ALEXANDRE YOSHIDA KODA
NATAN FIORAVANTI BALCEIRO
RODRIGO HAJIME ITO

GUIA ELETRÔNICO RFID PARA DEFICIENTES VISUAIS

São Paulo
2012

ALEXANDRE YOSHIDA KODA
NATAN FIORAVANTI BALCEIRO
RODRIGO HAJIME ITO

GUIA ELETRÔNICO RFID PARA DEFICIENTES VISUAIS

Trabalho de Conclusão de Curso
apresentado à Escola Politécnica da
Universidade de São Paulo

Área de Concentração: Engenharia Elétrica
(Sistemas Eletrônicos)

Orientador: Prof. Dr. Wang Jiang Chau

São Paulo
2012

ALEXANDRE YOSHIDA KODA
NATAN FIORAVANTI BALCEIRO
RODRIGO HAJIME ITO

GUIA ELETRÔNICO RFID PARA DEFICIENTES VISUAIS

Trabalho de Conclusão de Curso
apresentado à Escola Politécnica da
Universidade de São Paulo

São Paulo
2012

RESUMO

Este trabalho busca propor e descrever a fase inicial do desenvolvimento de um projeto na área de sistemas eletrônicos para auxiliar deficientes visuais a se locomoverem de forma autônoma em locais desconhecidos. Para isso, é proposto um sistema baseado em tecnologia de identificação por radio freqüência (RFID), juntamente com uma interface com o usuário utilizando *smartphones* com plataforma Android. O local deve ser devidamente preparado com piso tátil e etiquetas RFID passivas de 125kHz ao longo desse piso. Um sistema para ler as *tags* é proposto utilizando um módulo de leitura RFID, um microcontrolador e uma antena acoplada a uma bengala. O usuário primeiramente deve informar (por voz) o local de destino; ao caminhar pelo piso tátil, o sistema detectará a presença das *tags* no caminho, sabendo qual a sua localização e para onde guiá-lo, falando pelo *smartphone* as instruções do caminho.

Palavras-chave: Deficientes visuais. Guia. RFID. Android.

ABSTRACT

This work aims to propose and describe the initial development stage of a project in the field of electronic systems which helps visually impaired users to walk autonomously in unknown places. In this regard, it is proposed a radio-frequency identification (RFID) based system with an User Interface implemented on smart-phones with Android platform. The environment must be properly prepared with tactile paving and, along it, with 125kHz RFID passive tags. Due to read the 125 kHz tags, it is proposed a system using a RFID reader module, a microcontroller and an antenna coupled to a cane. The user has to inform initially (by voice) the target place; while walking along the tactile paving, the system will detect the presence of corresponding tags in the path, so that the system knows his location and can guide him, talking the route instructions through the smart-phone.

Key-words: Visually impaired. RFID. Guide. Android

.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code for Information Interchange
bps	Bits por segundo
cm	Centímetro
GHz	Gigahertz
HF	High frequency
Hz	Hertz
ID	Identification
kHz	Kilohertz
LF	Low frequency
MHz	Megahertz
RFID	Radio-Frequency IDentification
RX	Receptor
SMT	Surface-mount technology
TX	Transmissor
UART	Universal Asynchronous Receiver/Transmitter
UHF	Ultra high frequency

SUMÁRIO

1. INTRODUÇÃO

Segundo dados do IBGE de 2010, mais de 6,5 milhões de pessoas têm alguma deficiência visual no Brasil. Desse total, cerca de 530 mil pessoas são incapazes de enxergar (cegos) e seis milhões possuem grande dificuldade permanente de enxergar (baixa visão ou visão subnormal) (FUNDAÇÃO DORINA NORWILL PARA CEGOS).

O ato de ir e vir, tão simples para as pessoas sem nenhum tipo de deficiência, é uma das maiores dificuldades das pessoas com deficiência visual. Uma solução encontrada para essa dificuldade foi o uso do cão-guia, que é capaz de conduzir o deficiente de um ponto A até um ponto B em linha reta, parando em todas as mudanças de elevação (escadas, meio fio, etc) e conduzir o deficiente em torno de obstáculos. No entanto, mesmo com a ajuda do cão-guia, o usuário deve saber o caminho para chegar ao destino desejado, já que o cão deve receber comandos de orientação do seu dono durante a caminhada (INSTITUTO DE RESPONSABILIDADE E INCLUSÃO SOCIAL).

Dessa forma, os deficientes visuais ainda encontram uma grande dificuldade para caminhar em locais desconhecidos, necessitando sempre da ajuda de outras pessoas para dar informações sobre o local. Para minimizar essa dificuldade dos deficientes visuais, esse projeto propõe o desenvolvimento de um sistema eletrônico capaz guiá-los em ambientes previamente preparados utilizando *tags* RFID (identificação por radiofreqüência).

Nesse projeto será desenvolvido um sistema eletrônico para guiar deficientes físicos durante a localização de locais específicos em ambientes fechados preparados. Para isso, *tags* RFID devem ser previamente posicionadas em diversos pontos do ambiente e suas localizações devem ser armazenadas num banco de dados de forma a criar um mapeamento dos locais conhecidos. Utilizando um transceptor de radiofreqüência, as *tags* mais próximas são identificadas, sendo possível determinar sua posição no mapeamento do ambiente e fornecendo ao usuário o caminho a ser tomado para chegar ao destino desejado.

A interface com o usuário será implementada utilizando smartphones em plataforma Android devido a sua grande popularidade e a facilidade de implementar comandos de voz utilizando APIs *TextToSpeech*.

2. TECNOLOGIA RFID

Identificação por radiofreqüência consiste numa tecnologia de identificação automática que se baseia no armazenamento e recuperação de dados remotamente usando *tags* (etiquetas) e leitores (ZHOU e SHI, 2009). Essas *tags* contém *transponders* (transmissores de respostas) que emitem mensagens legíveis por leitores RFID especializados. As *tags* armazenam um número de identificação (ID), enquanto um leitor recupera informações do ID num banco de dados pré-definido para atuar conforme o programado (identificação de produto/preço, localização etc). As *tags* RFID se classificam em duas categorias, ativas ou passivas, dependendo da sua fonte de energia elétrica. Uma *tag* ativa contém sua própria fonte de energia, usualmente uma bateria; uma *tag* passiva obtém energia do sinal de um leitor externo. Leitores RFID também são classificados como ativos e passivos, dependendo do tipo de *tag* lido por eles (WEINSTEIN, 2005).

