, é relacionada com o tipo de material do meio. Some-se a isso o fato de que quanto maior a temperatura, maior a resistência, bem como o é proporcional também ao comprimento do meio. A próxima imagem é uma representação de uma resistência em um circuito elétrico.

Figura 5 – Representação de Resistência



FONTE: DO AUTOR, 2015

2.4 Arduino

Segundo a página oficial da fundação que mantém o projeto Arduino, Arduino é uma plataforma de prototipagem de código aberto baseado em *hardware* e *software* de uso facilitado (ARDUINO, 2015b). Isso significa que o Arduino foi projetado para viabilizar projetos de forma simplificada tornando protótipos mais fáceis de serem implementados.

Sem a utilização de uma plataforma do tipo Arduino, um projeto tende a levar um tempo maior na confecção de um protótipo. São mais fases, que vão desde a escolha do microcontrolador ou microprocessador a ser utilizado, até a manufatura da placa correspondente ao projeto, passando pela escolha de circuitos integrados mais adequados e outras decisões de projeto. Tudo isso, com o risco de após a confecção algum detalhe ter sido deixado de lado e haver necessidade de refazer, fora o possível prejuízo dos componentes. Banzi resume essa filosofia Arduino nas seguintes palavras:

> A filosofia do Arduino concentra-se em desenvolver projetos, e não em falar sobre eles. Ela representa uma busca constante por meios mais rápidos e poderosos de criarmos protótipos. Exploramos muitas técnicas, de prototipagem e desenvolvemos formas de pensar cada vez mais práticas (BANZI, 2012).

Plataformas do tipo Arduino são muito versáteis e há um grande ganho de tempo, resultados, fora a economia inerente. Uma vez que se opta pelo Arduino, é possível ter um protótipo mais rapidamente pronto. O mesmo Arduino pode ser utilizado em diversos projetos sejam concomitantes ou subsequentes e de forma muito rápida. São inúmeras possibilidades.

É muito comum a utilização desta plataforma justamente pela facilidade e escalabilidade da mesma. São inúmeras possibilidades de configuração e acoplamento de sensores e atuadores de forma a obter rapidamente o protótipo do projeto desenvolvido.

A comunidade de utilizadores de Arduino é composta por um público bastante variado como artistas, estudantes, programadores, amadores e profissionais de um modo geral e a sua utilização vai de objetos do dia a dia a instrumentos científicos bastante complexos.

2.4.1 História

O Arduino foi iniciado em 2005 na Itália, mais especificamente na cidade de Ivrea, no Interection Design Institute. Surgiu a partir da necessidade de estudantes de *design* de trabalharem com tecnologia, portanto precisava ser um meio fácil e como o público inicial era de estudantes, deveria ser também barato. O professor Massimo Banzi compartilhou essa problemática com o pesquisador visitante da Universidade de Malmö, David Cuartielles, na Suécia que também procurava uma solução semelhante. Nascia o Arduino (EVANS; NOBLE; HOCHENBAUM, 2013).

2.4.2 Arduino UNO

Arduino UNO refere-se a uma série de placas do da plataforma Arduino. Esta é uma placa que possui um microcontrolador ATmega328P, além de 14 pinos digitais de entrada e saída, dos quais 6 podem ser usados como saída PWM, 6 entradas analógicas, um cristal de quartzo de 16MHz, uma porta USB, uma entrada de força, um header ICSP, um botão de reset e demais especificações técnicas conforme próxima tabela.

Segundo a página oficial da comunidade Arduino, a placa UNO é a mais aconselhável para começar a projetos de eletrônica e codificação. É a mais robusta que se deve começar ainda segundo este site e é a que é mais usada e documentada de toda família Arduino (ARDUINO, 2015b).

Este modelo foi utilizado no lançamento da versão 1.0 do software Arduino (IDE), sendo ambas as versões de referência do Arduino.

Como tudo na plataforma Arduino, o UNO é também livre, a saber hardware livre, e portanto pode ser livremente construído por qualquer um. No Arduino, o microcontrolador já vem pré-programado com um bootloader. É possível não utilizar o bootloader e programar o ATmega328P através do header ICSP, mas isso não foi implementado neste projeto.

Para fins de alimentação, a placa UNO pode ser alimentada via USB ou através de fonte externa, como bateria 9V por exemplo. A placa opera em uma faixa que varia de 6 a 20V,

Figura 6 – Placa Arduino UNO

FONTE: (WIKIPEDIA, 2013)

entretanto se for utilizada tensão inferior a 7V a placa fica instável e se superior a 12V a mesma pode ser danificada. O ideal é manter a placa em uma faixa de operação entre 7V e 12V, portanto.

2.4.3 Motores de vibração Precision Microdrives 310-113 Pico Vibe^{TM} 10mm

Precision Microdrives 310-113 Pico Vibe^{TM} 10mm é um motor de vibração similar ao utilizado em celulares para alerta vibratório. São motores de vibração de tamanho reduzido e de preço acessível. As partes móveis deste possuem proteção dadas que estão encapsuladas.

Possuem um escopo de tensão que varia entre 2,5 e 3,8V, funcionando muito bem a 3V. Possuem massa de 1,2g e largura de 3,4mm. O diâmetro mede 10mm.

De funcionamento simples, basta a tensão de entrada para acionar os mesmos.

2.5 Programação

Segundo Dahl, Dijkstra e Hoare (1972), “a arte de programar consiste na arte de organizar e dominar a complexidade”. Isso exige do programador criatividade, dado que a tarefa de programação basicamente perpassa o desenvolvimento de um raciocínio lógico (FARRER et al., 1989). Além disso faz-se necessário escolher a linguagem que se vai efetuar a programação.

No caso deste trabalho, a escolha da linguagem foi baseada nas tecnologias que foram utilizadas. Isso implica que na programação Android foi utilizada a linguagem Java, dado que é a linguagem oficial do desenvolvimento desta tecnologia e recebe suporte amplo da comunidade de desenvolvedores e no caso de Arduino, a linguagem foi o Arduino Programming Language.

Tabela 2: Tabela de especificações técnicas do Arduino UNO.

Microcontrolador	ATmega328
Tensão Operacional	5V
Tensão de Entrada Recomendada	7-12V
Tensão de Entrada Extrema	6-20V
Pinos Digitais de E/S	14 (dos quais 6 proveêm saída PWM)
Pinos de Entrada Analógicos	6
Corrente DC por pino de E/S	40 mA
Corrente DC por pino de 3.3V	50 mA
Flash Memory	32 KB (ATmega328) dos quais 0.5 KB são usadas pelo *bootloader*
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidade do *Clock*	16 MHz
Comprimento	68,6mm
Largura	53,4mm
Peso	25g

FONTE:Adaptada de (ARDUINO, 2015a)

Figura 7 – Tabela de Referência do Motor de Vibração

Body Diameter:	10 mm [+/- 0.1]
Body Length:	3.4 mm [+/- 0.1]
Rated Operating Voltage:	3 V
Rated Vibration Speed:	12,200 rpm [+/- 2,500]
Typical Rated Operating Current:	60 mA
Typical Normalised Amplitude:	1.34 G

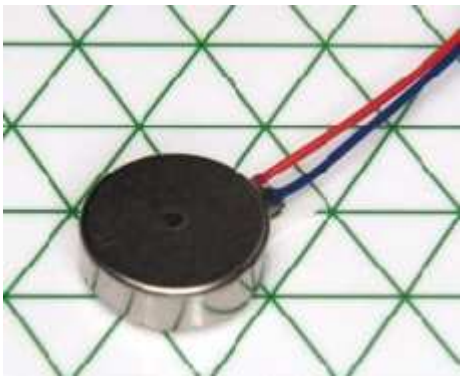
FONTE: (MICRODRIVES, 2016)

A este trabalho interessa entender o conceitos de programação utilizado na composição do mesmo. Assim, serão apresentadas definições de programação imperativa e da linguagem Wiring que é a utilizada na programação do Arduino.

2.5.1 Programação Imperativa

É o paradigma de programação que utiliza sentenças para alterar o estado de um determinado programa. Roberts (2015) afirma que a programação imperativa implica em dizer a máquina como fazer algo e esperar que como saída ocorra aquilo que você quer, acontecer.

Figura 8 – Motor de Vibração



FONTE: (MICRODRIVES, 2016)

Heidrich (2009) afirma que uma linguagem que utiliza paradigma imperativo tem seu foco na manipulação de variáveis enquanto que uma que utiliza orientação a objeto foca na manipulação de objetos que interagem entre si. Sebesta (2009) afirma que a orientação a objetos nada mais é do que a programação imperativa com mais recursos e que não é conveniente definir um paradigma exclusivo para esta última, portanto.