2.1. Tags ativas

Contendo a sua própria fonte de energia, as *tags* ativas transmitem uma potência de sinal relativamente elevada em comparação com as tags passivas e os leitores tem acesso a elas a partir de distâncias maiores, da ordem de dezenas de metros. A alimentação interna faz com que elas se tornem maiores e mais caras em relação às *tags* passivas, fazendo com que sistemas com *tags* ativas funcionem melhor para localização em longas distâncias, com leituras entre 20 a 100 metros (WEINSTEIN, 2005). Entre as *tags* ativas, existem também as chamadas *tags* semiativas, que podem permanecer em *stand-by* (dormentes) até receber um sinal de um leitor e, então, começar a transmitir as informações. Graças à presença da alimentação interna, as *tags* ativas podem operar em altas freqüências, usualmente 455 MHz, 2,45 GHz ou 5,8 GHz, dependendo das necessidades de distância de leitura e de memória.

2.2. Tags passivas

As *tags* passivas são menores e mais baratas que as *tags* ativas, podendo armazenar em torno de 2 Kbites de informação. Quando uma *tag* entra dentro do alcance (range) do leitor, ela recebe um sinal eletromagnético do leitor através de

sua antena. A tag armazena a energia do sinal num capacitor, um processo chamado acoplamento indutivo. Quando o capacitor atinge uma carga mínima, ele pode alimentar o circuito da tag, que transmite um sinal modulado ao leitor. Esse sinal de retorno contém a informação armazenada na *tag*.

Tags passivas operam em freqüências de 128 KHz, 13.6 MHz, 915 MHz ou 2,45 GHz e têm *range* de leitura de alguns centímetros até cerca de 10 metros. A escolha da freqüência depende do ambiente do sistema, que materiais o sinal deve atravessar e o range necessário (WEINSTEIN, 2005).

3. ESTADO DA ARTE

Nos últimos anos, o avanço tecnológico tem proporcionado o desenvolvimento de novas ferramentas que auxiliam o dia-a-dia de deficientes visuais. Bússola (KUDO, FRIDMAN, *et al.*, 2011), leitor de código de barras (GREGORY, KIMURA, *et al.*, 2011), medidor de volume (MASSAROPPE e REBUCCI, 2011) são alguns exemplos de projetos voltados a deficientes visuais apoiados pela Poli Cidadã.

Paralelamente, também tem sido desenvolvidos projetos especificamente para o auxílio na locomoção de deficientes visuais. Em 2007, (BAL e A.S, 2007) desenvolveram um sistema, o *Path-finder*, para auxiliar esse grupo de usuários a chegar a um destino em locais desconhecidos. O *Path-finder* é composto por um aplicativo instalado num servidor central, *tags* RFID passivas para identificar os usuários, leitores de *tags* RFID espalhados pelo local a ser mapeado e conectados diretamente ao servidor, além de auto-falantes conectados aos leitores. Um destino é associado a uma *tag* no aplicativo e os leitores, ao localizar a *tag*, envia ao servidor a detecção do usuário. De acordo com as informações da posição sequencial, a direção é definida e informada ao usuário pelos auto-falantes conectados nos leitores (instalados nas calçadas do ambiente).

Figura 1: Componentes do SESAMONET SESAMONET (SESAMONET, 2007)

Pesquisadores portugueses desenvolveram um sistema comercial de guia por GPS integrado a um sistema de informação Bluetooth (Guio® Solid Step). O sistema consiste num aparelho GPS com transmissão Bluetooth carregado pelo usuário. Outros aparelhos Bluetooth são espalhados pelo ambiente com informações especificas sobre seu local exato, como tipo de estabelecimento, e ao entrarem no campo do transmissor carregado pelo usuário, transmitem ao mesmo a possibilidade de acessar essas informações (GUIO SYSTEMS).

Cientistas do *European Commission's Joint Research Centre* desenvolveram um protótipo de um sistema, o SESAMONET (*Secure and Safe Mobility Network*), que usa *tags* RFID fixados no chão para guiar deficientes visuais em áreas pré-definidas. Cada *tag* envia os sinais de posição através de bengalas especiais a um *smartphone* contendo informações do local e gravações de voz guiam o usuário ao local de destino. Em parceria com autoridades da prefeitura de Laveno (Itália), foram instaladas *tags* num caminho de 2 km ao longo de um parque, iniciando-se da estação central da cidade (SESAMONET, 2007).

Estudantes da *Central Michigan University* desenvolveram um protótipo de bengala eletrônica inteligente baseada em RFID para ajudar deficientes visuais a evitar obstáculos no caminho. Esse sistema, batizado de *Smart Cane*, é composto por uma bengala, um microcontrolador (que funciona como cérebro do sistema), um leitor de *tags* passivas UHF, um teclado e um sensor ultrassônico. O leitor RFID e o microcontrolador são carregados em uma mochila e são responsáveis por orientar o caminho ao usuário, já o sensor ultrassônico é acoplado à bengala para detectar obstáculos. As *tags* RFID são fixadas no chão e servem como referência para a localização do usuário, e então, o microcontrolador informa a direção (direita ou esquerda) através de auto-falantes. Maiores detalhes sobre esse projeto pode ser encontrado em (O'CONNOR, 2009). O *Smart Cane* foi expandido por outros estudantes da *Central Michigan University*, sendo desenvolvido um cão-guia eletrônico, batizado de *Smart-Robot*. Um sistema GPS e um mini-robô foram agregados ao projeto, resultando num robô que faz a função de um cão-guia (YELAMARTHI, HAAS, *et al.*, 2010).

Figura 2: Usuário testando o Smart Cane (O'CONNOR, 2009)

4. ALTERNATIVAS

De maneira geral, para resolver o problema de guiar um deficiente visual, é necessário saber a sua localização (i.e., localizá-lo), fazer o roteamento ao destino e dar as instruções do caminho. Nessas tarefas, a localização do usuário é um ponto que se pode resolver de maneiras diferentes, cujos principais métodos serão explicados a seguir.

4.1. Localização por wi-fi

A técnica de localização por wifi se baseia em utilizar antenas específicas ou roteadores wi-fi para fazer a triangularização de sinais entre o ponto de acesso (emissor do sinal wi-fi) e os diversos roteadores.

Neste caso, também há utilização do sinal RSSI (*received signal strength identification)* para identificação entre os pontos de cruzamentos das retas originadas de cada roteador. E há uso de um computador para o processamento de dados e cálculos do algoritmo da triangularização.

Um ponto de acesso (*access* point) emite um sinal RSSI para todos os três (número mínimo de) roteadores (leitores).

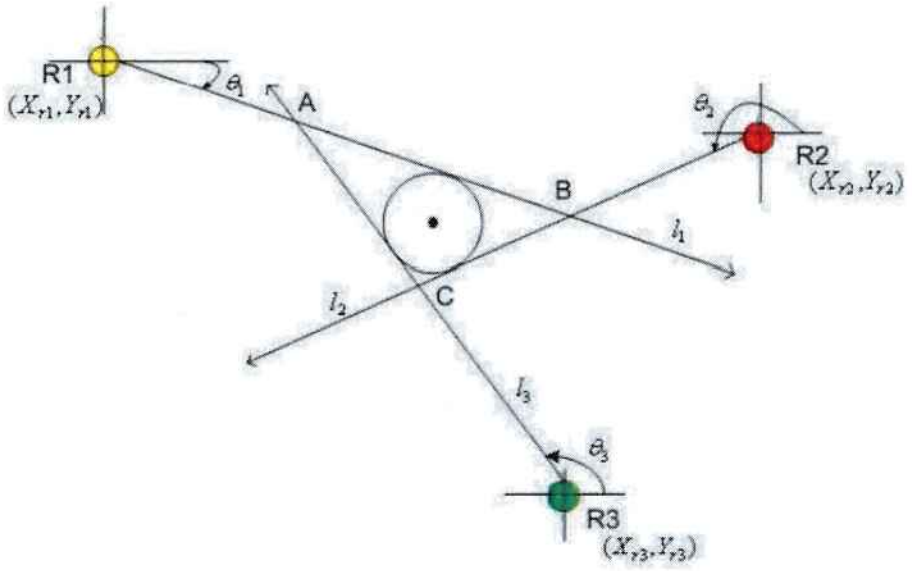


Figura 3: método da triangularização

Primeiramente, consideramos que cada roteador, R1, R2 e R3, possui sua coordenada (x, y) estabelecida em um referencial espacial único (elas são (Xr1,Yr1);

(Xr2,Yr2); (Xr3,Yr3), para os respectivos roteadores citados anteriormente). E seus eixos X (e Y também) são concorrentes entre si.

Obtemos as retas *l1*, *l2* e *l3*, para cada roteador, cada uma apresenta a seguinte expressão:

$$l1: y = -\tan \Theta 1 \,.\, x + Xr1.\tan \Theta 1 + Yr1$$

$$l2: y = \tan \Theta 2 \,.\, x - Xr2.\tan \Theta 2 + Yr2$$

$$l3: y = -\tan \Theta 3 \,.\, x + Xr3.\tan \Theta 3 + Yr3$$

Assim, podemos calcular os possíveis pontos de encontro entre 2 retas, por exemplo, entre *l1* e *l2,* pela fórmula e então, obtemos o ponto B:

$$x = \frac{Xr2.\tan \Theta 2 + Xr1.\tan \Theta 1 + Yr2 + Yr1}{\tan \Theta 2 + \tan \Theta 1}$$

E posteriormente, achamos o valor de y do ponto B.

Deste modo, obtemos também os pontos A e C.

O ponto de acesso (usuário) se encontra no incentro do triângulo ABC. Utilizando os pontos de intercepção entre as retas (A, B e C), os quais são os pontos de picos dos sinais RSSI lidos pelos roteadores, podemos finalmente calcular as coordenadas de A, B e C.

Os comprimentos dos três lados do triângulo são chamados de *a*, *b* e *c*. Portanto, pela fórmula abaixo, a coordenada do incentro do triângulo ABC (localização do usuário) pode ser finalmente obtida:

$$Xincenter = \frac{(b.x1 + c.x2 + a.x3)}{a + b + c}$$

$$Yincenter = \frac{(b.y1 + c.y2 + a.y3)}{a + b + c}$$

Onde (x1, y1) e (x2, y2) são os pontos A e B conectados pelo lado *a*; (x2, y2) e (x3, y3) são conectados pelo lado *b*; (x3, y3) e (x1, y1) são conectados pelo lado *c:*

$$a = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

$$b = \sqrt{(x3 - x2)^2 + (y3 - y2)^2}$$

$$c = \sqrt{(x1 - x3)^2 + (y1 - y3)^2}$$

Podemos notar que o método da triangularização exige um processamento rápido (um computador) para manter sempre a coordenada do usuário atualizada, devido a sua complexidade e aquisição e processamento de dados dos sinais RSSI pelos roteadores.

4.2. Localização espacial com *tags* ativas

Etiquetas ativas podem ser utilizadas para implementar um Sistema de Posicionamento em Tempo Real. Para implementá-lo, são necessários um leitor ativo (*active reader*) e etiquetas ativas (*active tags*), sendo que ambos devem possuir a propriedade RSSI.

A propriedade RSSI[1] (*received signal strength indication*) permite que o leitor obtenha uma informação da potência recebida (nível de força do sinal) de cada etiqueta em seu alcance.

Com o objetivo estimar a distância entre o leitor e a etiqueta, podemos traçar a curva sinal RSSI por distância (em metros), testando algumas das etiquetas ativas. O valor desse sinal recebido permitirá o cálculo da coordenada (x, y) do usuário a partir do método da trilateração, o qual será explicado adiante.

As etiquetas ativas devem ser distribuídas e calibradas com sua coordenada (x, y) dentro de um referencial espacial único, em um ambiente que se deseja mapear (a faculdade ou um *shopping center* por exemplo). O sinal RSSI emitido por cada etiqueta também deve ser testado, afim de não haver perdas da precisão na leitura deste sinal.

Cada etiqueta apresenta uma identificação única (*tag ID*) pré-definida pelo fabricante (normalmente). Após a calibração descrita anteriormente, a identificação

[1] RSSI: *Received Signal Strength Indication* é um termo utilizado em telecomunicações que se refere a uma medida presente num sinal recebido na faixa de radiofreqüência

de cada *tag ID* está associada a sua coordenada (x, y) num banco de dados (o qual também contém o mapa do local de interesse e a curva do sinal RSSI por distância).

O posicionamento dessas etiquetas proverá a localização do usuário com boa precisão pelo método da multilateração, junto com a leitura do sinal RSSI. Neste caso a trilateração é suficiente para calcular a coordenada do usuário, utilizando pelo menos três etiquetas e os sinais RSSI de cada uma delas.

Considerando um ambiente térreo, cada etiqueta (representada por BS1, BS2 e BS3) possui um par de coordenada definida (x ,y) em uma referência única (após serem calibradas). No ponto A, está localizado o usuário, a distância entre o ponto A e BS1, BS2 e BS3 é calculada pelo sinal RSSI das respectivas etiquetas. Assim, sabendo essas distâncias e as coordenadas dos pontos BS1, BS2 e BS3, a coordenada do ponto A é calculada pelo algoritmo da trilateração.