Heidrich (2009) afirma ainda que o paradigma imperativo possui como características flexibilidade, facilidade de migração de programas escritos em linguagens imperativas para outros paradigmas, relacionamento indireto com entrada/saída e difícil legibilidade dos programas.

2.5.2 Arduino Programming Language

É a linguagem de programação do Arduino e esta é baseada em wiring. A sintaxe é muito similar à da linguagem C e há muitas palavras reservadas e operadores coincidentes entre ambas.

Sucintamente, o programa possui duas funções. São elas as funções setup() e loop() e a IDE do Arduino já fornece um esqueleto com essas funções sempre que o usuário solicita nova *sketch*.

A função setup() é onde são configuradas as entradas e saídas da placa Arduino. Basicamente deve-se indicar a porta que será utilizada e determinar se será entrada ou saída. Além disso, nessa função pode-se inicializar variáveis, bibliotecas, etc. A função setup() é executada após a energização ou *reset* da placa e apenas essa única vez.

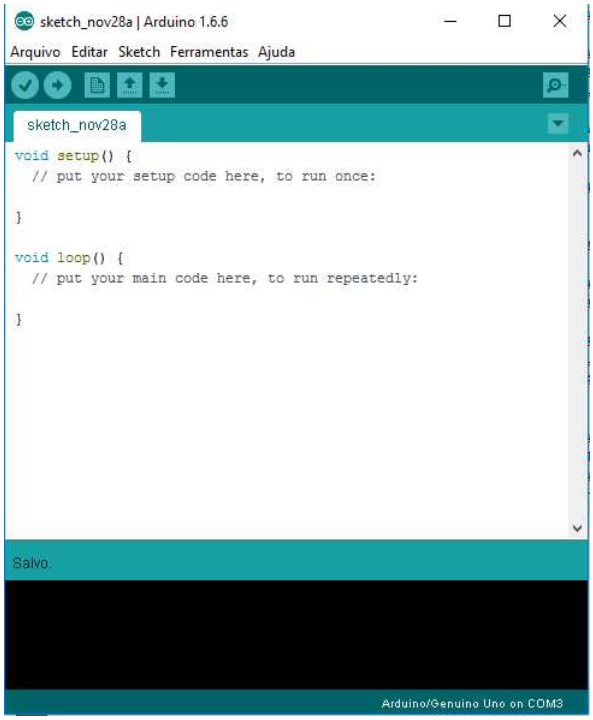
Já a função loop() é usada no controle ativo da placa. Ela é executada iteradamente e através dos controles de fluxos, condicionais e outras programações permite que o programa gravado tenha flexibilidade. De maneira mais simples, é nessa função que o programa muda e responde ao ambiente e às entradas que lhe forem dados.

No programa para Arduino podem ser declaradas variáveis dos mais variados tipos. Alguns tipos recebem variação do tipo *unsigned* como *char, int* e *long*. Recebe também declaração

de vetores, com o tipo *array*.

Toda essa programação é facilitada no ambiente que a comunidade Arduino oferece, o Arduino Integrated Development Environment. O Arduino Integrated Development Environment – ou Arduino Software (IDE) – é o ambiente onde os programas a serem carregados na placa Arduino são escritos e carregados na mesma.

Figura 9 – Tela inicial do Arduino IDE



FONTE: DO AUTOR, 2015

3 PROJETO DISPLAY LIGHT

Display Light é um projeto que visa melhorar a vida de deficientes visuais a partir da utilização de tecnologia e da Internet. A proposta, em linhas gerais, é promover inclusão e independência através de tecnologias acessíveis e de forma colaborativa.

3.1 Histórico

O projeto Display Light nasceu no primeiro semestre de 2014, denominado primeiramente de "Labrador Virtual".

Ainda sem forma definida, os integrantes ponderavam sobre um objetivo que mais provavelmente seria um aplicativo de locomoção para deficientes visuais dentro dos terminais de integração da cidade de São Luís, capital do Maranhão. Levantou-se dados quanto à população de deficientes visuais da ilha de São Luís, bem como a população do Estado, esboçou-se um levantamento de requisitos para o projeto e até mesmo um cronograma de atividades fora determinado.

Depois de diversas conversas, entretanto, percebeu-se que em vez de um aplicativo de celular que serviria localmente, poderia ser criado um conceito de desenvolvimento focado em acessibilidade móvel.

Assim, rebatizado de Display Light, o projeto superou a visão inicial de simples aplicação móvel.

3.2 Escopo do projeto Display Light

Com a ampliação do alcance do projeto, as metas precisaram ser redefinidas.

Não mais mera aplicação móvel, o Display Light visa fornecer uma interface de dispositivos móveis para deficientes visuais ao mesmo tempo que pretende fomentar o crescimento de uma comunidade entusiasta de inclusão e acessibilidade de portadores de deficiência visual às tecnologias móveis.

Diversas aplicações podem ser implementadas dentro do projeto. As mais abrangentes devem permitir ao deficiente visual exaurir os recursos de seu aparelho móvel e isso inclui interface totalmente dedicada a eles, aplicações rotineiras como editores de texto e planilhas eletrônicas e até mesmo jogos. Outras etapas incluem a criação de uma página virtual para compartilhamento de ideias e subprojetos relacionadas ao desenvolvimento inclusivo.

A ideia de uma aplicação de locomoção é uma etapa importante mas não sozinha deste grande projeto chamado Display Light. E fazê-la de maneira mais abrangente é torná-la mais útil e acessível. A esta etapa do projeto onde desenvolve-se uma aplicação para locomoção de deficientes visuais, batizou-se FollowVibe.

4 SISTEMA FOLLOWVIBE

O sistema FollowVibe é o primeiro passo do projeto supracitado.

FollowVibe é o sistema de locomoção para deficientes visuais do projeto Display Light. Busca total liberdade para deficiente. Neste primeiro incremento, o mesmo trabalhará focado em uma interface eficaz.

4.1 Método

Este trabalho utilizou método experimental, em consonância com a classificação feita por Prodanov e Freitas (2013). O dispositivo de interface aliado ao aplicativo desenvolvido visou provisão de mobilidade independente de deficiente visual ao mesmo tempo que o sentido da audição permanecia livre para qualquer outra atividade.

Foram etapas presentes neste projeto:

- Pesquisa e aplicação de literatura especializada;
- Levantamento das opções de tecnologias existentes;
- Projeto do dispostivo e do aplicativo;
- Desenvolvimento do dispositivo e do aplicativo;
- Testes do sistema desenvolvido.

4.2 Descrição do Sistema FollowVibe

FollowVibe é o sistema formado pelo aplicativo Android associado à luva com atuadores vibracalls, controlador arduino e as respectivas conexões. O sistema visa entregar ao usuário a indicação de rota a partir de dois pontos, sendo eles o atual e o destino, ao mesmo tempo que utiliza como interface dispositivo não visual, mais especificamente tátil.

O aplicativo manipula os dados de georreferenciamento. O usuário indica o destino desejado a partir da localização atual. O aplicativo, utilizando-se da tecnologia GPS, processa a rota a partir do ponto atual. Uma vez tendo calculado a rota, o aplicativo informa ao usuário a próxima direção considerando o ponto mais próximo do atual em relação a rota calculada. Isso é feito em intervalos pré-estabelecidos considerando apontamentos do próprio sistema sobre o melhor intervalo de tempo para atualizar o usuário.

Após essa etapa, o sistema entrega uma saída que, como é de se esperar, é a própria direção a seguir. Essa direção é um valor convencionado para esse sistema baseada na quantidade de direções pré-estabelecidas.

Figura 10 – Modelo Conceitual do Projeto

FONTE: DO AUTOR, 2015

Este valor de saída, o aplicativo envia ao controlador da luva. O controlador é responsável por processsar os dados oriundos do aplicativos e convertê-los em acionamento dos atuadores. Na prática, o usuário terá a próxima direção indicada entre oito possíveis. Essas oito direções são informadas a partir do acionamento dos atuadores, a saber, motores do tipo *vibracall*.

A luva que compõe este projeto é por certo o componente mais peculiar do mesmo. Isso de forma concomitante ao seu funcionamento bastante simplório. A peculiaridade pertinente a mesma deve-se ao fato de ser análoga a um objeto bastante comum (uma luva) e oferecer tão poderosa funcionalidade. O funcionamento, por demais simples, garante intuitividade e replicabilidade de forma a possibilitar o baixo custo do projeto.

A luva difere das demais por possuir atuadores táteis, um em cada dedo. São motores de vibração e trabalhando sistematicamente, oferecem ao usuário a direção a seguir a partir de alimentação prévia do sistema.