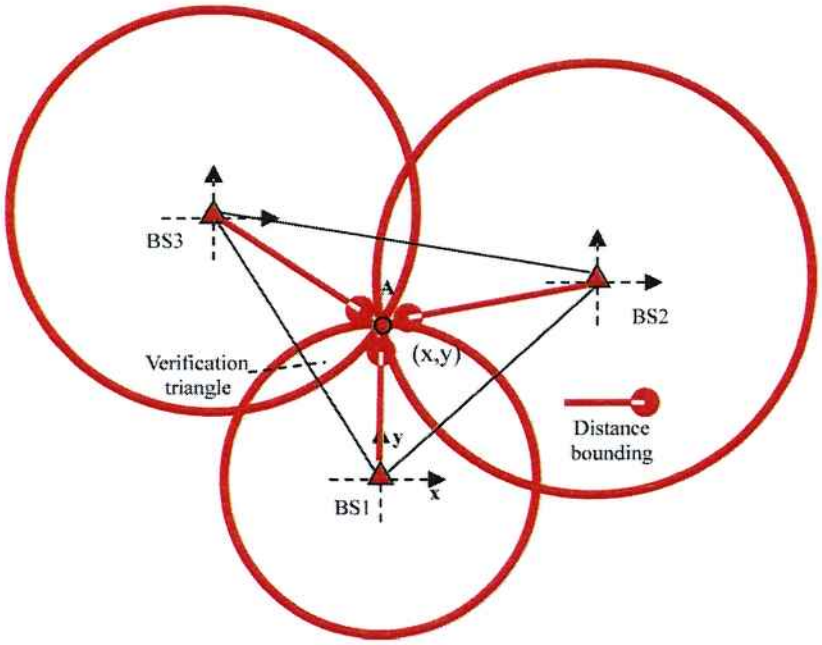


Figura 4: Esquema para a obtenção da localização na Trilateração (SWISS NATIONAL SCIENCE FOUNDATION, 2005)

As dificuldades de implementar esse tipo de solução são: traçar experimentalmente a curva RSSI por distância para etiqueta ativa (essa curva deve ser plotada para um número grande de etiquetas ativas, afim de obter uma média estatística do comportamento da curva); mapear um local, ou seja, disponibilizar as etiquetas de maneira eficiente para facilitar o máximo possível a coleta de sinais

RSSI[2]; e encontrar um fornecedor confiável de etiquetas ativas e leitores que suportem a propriedade RSSI (já foram encontrados vendedores chineses, porém, não há confiabilidade dos produtos de acordo com as pesquisas realizadas).

4.3. Localização utilizando *tags* passivas

A utilização de etiquetas passivas com o objetivo de localizar o usuário se baseia basicamente em um mapeamento eficiente do local, pois o alcance (*range*) dessas etiquetas é pequeno (usualmente menor que 50 centímetros).

Este método não está descrito em nenhum lugar na literatura: mapeamento com *passive tags*, e utilizá-las para fazer um roteamento são tarefas complexas.

O mapeamento do ambiente funciona da seguinte maneira: a disponibilidade e distribuição das etiquetas apresentam algum tipo de geometria (por exemplo, um caminho reto).

A posição de cada etiqueta está referenciada num banco de dados com sua coordenada (x, y), de acordo com sua identificação (*ID* – cada etiqueta apresenta um *ID* único), e utiliza-se um referencial espacial único.

A idéia é saber a posição do usuário quando ele entrar no *range* da etiqueta (excitada pelo leitor – *reader*), como este *range* é pequeno, podemos estimar sua coordenada de acordo com a última etiqueta lida, e a leitura da próxima etiqueta.

A simplicidade deste método, porém, requer um sistema de roteamento inteligente e complexo. Neste caso, não há algoritmos específicos criados para tal roteamento proposto.

Devemos ressaltar a variedade de tipos de etiquetas passivas de acordo com sua faixa de frequência de operação e alcance de transmissão: essas faixas são divididas em baixa (LF), alta (HF) e ultra alta frequência (UHF).

A faixa de baixa frequência está em torno de centenas de kHz (normalmente 125 e 134.2 kHz, frequência de leitores e etiquetas neste caso).

---

[2] RSSI já foi utilizado para estimar a distância entre dois pontos de maneira não muito precisa (ver link de artigos na bibliografia), porém, com mais de dois pontos, podemos estimar com melhor precisão a coordenada de um leitor (em nosso caso)

Utilizar etiquetas e leitores LF implica em baixo custo (do leitor e etiquetas), porém menor alcance de leitura (cerca de poucos centímetros, normalmente menor que 5 centímetros de *range*). Assim, há limitações como o tempo de resposta da etiqueta, pois ao entrar no alcance do leitor, existe um período de tempo para que a etiqueta seja excitada e envie as informações para o leitor Rfid.

As vantagens de utilizar a faixa LF são: um alto número de etiquetas permite mapear lugares não muito amplos, pois o alcance de leitura limitado faz com o número de etiquetas aumente linearmente (deve-se utilizar mais do que uma etiqueta para localizar um ponto no espaço). Uma limitação é a aplicação em apenas 2 dimensões.

Seguem abaixo imagens de um leitor e de etiquetas para a faixa LF:

Figura 5: Exemplo de módulo RFID 125 kHz (ITead Studio)

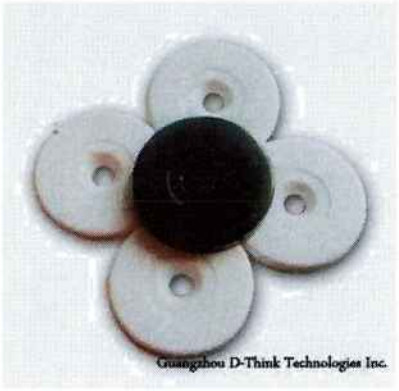


Figura 6: Exemplo de *tags* RFID 125 kHz (RFID Shop, 2012)

Para a faixa HF (entre 3 a 27 MHz), há um custo maior (o leitor é pelo menos quatro vezes mais caro, e a etiqueta é duas vezes mais cara, em relação as de baixa frequência). O alcance de leitura para essa faixa é de 5 a 12 centímetros, assim é possível mapear lugares maiores e utilizar apenas uma etiqueta para localização de um ponto no espaço (ainda há limitação de 2 dimensões para

aplicação nessa faixa de frequência), ou seja, o tempo de resposta entre o leitor e a etiqueta é maior devido ao alcance de leitura (quando comparado à faixa LF).

Seguem abaixo imagens de um leitor e de etiquetas para a faixa HF:

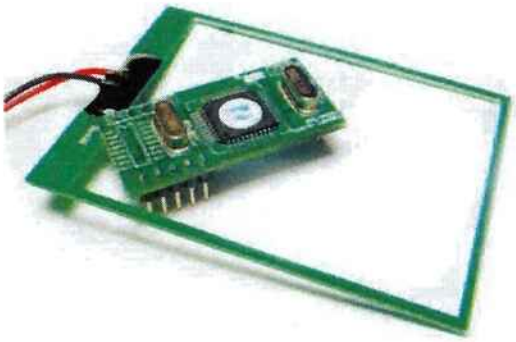


Figura 7: Exemplo de módulo RFID 13,56 MHz (ITead Studio)

Figura 8: Exemplo de *tags* RFID 13,56 MHz (RFID Shop)

Para a faixa de frequência UHF (entre 433 Mhz a 6 Ghz), há um custo muito alto para o leitor e a etiqueta (de 10 a 20 vezes mais caro em relação aos componentes de LF), porém o alcance de leitura pode chegar a dezenas de metros (de acordo com alguns fabricantes). Para essa faixa de frequência é possível fazer localização em 3 dimensões por triangularização de sinais.

Seguem abaixo imagens de um leitor e de etiquetas para a faixa UHF:

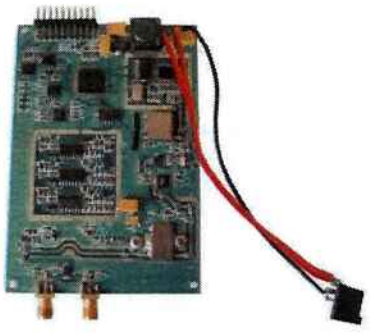


Figura 9: Exemplo de módulo RFID UHF passivo (Made-in-china.com)

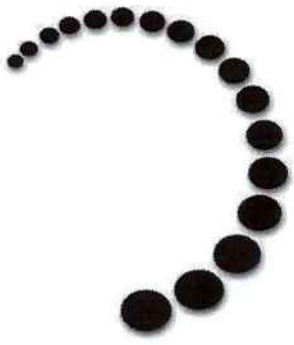


Figura 10: Exemplo de tags RFID UHF passivas (Made-in-china.com)

5. METODOLOGIA

Como visto nos sistemas atuais no estado da arte, já existem vários projetos na mesma linha do projeto aqui apresentado, porém nenhum deles analisa a fundo o sistema em ambientes fechados, onde quase nunca é possível a utilização de GPS, por exemplo, e que é exatamente o tema central de nossa pesquisa. Baseamo-nos assim no que os sistemas atuais utilizam como transmissão secundária (RFID, Bluetooth, ultrassom etc.), mas com o foco de transformá-las no sistema principal de localização e transmissão de dados do nosso projeto.

5.1. Concepção

Como dito, o atual projeto propõe o desenvolvimento de um sistema eletrônico capaz de guiar deficientes visuais em locais fechados (tais como hospitais, shopping centers, escolas, etc) utilizando tecnologia de identificação por radiofreqüência (RFID) (ZHOU e SHI, 2009).

Para realizar tal tarefa, o projeto consistirá no desenvolvimento e integração de três subsistemas:

- Módulo do leitor
- Banco de dados
- Sistema de roteamento
- Interface com o usuário

5.1.1. Módulo do leitor

O módulo do leitor é responsável por emitir o sinal de radiofreqüência no ambiente e receber o sinal de resposta das *tags* localizadas dentro do range do leitor. Ele ainda será responsável por transmitir a informação das *tags* ao *smartphone*, que fará o devido tratamento para determinar a localização do usuário.

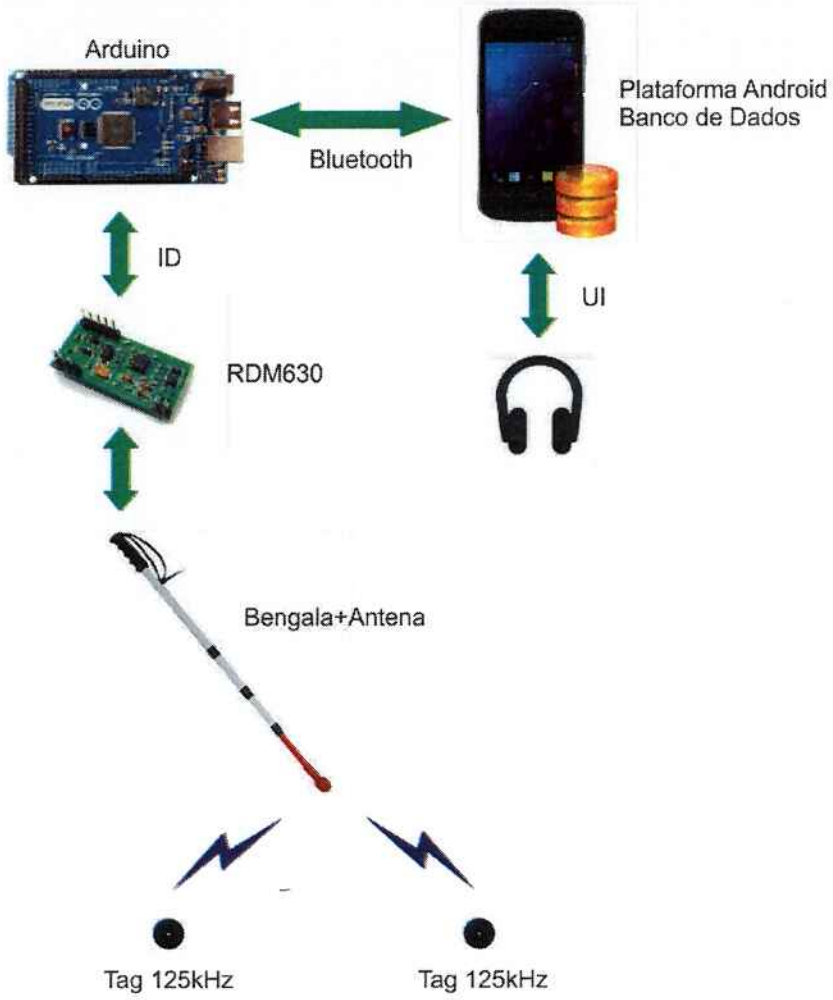


Figura 11: Princípio de funcionamento do sistema

5.1.2. Banco de dados

O banco de dados irá armazenar as informações de todas as *tags* instaladas no mapeamento do local. Além de conter os números de identificação das *tags* (*IDs*), também conterá a informação de suas respectivas posições espaciais e informações pertinentes sobre o local em que elas estão (cruzamento, fim de curso, porta, etc.). O banco de dados será armazenado no *smartphone*.

5.1.3. Sistema de Roteamento

O sistema de roteamento consiste na implementação de um software para a determinação do caminho que o usuário deverá seguir para chegar ao destino desejado. O software deverá receber o *ID* da *tag*, verificar a posição correspondente no banco de dados e calcular a melhor rota do ponto atual até o destino.

5.1.4. Interface com o usuário (UI)

Através desse subsistema, o deficiente visual poderá informar o local de destino e receberá as informações da rota a ser seguida. Essa interface será desenvolvida para *smartphones* com plataforma *Android* devido à simplicidade no desenvolvimento de aplicações com reconhecimento e síntese de voz.