É utilizada luva em apenas uma das mãos. As pontas dos dedos do usuário devem ficar de fora, portanto a luva confeccionada não cobre toda a mão. Também por isso, os atuadores ficam mais próximos das bases dos dedos que das extremidades.

4.3 Especificações da luva

A luva é o principal componente de interface deste trabalho no que tange deslocamento e, para que efetivamente seu uso seja relevante na tarefa a que se propõe, faz-se necessário enumerar alguns detalhes de utilização.

Para este fim, algumas convenções foram estabelecidas. A primeira a ser mencionada diz respeito ao direcionamento do sistema, mais especificamente quais as direções que o sistema tem por saída. Foram escolhidos os conhecidos pontos cardeais e os colaterais para representar a saída deste, especialmente pelo fato de serem amplamente conhecidos.

São pontos cardeais:

- Norte;
- Sul;
- Leste;
- Oeste;

São pontos colaterais:

- Nordeste;
- Sudeste;
- Noroeste;
- Sudoeste;

Para representar os referidos pontos, serão usados os motores de vibração presentes na luva. Os referidos motores são acionados na direção indicada obedecendo à convenção mencionada na tabela a seguir que indica o valor recebido do aplicativo, a direção a seguir e o dedo que terá o motor acionado.

Tabela 3: Tabela de valores, direções e acionamento de motores.

Valor	Direção	Motor(es) acionado(s)
0	Nenhuma	Nenhum
1	Norte	Médio
2	Nordeste	Médio e indicador
3	Leste	Polegar
4	Sudeste	Polegar e indicador
5	Sul	Polegar, médio e mínimo
6	Sudoeste	Mínimo e anelar
7	Oeste	Mínimo
8	Noroeste	Anelar e médio
9	Parada	Todos os dedos

FONTE: DO AUTOR, 2015

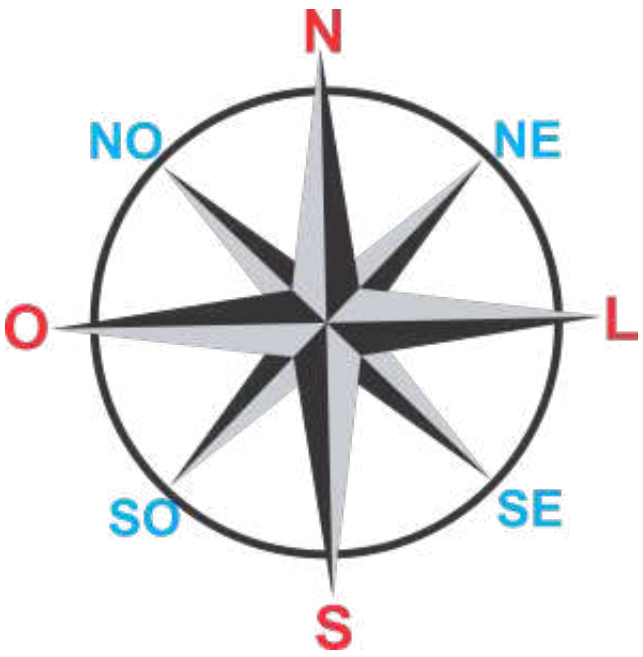
Sobre o acionamento de parada, o que há para ser dito, refere-se à funcionalidade. A parada será ativada sempre que algum erro ocorrer no sistema (erro de qualquer tipo como falha na comunicação) ou quando o usuário chegar ao destino.

Sobre o tempo de acionamento dos motores quando a luva for utilizada, os motores para quaisquer dos comandos serão acionados e assim permanecerão pelo intervalo de um segundo. O aplicativo realimenta o controlador a cada cinco segundos, o que garante ao usuário uma nova saída em tempo hábil.

Para este trabalho, foi convencionado também que a luva é a esquerda, o que permite total liberdade de utilização da mão direita.

A convenção utilizada nos motores a serem acionados para cada direção levou em consideração uma lógica muito simples que deve servir para facilitar a vida do usuário. Pode-se notar que percorrendo a rosa dos ventos (veja figura abaixo), partindo da direção norte e indo em sentido horário, os motores acionados seguem o fato que se for ponto cardeal o número de motores acionados é ímpar e se for ponto colateral é par.

Figura 11 – Rosa dos ventos



FONTE: (VOGAS, 2014)

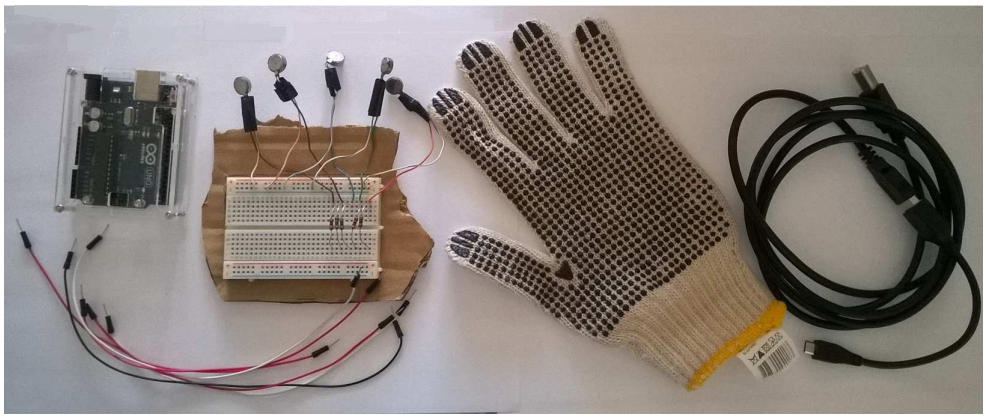
A parada, dado o caráter emergencial, foge a regra. Se a direção colateral for adjacente ao norte, o dedo médio será acionado, se adjacente ao sul ou mínimo ou polegar, considerando o lado, que deve ser acionado. Com a mão esquerda com a palma para baixo, a compreensão do acionamento é bastante intuitiva.

4.4 Materiais

Basicamente, para o desenvolvimento deste protótipo foram utilizados um Arduino UNO R3, uma protoboard de quatrocentos furos, cinco resistores de oitenta e dois ohms, cinco motores de vibração e diversos jumpers.

Fora isso, uma luva de pigmento simples e um cabo USB para estabelecer conexão com o aparelho smartphone.

Figura 12 – Materiais utilizados



FONTE: DO AUTOR, 2015

4.5 Implementação

A luva fora projetada a partir do programa para Arduino. Uma vez implementado o programa, sabe-se quais portas utilizar na confecção da luva. A seguir o trecho do código correspondente a função setup() do programa que controla o Arduino e consequentemente a luva, onde são configuradas as portas que corresponderão à saída dos motores Arduino e o comentário das mesmas informando qual dedo as mesmas aciona.

```
void setup() {
        pinMode(2, OUTPUT); //Dedo Polegar
        pinMode(3, OUTPUT); //Dedo Indicador
        pinMode(4, OUTPUT); //Dedo Médio
        pinMode(5, OUTPUT); //Dedo Anelar
        pinMode(6, OUTPUT); //Dedo Mínimo
        Serial.begin(9600);
        // Abre a porta serial e atribui valor para 9600bps
}
```

A escolha sobre qual motor deve ser acionado é feita logicamente. O aplicativo fornece um número que é comparado no controlador. Se em determinado laço a sentença se fizer verdadeira, o programa do Arduino indica qual motor acionar.

No código abaixo, se o valor recebido pelo Arduino a partir do aplicativo for '1', ou seja, direção 'NORTE', o motor a ser acionado deverá ser o motor '4', que corresponde ao dedo médio. Pela tabela de referência de direção já citada, o usuário deverá interpretar como sentido norte e então seguir a direção a sua frente.

```
if(incomingByte == 1) //NORTE
                {
                    digitalWrite(2, LOW);
                    digitalWrite(3, LOW);
                    digitalWrite(4, HIGH);   //Liga motor de vibração
                    digitalWrite(5, LOW);    //(HIGH é voltagem alta)
                    digitalWrite(6, LOW);
                    delay(1000);
                }
```

Uma vez tendo ajustado as portas, agora mostrar-se-á a estrutura do Arduino. Na referida estrutura deve-se observar as portas citadas no código, pois elas corresponderão as portas que serão utilizadas. Além das portas de '2' a '6', serão utilizadas as portas de terra (GRD) após a porta '13' e a porta USB para alimentação da placa Arduino via smartphone e consequente trânsito de dados.

Uma vez conectados os jumpers às portas respectivas, os mesmos serão conectados a protoboard para ajuste na tensão. Ora, a tensão que sai das portas da placa Arduino é de 5V e os motores vibracall suportam apenas 3,6V, funcionando bem a valores próximos a 3V. Valores próximos a 3,6V fazem os motores esquentar muito rapidamente e aumentam o risco dos mesmos queimarem a curto prazo.