5.1.5. Descrição do mapeamento

Além desses subsistemas, ainda é necessário distribuir as *tags* no ambiente a ser mapeado, importante para a geração do banco de dados e a abordagem do algoritmo de roteamento.

As *tags* serão implantadas no chão, principalmente através das trilhas de piso tátil (relevo que sinaliza o caminho preferencial para os deficientes visuais, conforme ilustrado na Figura 12). Fora dessas trilhas, devem ser colocadas *tags* para cada local de destino. Dessa forma, o alcance de leitura das *tags* não limita a densidade de sua distribuição, já que ao detectar uma *tag*, o sistema de roteamento saberá qual deve ser a próxima *tag* e o usuário sabe por onde caminhar. Isso permite haver longos trechos sem a leitura de *tags*, caso necessário.

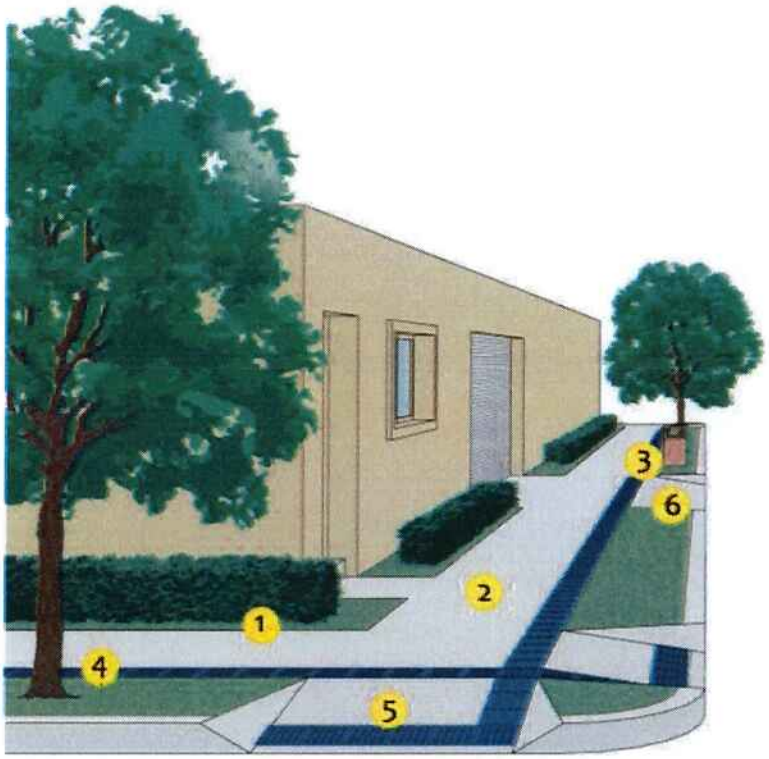


Figura 12: modelo de um caminho seguro para deficientes visuais (CASSOL CENTER LAR)

6. IMPLEMENTAÇÃO

Os subsistemas descritos anteriormente serão implementados separadamente, conforme descrito a seguir:

i) Implementação do módulo do leitor: integração do módulo RDM630 (leitor de *tags* RFID 125kHz) com o Arduino através da conexão das saídas desse módulo (utilizando a interface TTL RS232) nas entradas digitais do Arduino (portas UART). Uma PCB (*printed circuit board*) deverá ser projetada para realizar essa integração mecânica e elétrica entre o RDM630 e o Arduino. Será desenvolvido o programa que gerenciará a recepção dos dados do módulo no microcontrolador e enviará via USB ao *smartphone*.

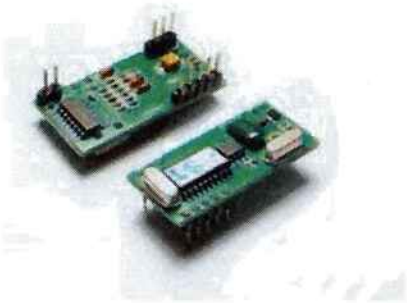


Figura 13: Módulo RDM630

Figura 14: Arduino MEGA ADK

ii) Implementação do banco de dados: após as *tags* terem sido posicionadas no ambiente, o banco de dados será criado no *SQLite* e gravado no cartão de memória do *smartphone*.

iii) Implementação do sistema de roteamento: o algoritmo de roteamento será programado em Java utilizando o kit de desenvolvimento de software para Android (Android SDK).

iv) Implementação da Interface com o Usuário: a aplicação para interagir com o usuário também será desenvolvido em Java no Android SDK. Será utilizado a API *TextToSpeech* para a geração da fala com as instruções de roteamento. O local de destino será informado utilizando a API de reconhecimento de voz do Android.

A arquitetura do aplicativo que será desenvolvido para o Android está ilustrada na Figura 15.

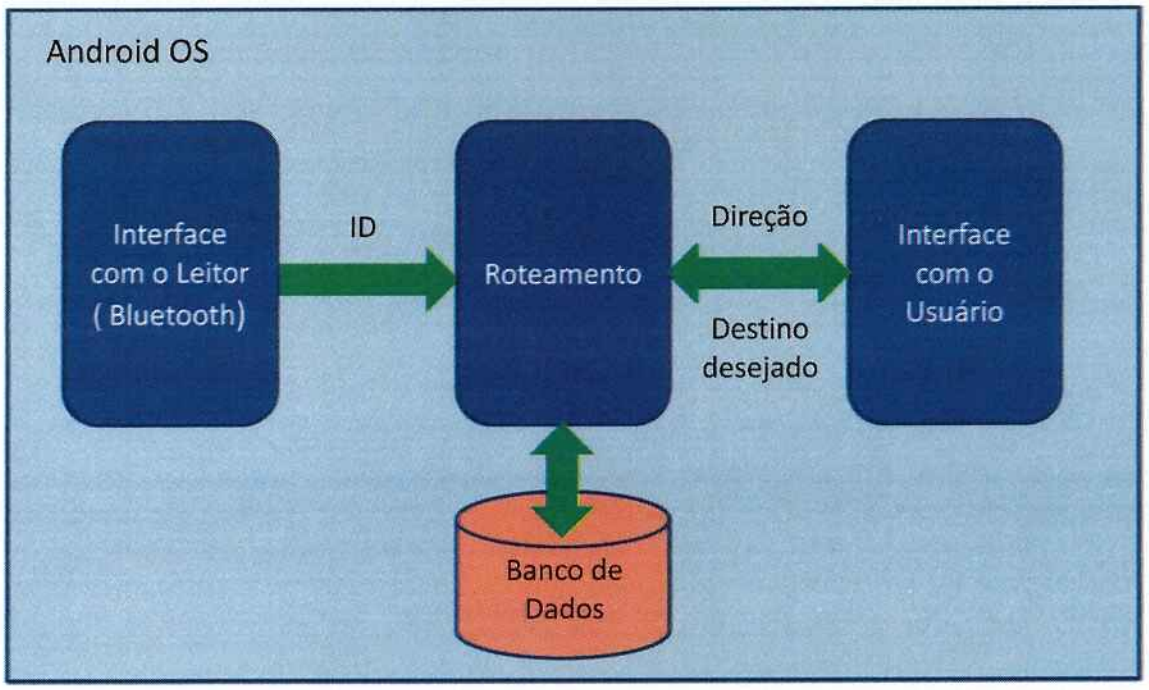


Figura 15: Estrutura do *Software* na plataforma Android

6.1. Implementação do módulo leitor

A implementação do módulo leitor divide-se em duas tarefas, leitura das *tags* e transmissão dos dados ao *smartphone*, descritas a seguir.