Assim, para diminuir a tensão de saída, são utilizados resistores de 82Ω. Após a conexão destes nos respectivos canais, os motores podem ser conectados e então iniciar-se-á a utilização.

5 TESTE E ANÁLISE DE RESULTADOS

A próxima sessão descreverá o processo de testes do sistema FollowVibe. Serão descritos os dispositivos utilizados para realizar a execução do aplicativo e do software de direcionamento de vibrações. Serão descritos também o ambiente de testes e os resultados obtidos, bem como a análise destes resultados.

5.1 Ambiente de testes

O teste fora realizado no dentro do campus Paulo VI da Universidade Estadual do Maranhão. Como requisito de realização do teste, o local onde o sistema será operado deve ser mapeado pelo Google Maps.

Foram selecionados três pontos distintos dentro da Universidade. Foram eles:

- Centro de Ciências Agrárias, definido no sistema como “agronomia”;
- Centro de Ciências Tecnológicas, definido no sistema como “engenharia”;
- Uemanet, definido como “uemanet”;

Para estabelecer a rota, foi utilizado uma rede de dados móvel. O dispositivo utilizado foi um smartphone que possui o sistema operacional Android versão 4.4.2 KitKat, com memória RAM de 2 gigabytes e memória de armazenamento interno de 8 gigabytes. O smartphone possui serviço de GPS e o serviço de reconhecimento de voz. Alguns recursos utilizados pelo sistema não funcionam muito bem em versões do Android que utilizam a API de desenvolvimento 17, ou seja, versões do android 4.2.2 para baixo não executarão o software. As versões do Android compatíveis com o sistema são as versões KitKat(4.4) e superiores.

5.2 Metodologia de avaliação de resultados

Para avaliar os resultados, adicionamos um componente visual no aplicativo que mostrará o resultado do cálculo de direção. Assim, durante o percurso, poderemos avaliar o número de direcionamentos efetivos realizados pelo sistema. Foram adicionados também componentes visuais para mostrar as coordenadas atuais do usuário. A cada atualização das coordenadas, uma nova direção será disponibilizada.

A primeira rota realizada foi Centro de Ciências Tecnológicas – Centro de Ciências Agrárias. Uma vez localizada a direção norte, o sistema funcionou com normalidade no deslocamento até o prédio designado como destino. Observou-se que a imprecisão do GPS pouco compromete a avaliação do sistema como sistema de navegação.

Isso implica que uma vez que fora selecionado o Centro de Ciências Agrárias como destino, o local que o sistema indicará será muito próximo, como de fato foi, do que fora

armazenado. O sistema apontou que o destino fora alcançado com sucesso a uma distância de menos de cinco metros do que fora previamente marcado.

Os motores de vibração funcionaram normalmente.

A tabela a seguir aponta as direções desejadas e as que o sistema calculou até o destino.

Tabela 4: Tabela de valores de possíveis direções para a rota "engenharia - agronomia".

Valor desejado	Valor real
Direita	Direita
Direita	Direita
Direita	Direita
Direita	Direita
Direita	Sudoeste
Direita	Sudoeste
Sudoeste	Sudoeste
Sudoeste	Sudoeste
Sudoeste	Sudoeste
Direita	Sudoeste
Direita	Direita
Direita	Direita
Direita	Direita
Direita	Direita
Direita	Direita
Destino	Destino

FONTE: DO AUTOR, 2015.

O segundo destino escolhido foi o caminho de volta até o Centro de Ciências Tecnológicas, partindo do Centro de Ciências Agrárias. Novamente, o sistema se mostrou eficaz. Os resultados não diferiram muito do anterior. Novamente pode-se observar uma diferença do ponto previamente marcado com destino e o indicado pelo sistema de aproximadamente cinco metros.

Para terceira rota, fora escolhida uma ida do Centro de Ciências Tecnológicas até a Uemanet. Um ponto importante a ser considerado é que este percurso é mais de quatro vezes e meia maior do que os anteriormente percorridos. Uma vez acionado, novamente o sistema teve uma taxa muito boa de acertos.

Ao término do experimento e de posse dos resultados, a análise que pode-se fazer é satisfatória. O sistema obteve êxito nas três tomadas que foram realizadas. Alguns desacertos em relação ao ideal, o que em grande parte se deve ao fato de não ser utilizado nenhum aparato de inteligência artificial e a limitação do GPS. Mesmo assim, as diferenças entre o ideal e o resultado obtido não comprometeram o resultado, dado que a rota apontada levou ao destino desejado.

A interface do aplicativo se mostrou eficaz. A inserção de destino é realizada a contento, eventualmente devendo ser repetida por limitação da própria API, sem entretanto comprometer o funcionamento.

Tabela 5: Tabela de valores de possíveis direções para a rota "agronomia - engenharia".

Valor desejado	Valor real
Esquerda	Esquerda
Esquerda	Esquerda
Esquerda	Esquerda
Esquerda	Esquerda
Esquerda	Esquerda
Noroeste	Esquerda
Noroeste	Noroeste
Noroeste	Noroeste
Noroeste	Noroeste
Noroeste	Noroeste
Noroeste	Noroeste
Esquerda	Noroeste
Esquerda	Noroeste
Esquerda	Esquerda
Esquerda	Esquerda
Esquerda	Esquerda
Esquerda	Esquerda
Esquerda	Esquerda
Destino	Destino

FONTE: DO AUTOR, 2015

A interface tátil também foi bem-sucedida, dado que respeita as limitações de um possível usuário. Entretanto algumas melhorias deverão ser efetuadas. Algumas vezes, o acionamento de mais de um motor gerou dúvida sobre a próxima direção, o que exigia do operador aumento no nível de atenção para determinar qual era a saída indicada pelo sistema. Uma provável hipótese é que a proximidade entre os motores de vibração cause essa ambiguidade, o que pode ser resolvido com o deslocamento dos motores para a falange central dos dedos.

Tabela 6: Tabela de valores de possíveis direções para a rota "agronomia - uemanet".

Valor desejado	Valor real
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Sudeste	Sudeste
Leste	Sudeste
Leste	Sudeste
Leste	Nordeste
Leste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Nordeste
Nordeste	Destino
Nordeste	Destino
Nordeste	Destino
Destino	Destino

FONTE: DO AUTOR, 2015

6 CONCLUSÃO

Esta pesquisa apresentou uma parte do sistema Display Light, mais especificamente o dispositivo de interface tátil do aplicativo FollowVibe que é componente desse sistema. O sistema é focado na locomoção de deficientes visuais com ênfase em georreferenciamento.

O dispositivo descrito é composto de uma placa do tipo Arduino UNO e com um programa carregado, seleciona que motor(es) de vibração deve(m) ser acionado(s), dadas as entradas do aplicativo.

O aplicativo recebe o destino via comando de voz. Utilizando a internet e locais pré-selecionados, o aplicativo recebe a rota. O aplicativo então processa os dados recebidos e a direção seguinte das possíveis. Assim, este envia um número inteiro ao controlador Arduino, que conforme fora dito, indicará via motor de vibração que direção seguir.

Foram pré-carregados no aplicativo, locais comuns da Universidade Estadual do Maranhão, para efeito de testes. Os locais foram o Centro de Ciências Tecnológicas, a Uemanet e o Centro de Ciências Agrárias.

6.1 Trabalhos Futuros

Embora o sistema tenha apresentado funcionamento bastante satisfatório alguns incrementos podem ser feitos no sentido de o tornar mais útil a comunidade de deficientes visuais. Para este momento, apresentam-se as seguintes propostas:

- Utilizar lógica *fuzzy* para melhorar a precisão no direcionamento.
- Disponibilizar ao usuário a direção a seguir independente de referenciamento prévio, utilizando os sensores disponibilizados pela API do Android.
- Utilizar comunicação sem fio entre os componentes do sistema.
- Desenvolver dispositivo proprietário, reduzindo assim os custos de desenvolvimento.

Referências

ARDUINO. *Arduino Uno and Genuino Uno*. 2015. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 09.11.2015.

ARDUINO. *What is Arduino?* 2015. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 28.11.2015.

BANZI, M. *Primeiros Passos com Arduino*. São Paulo: Novatec, 2012.

BERSCH, R. *Introdução às Tecnologias Assistivas*. 2008. Disponível em: <http://www.assistiva.com.br/Introducao_Tecnologia_Assistiva.pdf>. Acesso em: 10.08.2015.

BOYLESTAD, R. L. *Introdução à Análise de Circuitos*. 10. ed. São Paulo: Prentice Hall, 2004.