6.1.1. Leitura das tags

As *tags* são lidas utilizando o módulo RDM6300, projetado para a leitura dos *IDs* de *tags* RFID passivas de 125 kHz. Ao se aproximar uma *tag* de sua antena, esse módulo decodifica o sinal de rádio freqüência recebido da *tag* e o envia serialmente através de sua interface UART.

Portanto, do ponto de vista do microcontrolador, o módulo RDM6300 é visto simplesmente como um transmissor e receptor de dados seriais. Para isso, é utilizado uma das portas seriais (TX e RX) do Arduino e os *IDs* são lidos no momento em que é sinalizado que dados estão disponíveis nessa porta.

O módulo RDM6300 e o Arduino foram conectados de acordo com a Tabela 1. A Figura 16 ilustra essas conexões.

Tabela 1: Conexões entre o Arduino e o módulo RDM6300

	Conexões	
	Arduino	**RDM6300**
Pinos	PIN 17 (RX2)	P1-PIN1 (TX)
	PIN 16 (TX2)	P1-PIN2 (RX)
	GND	P1-PIN4 (GND)
	5V	P1-PIN5 (+5V)
	5V	P3-PIN2 (+5V)
	GND	P3-PIN3 (GND)

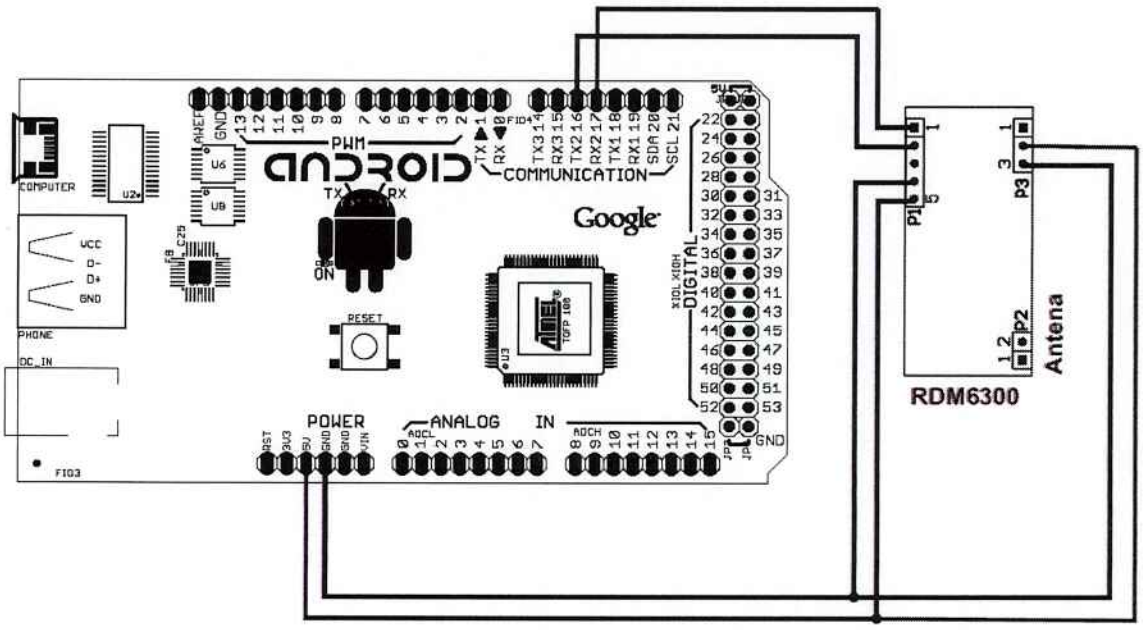


Figura 16: Esquema da conexões entre o Arduino e o módulo RM6300

Os dados são enviados via serial a uma taxa de 9600 bps (*baud rate*) seguindo a seguinte sequência: "*start of text*" (ASCII 0x02), sequência de 10 caracteres ASCII indicando o *ID*, 2 caracteres ASCII para paridade (*checksum*) e "*end of text*" (ASCII 0x03). A Figura 17 ilustra o pacote de dados enviado pelo módulo RDM6300 na leitura de uma *tag* conforme descrito no parágrafo anterior.

02	**10ASCII Data Characters**	Checksum	03

Figura 17: Formato dos dados enviado pelo módulo RDM6300 (RDM630 Specification)

Dessa forma, o microcontrolador permanece sempre a espera de dados em sua porta serial 2 e, se um dado lido for igual ao ASCII 0x02, os dados lido em seguida são armazenados em uma cadeia de caracteres para a composição do *ID*.

Quando o ASCII 0x03 é lido, a cadeia de caracteres é fechada, sendo que os dois últimos caracteres representam a paridade do *ID*.

A paridade dos 10 caracteres ASCII é gerada através do ou-exclusivo lógico (XOR) dos 5 pares de caracteres do *ID*. Por exemplo, para o *ID* 0x62E3086CED, os cinco pares são 0x62, 0xE3, 0x08, 0x6C, 0xED e a paridade é então calculada (RDM630 Specification):

```
Paridade = 0x62 XOR 0xE3 XOR 0x08 XOR 0x6C XOR 0xED = 0x08
```

Ao terminar de ler os dados de uma *tag*, é executada uma rotina no Arduino para verificar se a paridade recebida está de acordo com a paridade calculada com os cinco primeiros pares de dados, conforme exemplificado acima. Se os valores da paridade calculada e da paridade recebida via serial não forem iguais, o valor do *ID* recebido é desconsiderado.

6.1.2. Envio dos dados para o *smartphone*

Quando o Arduino recebe o valor de um ID com sucesso (paridade consistente), esse valor deve ser enviado ao *smartphone* para que as rotinas de roteamento sejam executadas. Para estabelecer uma conexão de comunicação entre o *smartphone* e o Arduino, é utilizado o módulo Bluetooth Mate Silver (**Erro! Fonte de eferência não encontrada.**) da Sparkfun, que utiliza o módulo Bluetooth RN-42 e simplifica a sua utilização em placas SMT.