BRAMBRING, M. *Mobility and orientation processes of the blind*. Dordrecht, The Netherlands: In Warren D.H. and Strelow E.R., 1985. (Electronic spatial sensing for the blind).

BRASIL. *DECRETO Nº 3.298, DE 20 DE DEZEMBRO DE 1999. Regulamenta a Lei no 7.853, de 24 de outubro de 1989, dispõe sobre a Política Nacional para a Integração da Pessoa Portadora de Deficiência, consolida as normas de proteção, e dá outras providências.* [S.l.]: Diário Oficial da Republica Federativa do Brasil, 1999.

BRASIL. *Programa de Capacitação de Recursos Humanos do Ensino Fundamental – Deficiência Visual*. Brasília: MEC/ SEESP, 2001. I. (Atualidades Pedagógicas 06, I).

BRASIL. *Tecnologia Assistiva*. [S.l.]: Comitê de Ajudas Técnicas, 2009.

DAHL, O. J.; DIJKSTRA, E. W.; HOARE, C. A. R. *Structured Programming*. London: Academic Press, 1972.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. *Arduino em ação*. 1. ed. São Paulo: Manning Publications Co, 2013.

FARRER, H. et al. *Algoritmos Estruturados*. 2. ed. [S.l.]: LTC, 1989.

FILHO, J. de S. R. et al. *Argos - auxílio à locomoção de deficientes visuais a partir de pulseira microcontrolada*. 2011.

FILHO, T. G. Tecnologia assistiva: favorecendo práticas pedagógicas inclusivas. *PROFISSÃO MESTRE*, 2010.

G1. *23,9% dos brasileiros declaram ter alguma deficiência, diz IBGE*. 2012. Disponível em: <http://g1.globo.com/brasil/noticia/2012/04/239-dos-brasileiros-declaram-ter-alguma-deficiencia-diz-ibge.html>. Acesso em: 03.12.2015.

GIL, M. *Cadernos da TV Escola: Deficiência Visual*. [S.l.]: Ministério da Educação, 2000.

GLOBO. *Estudantes criam equipamento para locomoção de deficientes visuais*. 2012. Disponível em: <http://g1.globo.com/mg/sul-de-minas/noticia/2012/04/estudantes-criam-equipamento-para-locomocao-de-deficientes-visuais.html>. Acesso em: 10.11.2015.

HARPER, S.; GREEN, P. A travel flow and mobility framework for visually impaired travellers. *University of Manchester*, 2002.

HEIDRICH, L. Paradigmas de programação – imperativo x orientado a objeto. *Unisinos*, 2009.

HERSH, M. A.; JOHNSON, M. A. *Assistive Technology for Visually Impaired and Blind People*. 1. ed. USA: Springer, 2008.

MICRODRIVES, P. *Product Data Sheet Pico Vibe^{TM} 10mm Vibration Motor - 3mm Type*. London: Precision Microdrives Limited, 2016. Disponível em: <https://catalog.precisionmicrodrives.com/order-parts/datasheet/310-113-10mm-vibration-motor-3mm-type>.

NILSSON, J. W.; RIEDEL, S. A. *Circuitos Elétricos*. 8. ed. São Paulo: Prentice Hall, 2008.

ORGANIZATION World Health. *Visual impairment and blindness*. 2014. Disponível em: <http://www.who.int/mediacentre/factsheets/fs282/en/>. Acesso em: 01.12.2015.

PRODANOV, C. C.; FREITAS, E. C. de. *Metodologia do trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico*. 2. ed. Novo Hamburgo - Rio Grande do Sul: Universidade Feevale, 2013.

RADABAUGH, M. P. Study on the financing of assistive technology devices of services for individuals with disabilities. *A report to the president and the congress of the United State, National Council on Disability*, mar. 1993.

ROBERTS, P. *Imperative Vs Declarative*. 2015. Disponível em: <http://latentflip.com/imperative-vs-declarative>. Acesso em: 08.11.2015.

Sá, E. D. de. *Acessibilidade: as pessoas cegas no itinerário da cidadania. In: Inclusão: Revista da Educação Especial.* [S.l.]: MEC, v.1, n.1, p.13-18, 2005.

SARTORETTO, M. L.; BERSCH, R. *O que é Tecnologia Assistiva?* 2015. Disponível em: <http://www.assistiva.com.br/tassistiva.html#topo>. Acesso em: 15.12.2015.

SEBESTA, R. W. *Concepts of Programming Language*. 10. ed. [S.l.]: Pearson, 2009.

SHAMS, L.; KAMITANI, Y.; SHIMOJO, S. A visual illusion induced by sound. *Multisensory Research Conference*, 2000.

TELFORD, C. W.; SAWREY, J. M. *O Indivíduo Excepcional*. 3. ed. Rio de Janeiro: Zahar Editores, 1972.

VOGAS, H. *Níveis de Conhecimento: Por Onde Começar e Até Onde Você Deve Estudar Cada Assunto?* 2014. Disponível em: <http://triadedaaprovacao.com/niveis-de-conhecimento-por-onde-comecar-e-ate-onde-voce-deve-estudar-cada-assunto/>. Acesso em: 17.12.2015.

WIKIPEDIA. 2013. Disponível em: <https://upload.wikimedia.org/wikipedia/commons/3/38/Arduino_Uno_-_R3.jpg>. Acesso em: 12.11.2015.

YOUNG, H. D.; FREEDMAN, R. A. *Física III: Eletromagnetismo*. 12. ed. São Paulo: Addison Wesley, 2009.

Apêndices

APÊNDICE A – Código-fonte do aplicativo FollowVibe

```
  # -*- coding: utf-8 -*-
/**
 * Classe MapActivity
 *
 */
package com.example.antonio.darkdisplayv2;

import android.app.FragmentTransaction;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.GoogleMapOptions;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;

import org.json.JSONException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

// Codigo de conexao e comunicacao com arduino desenvolvido por Eric do Androider.
// Site: http://android-er.blogspot.com.br/2014/09/send-data-from-android-to-arduino-uno.html
```

```
//

public class MapActivity extends AppCompatActivity implements
        OnMapReadyCallback,
        LocationListener,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener {

    public static final int RESULT_SPEECH = 1;
    private GoogleMap       mGoogleMap;
    private GoogleApiClient mGoogleApiClient;
    private Location        mCurrentLocation;
    private LocationRequest mLocationRequest;
    private boolean         mRequestingLocationUpdates;
    private String          mRequestingRouter;
    private Route           mRoute;
    private LatLng          mDestination;
    private List<Marker>    mArrayMarkers;
    private boolean         isDestination;
    private Map<String, LatLng> pontosDestino;

    private TextView latitude;
    private TextView longitude;
    private TextView mDirection;
    private TextView voiceText;

    /*Variaveis para a comunicacao com o arduino*/

    private static final int targetVendorID = 2341;
    private static final int targetProductID = 43;

    UsbDevice deviceFound          = null;
    UsbInterface usbInterfaceFound = null;
    UsbEndpoint endpointIn         = null;
    UsbEndpoint endpointOut        = null;

    private static final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";

    PendingIntent mPermissionIntent;
    UsbInterface usbInterface;
    UsbDeviceConnection usbDeviceConnection;

    private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (ACTION_USB_PERMISSION.equals(action)) {

                Toast.makeText(MapActivity.this, "ACTION_USB_PERMISSION",
                        Toast.LENGTH_LONG).show();

                synchronized (this) {
                    UsbDevice device = (UsbDevice) intent
                            .getParcelableExtra(UsbManager.EXTRA_DEVICE);

                    if (intent.getBooleanExtra(
```

```
                        UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                    if (device != null) {
                        connectUsb();
                    }
                } else {
                    Toast.makeText(MapActivity.this,
                            "permission denied for device " + device,
                            Toast.LENGTH_LONG).show();
                }
            }
        }
    }
};

private final BroadcastReceiver mUsbDeviceReceiver = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (UsbManager.ACTION_USB_DEVICE_ATTACHED.equals(action)) {

            deviceFound = (UsbDevice) intent
                    .getParcelableExtra(UsbManager.EXTRA_DEVICE);
            Toast.makeText(
                    MapActivity.this,
                    "ACTION_USB_DEVICE_ATTACHED: \n"
                            + deviceFound.toString(), Toast.LENGTH_LONG)
                    .show();

            connectUsb();

        } else if (UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)) {

            UsbDevice device = (UsbDevice) intent
                    .getParcelableExtra(UsbManager.EXTRA_DEVICE);

            Toast.makeText(MapActivity.this,
                    "ACTION_USB_DEVICE_DETACHED: \n" + device.toString(),
                    Toast.LENGTH_LONG).show();

            if (device != null) {
                if (device == deviceFound) {
                    releaseUsb();
                }else{
                    Toast.makeText(MapActivity.this,
                            "device == deviceFound, no call releaseUsb()\n" +
                                    device.toString() + "\n" +
                                    deviceFound.toString(),
                            Toast.LENGTH_LONG).show();
                }
            }else{
                Toast.makeText(MapActivity.this,
                        "device == null, no call releaseUsb()", Toast.LENGTH_LONG).show();
            }
        }
    }
```

```
};


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_map);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    voiceText = (TextView) findViewById(R.id.textVoice);

    pontosDestino = new HashMap<String, LatLng>();

    pontosDestino.put("agronomia", new LatLng(-2.583090, -44.208368));
    pontosDestino.put("engenharia", new LatLng(-2.582919, -44.209320));
    pontosDestino.put("uemanet", new LatLng(-2.580460, -44.206854));
    pontosDestino.put("letras", new LatLng(-2.575889, -44.209983));

    // Configuracoes iniciais do Google Maps
    //
    GoogleMapOptions options = new GoogleMapOptions();
    options.mapType(GoogleMap.MAP_TYPE_HYBRID);

    // Colocando o fragment que contera o mapa na atividade
    //
    MapFragment mMapFragment = MapFragment.newInstance(options);

    // Setando a interface que contera o metodo que sera
    // chamado quando o mapa estiver pronto
    //
    mMapFragment.getMapAsync(this);

    FragmentTransaction fragmentTransaction =
            getFragmentManager().beginTransaction();
    fragmentTransaction.add(R.id.mapContainer, mMapFragment);
    fragmentTransaction.commit();

    // Criando uma instancia de GoogleApiClient
    //
    this.buildGoogleApiClient();

    // Criando um objeto de configuracoes de atualizacao de rotas
    //
    this.createLocationRequest();

    // Obtendo os textViews de latitude e longitude
    //
    latitude      = (TextView) findViewById(R.id.textCurrentLatitude);
    longitude     = (TextView) findViewById(R.id.textCurrentLongitude);
    mDirection    = (TextView) findViewById(R.id.textDirection);
    mArrayMarkers = new ArrayList<Marker>();
    isDestination = false;

    // Configuracao da conexao com o arduino
    //
    // register the broadcast receiver
```

```
        mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(
                ACTION_USB_PERMISSION), 0);
        IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
        registerReceiver(mUsbReceiver, filter);
        registerReceiver(mUsbDeviceReceiver, new IntentFilter(
                UsbManager.ACTION_USB_DEVICE_ATTACHED));
        registerReceiver(mUsbDeviceReceiver, new IntentFilter(
                UsbManager.ACTION_USB_DEVICE_DETACHED));

        // Conexao com o arduino;
        //
        connectUsb();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_map, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void onStart() {
        super.onStart();
        mGoogleApiClient.connect();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

    @Override
    protected void onStop() {

        super.onStop();
```

```
        this.stopLocationUpdates();
        mGoogleApiClient.disconnect();

        mRequestingLocationUpdates = false;
    }

    @Override
    protected void onDestroy() {
        releaseUsb();
        unregisterReceiver(mUsbReceiver);
        unregisterReceiver(mUsbDeviceReceiver);
        super.onDestroy();
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mGoogleMap = googleMap;
        mRoute     = new Route(googleMap);

        this.configClickAction();
    }

    @Override
    public void onConnected(Bundle bundle) {
        mCurrentLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);

        latitude.setText("Lat: "+mCurrentLocation.getLatitude());
        longitude.setText("Lng: "+mCurrentLocation.getLongitude());

        this.startLocationUpdates();
        mRequestingLocationUpdates = true;

        configLocation(new LatLng(mCurrentLocation.getLatitude(),
         mCurrentLocation.getLongitude()));

        getVoiceLocal();
    }

    @Override
    public void onConnectionSuspended(int i) {
        // Do nothing
    }

    @Override
    public void onLocationChanged(Location location) {
        mCurrentLocation = location;
        updateUI();

        if(!isDestination) {
            if(mRoute.getPoints().size() != 0) {
                int index = PositionUtil.getClosestPointOfLatLngList(
                        new LatLng(location.getLatitude(), location.getLongitude()),
                        mRoute.getPoints());

                Log.i("DATA","Index: "+index+" - Size: "+mRoute.getPoints().size()+" Points:
                "+mRoute.getPoints().toString());
```

```
            if(index == mRoute.getPoints().size() - 1) {
                isDestination = true;
                mDirection.setText("Chegou ao destino!");
            } else {
                int direction = PositionUtil.getDirectionRelationNorth(
                        mRoute.getPoints().get(index),
                        mRoute.getPoints().get(index + 1));

                // Enviando os dados para a luva
                //
                sendDirectionToGlove(direction);

                String way = PositionUtil.indicateDirection(direction);
                mDirection.setText(way);
            }
        } else {
            mDirection.setText("Aguardando...");
        }

    } else {
        mDirection.setText("Chegou ao destino!");
    }

    configLocation(new LatLng(location.getLatitude(),location.getLongitude()));
}

@Override
public void onConnectionFailed(ConnectionResult connectionResult) {

}

protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();
}

protected void createLocationRequest() {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(10000);
    mLocationRequest.setFastestInterval(5000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}

protected void startLocationUpdates() {
    LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
}

protected void stopLocationUpdates() {
    LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
}

public void updateUI() {
    latitude.setText("Lat: " + mCurrentLocation.getLatitude());
```

```
        longitude.setText("Lng: " + mCurrentLocation.getLongitude());
    }

    public void configLocation(LatLng location) {
        CameraPosition camera = new CameraPosition.Builder()
                .target(location)
                .zoom(17)
                .build();
        CameraUpdate cameraUpdate = CameraUpdateFactory.newCameraPosition(camera);

        this.mGoogleMap.setMyLocationEnabled(true);
        this.mGoogleMap.animateCamera(cameraUpdate);

        UserPosition userPosition = new UserPosition();
        mGoogleMap.setLocationSource(userPosition);
        userPosition.setLocation(location);
    }

    public void getRoute() {

        if(mDestination != null) {
            String url = "https://maps.googleapis.com/maps/api/directions/json?" +
                    "origin=" +mCurrentLocation.getLatitude()+","
                    +mCurrentLocation.getLongitude()+
                    "&destination="+mDestination.latitude+","
                    +mDestination.longitude+"&sensor=false&key=" +
                    "AIzaSyDeOrRD5AtzVZAtonCcC1aPFIbfZzkVEHY&mode=walking";

            // Montando a requisicao, que recebera uma resposta do tipo String
            //
            StringRequest request = new StringRequest(Request.Method.GET, url,
                    new Response.Listener<String>() {
                        @Override
                        public void onResponse(String responseValue) {

                            mRequestingRouter = responseValue;
                            drawRoute();
                            Toast.makeText(MapActivity.this,
                             "Rota obtida com sucesso.", Toast.LENGTH_LONG).show();
                        }
                    }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {

                    // Mostrando no console de erros o que aconteceu
                    //
                    Log.i("ERROR_REQUEST", error.toString());

                    // Mostrando ao usuario que nao foi possivel realizar a requisicao
                    //
                    Toast.makeText(MapActivity.this,
                    "Nao foi possivel realizar a requisicao da rota.",
                    Toast.LENGTH_LONG).show();
                }
            });

            VolleySingleton.getmInstance(this).addToRequestQueue(request);
```

```
        } else {
            Toast.makeText(this,
            "Nao e possivel obter a rota: Indique a origem e o destino primeiro.",
            Toast.LENGTH_LONG).show();
        }
    }

    public void drawRoute() {
        // Verificando se a resposta foi atribuida
        //
        if (mRequestingRouter == null) {
            Toast.makeText(this,
            "Rota nao foi obtida, obtenha primeiramente a rota",
            Toast.LENGTH_LONG).show();
            return;
        }

        // Realizando a tentativa de transformar a string em um objeto json
        //
        try {
            List<LatLng> list = ParseJson.parseGoogleMapsJson(mRequestingRouter);

            for (LatLng point : list) {

                // Passando os pontos para o objeto de desenho
                //
                mRoute.addPoint(point);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    public void configClickAction() {
        mGoogleMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
            @Override
            public void onMapClick(LatLng latLng) {
                getVoiceLocal();
            }
        });
    }

    public void resetMap() {
        mDestination      = null;
        mRequestingRouter = null;
        isDestination     = false;
        mRoute.clearRoute();
        for(Marker mMarker : mArrayMarkers) {
            mMarker.remove();
        }
    }

    public void getVoiceLocal() {
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, "pt-BR");

        try {
```

```
            startActivityForResult(intent, RESULT_SPEECH);
            voiceText.setText("");
        } catch (ActivityNotFoundException e) {
            Toast.makeText(getApplicationContext(),
                    "Opps! Your device doesn't support Speech to Text",
                    Toast.LENGTH_SHORT).show();
        }
    }

    public void getVoiceLocal(View view) {
        getVoiceLocal();
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        switch(requestCode) {
            case RESULT_SPEECH: {
                if(resultCode == RESULT_OK && data != null) {
                    ArrayList<String> text = data.getStringArrayListExtra(
                    RecognizerIntent.EXTRA_RESULTS);
                    String destino = text.get(0);
                    voiceText.setText(destino);

                    doMappedAction(destino);
                }
            }
            break;
        }
    }

    public void doMappedAction(String action) {

        String lowerCase = action.toLowerCase();
        if(lowerCase.equals("limpar")) {
            resetMap();
            return;
        }

        for(String key : pontosDestino.keySet()) {
            if(key.equals(lowerCase)) {
                mDestination = pontosDestino.get(lowerCase);
                getRoute();
            }
        }
    }

    private void connectUsb() {

        Toast.makeText(MapActivity.this, "connectUsb()", Toast.LENGTH_LONG)
                .show();

        searchEndPoint();

        if (usbInterfaceFound != null) {
            setupUsbComm();
```

```
        }

    }

    private void releaseUsb() {

        Toast.makeText(MapActivity.this, "releaseUsb()", Toast.LENGTH_LONG)
                .show();

        if (usbDeviceConnection != null) {
            if (usbInterface != null) {
                usbDeviceConnection.releaseInterface(usbInterface);
                usbInterface = null;
            }
            usbDeviceConnection.close();
            usbDeviceConnection = null;
        }

        deviceFound = null;
        usbInterfaceFound = null;
        endpointIn = null;
        endpointOut = null;
    }

    private void searchEndPoint() {

        usbInterfaceFound = null;
        endpointOut = null;
        endpointIn = null;

        // Search device for targetVendorID and targetProductID
        if (deviceFound == null) {
            UsbManager manager = (UsbManager) getSystemService(Context.USB_SERVICE);
            HashMap<String, UsbDevice> deviceList = manager.getDeviceList();
            Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();

            while (deviceIterator.hasNext()) {
                UsbDevice device = deviceIterator.next();

                if (device.getVendorId() == targetVendorID) {
                    if (device.getProductId() == targetProductID) {
                        deviceFound = device;
                    }
                }
            }
        }

        if (deviceFound == null) {
            Toast.makeText(MapActivity.this, "device not found",
                    Toast.LENGTH_LONG).show();
        } else {
            String s = deviceFound.toString() + "\n" + "DeviceID: "
                    + deviceFound.getDeviceId() + "\n" + "DeviceName: "
                    + deviceFound.getDeviceName() + "\n" + "DeviceClass: "
                    + deviceFound.getDeviceClass() + "\n" + "DeviceSubClass: "
                    + deviceFound.getDeviceSubclass() + "\n" + "VendorID: "
                    + deviceFound.getVendorId() + "\n" + "ProductID: "
```

```
                + deviceFound.getProductId() + "\n" + "InterfaceCount: "
                + deviceFound.getInterfaceCount();

        // Search for UsbInterface with Endpoint of USB_ENDPOINT_XFER_BULK,
        // and direction USB_DIR_OUT and USB_DIR_IN

        for (int i = 0; i < deviceFound.getInterfaceCount(); i++) {
            UsbInterface usbif = deviceFound.getInterface(i);

            UsbEndpoint tOut = null;
            UsbEndpoint tIn = null;

            int tEndpointCnt = usbif.getEndpointCount();
            if (tEndpointCnt >= 2) {
                for (int j = 0; j < tEndpointCnt; j++) {
                    if (usbif.getEndpoint(j).getType() ==
                    UsbConstants.USB_ENDPOINT_XFER_BULK) {
                        if (usbif.getEndpoint(j).getDirection() ==
                        UsbConstants.USB_DIR_OUT) {
                            tOut = usbif.getEndpoint(j);
                        } else if (usbif.getEndpoint(j).getDirection() ==
                        UsbConstants.USB_DIR_IN) {
                            tIn = usbif.getEndpoint(j);
                        }
                    }
                }

                if (tOut != null && tIn != null) {
                    // This interface have both USB_DIR_OUT
                    // and USB_DIR_IN of USB_ENDPOINT_XFER_BULK
                    usbInterfaceFound = usbif;
                    endpointOut = tOut;
                    endpointIn = tIn;
                }
            }
        }
    }
}

private boolean setupUsbComm() {

    // for more info, search SET_LINE_CODING and
    // SET_CONTROL_LINE_STATE in the document:
    // "Universal Serial Bus Class Definitions for Communication Devices"
    // at http://adf.ly/dppFt
    final int RQSID_SET_LINE_CODING = 0x20;
    final int RQSID_SET_CONTROL_LINE_STATE = 0x22;

    boolean success = false;

    UsbManager manager = (UsbManager) getSystemService(Context.USB_SERVICE);
    Boolean permitToRead = manager.hasPermission(deviceFound);

    if (permitToRead) {
        usbDeviceConnection = manager.openDevice(deviceFound);
        if (usbDeviceConnection != null) {
            usbDeviceConnection.claimInterface(usbInterfaceFound, true);
```

```
                int usbResult;
                usbResult = usbDeviceConnection.controlTransfer(0x21, // requestType
                        RQSID_SET_CONTROL_LINE_STATE, // SET_CONTROL_LINE_STATE
                        0, // value
                        0, // index
                        null, // buffer
                        0, // length
                        0); // timeout

                Toast.makeText(
                        MapActivity.this,
                        "controlTransfer(SET_CONTROL_LINE_STATE): " + usbResult,
                        Toast.LENGTH_LONG).show();

                // baud rate = 9600
                // 8 data bit
                // 1 stop bit
                byte[] encodingSetting = new byte[] { (byte) 0x80, 0x25, 0x00,
                        0x00, 0x00, 0x00, 0x08 };
                usbResult = usbDeviceConnection.controlTransfer(0x21, // requestType
                        RQSID_SET_LINE_CODING, // SET_LINE_CODING
                        0, // value
                        0, // index
                        encodingSetting, // buffer
                        7, // length
                        0); // timeout
                Toast.makeText(MapActivity.this,
                        "controlTransfer(RQSID_SET_LINE_CODING): " + usbResult,
                        Toast.LENGTH_LONG).show();
            }

        } else {
            manager.requestPermission(deviceFound, mPermissionIntent);
            Toast.makeText(MapActivity.this, "Permission: " + permitToRead,
                    Toast.LENGTH_LONG).show();
        }

        return success;
    }

    private void sendDirectionToGlove(int direction) {
        if(deviceFound != null){

            int message = direction;
            byte[] bytesOut = ByteBuffer.allocate(4).putInt(message).array();
            int usbResult = usbDeviceConnection.bulkTransfer(
                    endpointOut, bytesOut, bytesOut.length, 0);

        }else{
            Toast.makeText(MapActivity.this,
                    "deviceFound == null",
                    Toast.LENGTH_LONG).show();
        }
    }
}
```

```
/**
 * Classe ParseJson
 *
 */

package com.example.antonio.darkdisplayv2;

import com.google.android.gms.maps.model.LatLng;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class ParseJson {

    // Retorna o caminho entre a origem e o destino em forma de list de LatLngs ,
    // para ser consumido
    // pelo map application, e entao desenhar o caminho no mapa
    //
    public static List<LatLng> parseGoogleMapsJson(String json) throws JSONException {

        JSONObject result = new JSONObject(json);
        JSONArray routes = result.getJSONArray("routes");

        long distanceForSegment = routes.getJSONObject(0).getJSONArray("legs")
                .getJSONObject(0).getJSONObject("distance").getInt("value");

        List<LatLng> lines = new ArrayList<LatLng>();

        JSONArray steps = routes.getJSONObject(0)
         .getJSONArray("legs")
         .getJSONObject(0)
         .getJSONArray("steps");
        for(int i = 0; i < steps.length(); i++) {
            String polyline = steps.getJSONObject(i)
             .getJSONObject("polyline")
             .getString("points");

            for(LatLng p : decodePolyline(polyline)) {
                lines.add(p);
            }
        }

        return lines;
    }

    // Responsavel por pegar o conjunto de pontos criptografados e decolifica-los
    // uma list de LatLng. Metodo obtido da comunidade de desenvolvedores
    //
    private static List<LatLng> decodePolyline(String encoded) {

        List<LatLng> poly = new ArrayList<LatLng>();
```

```
        int index = 0, len = encoded.length();
        int lat = 0, lng = 0;

        while (index < len) {
            int b, shift = 0, result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lat += dlat;

            shift = 0;
            result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lng += dlng;

            LatLng p = new LatLng((((double) lat / 1E5)),
                    (((double) lng / 1E5)));
            poly.add(p);
        }

        return poly;
    }
}



/**
 * Classe PositionUtil
 *
 */

package com.example.antonio.darkdisplayv2;

import com.google.android.gms.maps.model.LatLng;

import java.util.List;


public class PositionUtil {

    public static int getClosestPointOfLatLngList(LatLng point, List<LatLng> points) {

        int index = 0;

        double minDistance = Math.sqrt(
         Math.pow((point.latitude - points.get(0).latitude), 2) +
         Math.pow((point.longitude - points.get(0).longitude), 2));

        for(int i = 0; i < points.size(); i++) {
```

```
            double auxDistance = Math.sqrt(
             Math.pow((point.latitude - points.get(i).latitude), 2) +
             Math.pow((point.longitude - points.get(i).longitude), 2));

            if(auxDistance < minDistance) {
                index = i;
                minDistance = auxDistance;
            }
        }

        return index;
    }

    /**
     * Funcao que retorna a direcao que o usuario deve seguir, em relacao ao norte,
     * na rota
     *
     * 0 - Parado, aguardando informacoes
     * 1 - Para frente
     * 2 - Para o nordeste, frente e direita
     * 3 - Para a direita
     * 4 - Para o sudeste, atras e direita
     * 5 - Para tras
     * 6 - Para o sudoeste, atras e esquerda
     * 7 - Para a esquerda
     * 8 - Para o noroeste, frente e esquerda
     *
     * */
    public static int getDirectionRelationNorth(LatLng now, LatLng prox) {

        int direction = 0; // Significa parado - o usuario deve aguardar informacoes

        if (now.latitude < prox.latitude) { // Frente
            if(now.longitude < prox.longitude) { // Direita
                direction = 2;
            } else if (now.longitude > prox.longitude) { // Esquerda
                direction = 8;
            } else { // Somente para frente
                direction = 1;
            }
        } else if (now.latitude > prox.latitude) { // Atras
            if(now.longitude < prox.longitude) { // Direita
                direction = 4;
            } else if (now.longitude > prox.longitude) { // Esquerda
                direction = 6;
            } else { // Somente para tras
                direction = 5;
            }
        } else { // Ou para esquerda ou para direita
            if(now.longitude < prox.longitude) { // Direita
                direction = 3;
            } else if (now.longitude > prox.longitude) { // Esquerda

                direction = 7;
            } else { // Parado, aguardando informacao
                direction = 0;
            }
```

```
        }

        return direction;
    }

    public static String indicateDirection(int direction) {
        String retorno = "";

        switch (direction) {
            case 0: retorno = "Parado"; break;
            case 1: retorno = "Frente"; break;
            case 2: retorno = "Nordeste"; break;
            case 3: retorno = "Direita"; break;
            case 4: retorno = "Sudeste"; break;
            case 5: retorno = "Atras"; break;
            case 6: retorno = "Sudoeste"; break;
            case 7: retorno = "Esquerda"; break;
            case 8: retorno = "Noroeste"; break;
        }
        return retorno;
    }
}



/**
 * Classe Route
 *
 */

package com.example.antonio.darkdisplayv2;

import android.graphics.Color;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;

import java.util.ArrayList;
import java.util.List;

public class Route {

    private List<LatLng>    mPoints;
    private Polyline        mPolyline;
    private PolylineOptions mPolyOptions;

    public Route(GoogleMap map) {
        mPoints      = new ArrayList<LatLng>();
        mPolyOptions = new PolylineOptions();
        mPolyline    = map.addPolyline(mPolyOptions);
        mPolyline.setColor(Color.BLUE);
    }

    public Route addPoint( LatLng point ) {
        if( point != null ) {
```

```
            mPoints.add(point);
            mPolyline.setPoints(mPoints);
        }
        return this;
    }

    public List<LatLng> getPoints() {
        return mPoints;
    }

    public void clearRoute() {
        this.getPoints().clear();
        mPolyline.setPoints(mPoints);
    }
}


/**
 * Classe UserPosition
 *
 */

package com.example.antonio.darkdisplayv2;

import android.location.Location;
import android.location.LocationManager;

import com.google.android.gms.maps.LocationSource;
import com.google.android.gms.maps.model.LatLng;

public class UserPosition implements LocationSource {

    private OnLocationChangedListener locationChangedListener;
    @Override
    public void activate(OnLocationChangedListener onLocationChangedListener) {
        this.locationChangedListener = onLocationChangedListener;
    }

    @Override
    public void deactivate() {

    }

    public void setLocation(LatLng location) {
        Location loc = new Location(LocationManager.GPS_PROVIDER);

        loc.setLatitude(location.latitude);
        loc.setLongitude(location.longitude);

        if(this.locationChangedListener != null) {
            this.locationChangedListener.onLocationChanged(loc);
        }
    }
}
```

```
/**
 * Classe VolleySingleton
 *
 */

package com.example.antonio.darkdisplayv2;

import android.content.Context;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.Volley;


public class VolleySingleton {
    private static VolleySingleton mInstance;
    private RequestQueue mRequestQueue;
    private static Context mContext;

    private VolleySingleton(Context context) {
        mContext = context;
        mRequestQueue = getRequestQueue();
    }

    public static synchronized VolleySingleton getmInstance(Context context) {
        if(mInstance == null) {
            mInstance = new VolleySingleton(context);
        }

        return mInstance;
    }

    public RequestQueue getRequestQueue() {
        if(mRequestQueue == null) {
            mRequestQueue = Volley.newRequestQueue(mContext.getApplicationContext());
        }

        return mRequestQueue;
    }

    public <T> void addToRequestQueue(Request<T> req) {
        getRequestQueue().add(req);
    }
}
```

APÊNDICE B – Código-fonte da placa Arduino

```
int incomingByte = 0;   // Armazenar valor da porta USB

void setup() {
        pinMode(2, OUTPUT); //Dedo Polegar
        pinMode(3, OUTPUT); //Dedo Indicador
        pinMode(4, OUTPUT); //Dedo Médio
        pinMode(5, OUTPUT); //Dedo Anelar
        pinMode(6, OUTPUT); //Dedo Mínimo
        Serial.begin(9600);     // Abre a porta serial e atribui valor para 9600bps
}

void loop() {

                    digitalWrite(2, LOW);   // Desliga motor (LOW é voltagem baixa)
                    digitalWrite(3, LOW);
                    digitalWrite(4, LOW);
                    digitalWrite(5, LOW);
                    digitalWrite(6, LOW);

        if (Serial.available() > 0) { // Envia dado apenas se receber dado

                incomingByte = Serial.read(); // Ler dado e atribui à variável

                if(incomingByte == 0) //Compara dado da variável com '0'
                {
                    digitalWrite(2, LOW);
                    digitalWrite(3, LOW);
                    digitalWrite(4, LOW);
                    digitalWrite(5, LOW);
                    digitalWrite(6, LOW);
                    delay(1000);              // Espera 1000 milisegundos ( 1segundo )
                }

                if(incomingByte == 1) //NORTE
                {
                    digitalWrite(2, LOW);
                    digitalWrite(3, LOW);
                    digitalWrite(4, HIGH);   //Liga motor de vibração (HIGH é voltagem alta)
                    digitalWrite(5, LOW);
                    digitalWrite(6, LOW);
                    delay(1000);
                }

                if(incomingByte == 2) //NORDESTE
                {
                    digitalWrite(2, LOW);
                    digitalWrite(3, HIGH);
                    digitalWrite(4, HIGH);
                    digitalWrite(5, LOW);
                    digitalWrite(6, LOW);
                    delay(1000);
                }
```

```
if(incomingByte == 3) //LESTE
{
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
    delay(1000);
}

if(incomingByte == 4) //SUDESTE
{
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
    delay(1000);
}

if(incomingByte == 5) //SUL
{
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
    delay(1000);
}

if(incomingByte == 6) //SUDOESTE
{
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    delay(1000);
}

if(incomingByte == 7) //OESTE
{
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
    delay(1000);
}

if(incomingByte == 8) //NOROESTE
{
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
```

```
                digitalWrite(6, LOW);
                delay(1000);
            }

            if(incomingByte == 9) //PARADA
            {
                digitalWrite(2, HIGH);
                digitalWrite(3, HIGH);
                digitalWrite(4, HIGH);
                digitalWrite(5, HIGH);
                digitalWrite(6, HIGH);
                delay(1000);
            }
        }
}
```