A comunicação com o módulo Bluetooth é feita através de sua interface serial (RX/TX), o que viabiliza uma interconexão direta com o Arduino. As conexões entre o módulo Bluetooth e o Arduino são descritas na Tabela 2 e ilustradas na Figura 18.

Tabela 2: Conexões entre o Arduino e o módulo Bluetooth

	Conexões	
	Arduino	**Módulo Bluetooth**
Pinos	PIN 15 (RX3)	TX-0
	PIN 14 (TX3)	RX-0
	5V	VCC
	GND	GND

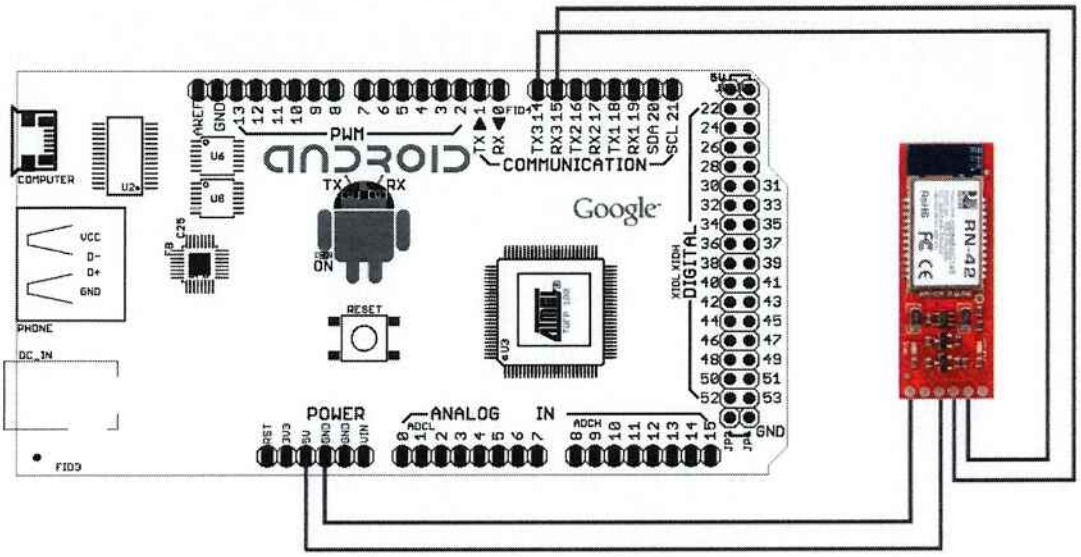


Figura 18: Esquema das conexões entre o módulo Bluetooth e o Arduino

A princípio, quando o valor do ID de uma tag é lida na porta serial do módulo RFID (RDM6300), bastaria apenas enviar esses dados para a porta serial utilizada pelo módulo Bluetooth. No entanto, sempre que uma *tag* está no alcance de leitura da antena, o módulo RFID permanece enviando o *ID* a uma taxa de aproximadamente 5 *IDs* por segundo, o que faz com que o mesmo *ID* seja lido pelo Arduino várias vezes. Para evitar que seja enviado o mesmo *ID* repetidamente ao smartphone, é implementado nesse projeto um buffer que armazena um número N de *IDs* enviados ao *smartphone*, de forma que o Arduino somente enviará os *IDs* que não estiverem nesse buffer (portanto, ao ler um *ID* que já foi enviado ao smartphone, esse *ID* só será enviado novamente depois que for lido outros N *IDs*).

Antes de se enviar os dados do ID para o módulo Bluetooth, o valor do *ID* é encapsulado de forma que o software no Android possa saber quando o valor recebido via Bluetooth se inicia e se encerra e que esse valor corresponde a um *ID*. É utilizado o caractere "<" para sinalizar que se está iniciando uma transmissão de valores e o caractere ">" para sinalizar que se está encerrando a sequência de valores. Para sinalizar que os valores correspondem ao *ID* de uma *tag* RFID, é utilizado o caractere "`i`" após a sinalização de inicio dos dados. Por exemplo, o *ID* `62E3086CED08` é transmitido como `<i62E3086CED>`.

6.2. Implementação do banco de dados

Todas as *tags* instaladas no mapeamento do local estarão contidas no banco de dados, fornecendo as informações pertinentes para o traçado da trajetória. Além de conter os números de identificação das *tags* (*IDs*), também conterá a informação de suas respectivas posições espaciais e informações pertinentes sobre o local em que elas estão (cruzamento, fim de curso, porta, etc.). O banco de dados será montado utilizando o *software SQLite Expert Personal* (*freeware*) e será armazenado no *smartphone*. O Android possui uma API que suporta a leitura e/ou gravação do arquivo no formato gerado pelo *SQLite*.

Para maior simplicidade e eficiência no acesso e armazenamento dos dados, foram implementados três tabelas no banco de dados: `tagGrupo`, `grupoInfos` e `destinos`. Visto que pode haver diversas *tags* referenciando o mesmo local, as tabelas `tagGrupo` e `grupoInfos` foram implementadas para que se evitasse o armazenamento de dados redundante de forma que os diferentes IDs correspondentes a um mesmo local no mapeamento são associados ao seu grupo (um número posivito).

Na tabela `tagGrupo`, as linhas contém os seguintes campos:

- `RecNo`: índex automático para identificação da linha;
- `tagID`: representa o *ID* de uma *tag* utilizada no mapeamento do local;
- `grupo`: representa o número do grupo a qual a *tag* da linha atual pertence.

RecNo	tagID	grupo
1	1000000001	1
2	1000000002	1
3	1000000003	1
4	1000000004	1
5	1000000005	1
6	1000000006	2
7	1000000007	2
8	1000000008	2
9	1000000009	2
10	1000000010	2
11	1000000011	3
12	1000000012	3
13	1000000013	3
14	1000000014	3
15	1000000015	3

Figura 19: tabela `tagGrupo` do banco de dados

A tabela grupoInfos contém os seguintes campos:

- `RecNo`: assim como na tabela anterior, é um índex automático para identificação da linha;
- `grupo`: informa o grupo ao qual as informações das colunas seguintes estão relacionadas;
- `pesoN`: informa o peso (distância em metros) do grupo atual até o grupo posicionado a norte;
- `pesoS`: informa o peso (distância em metros) do grupo atual até o grupo posicionado a sul;
- `pesoL`: informa o peso (distância em metros) do grupo atual até o grupo posicionado a leste;
- `pesoO`: informa o peso (distância em metros) do grupo atual até o grupo posicionado a oeste;
- `grupoN`: informa qual o grupo está posicionado a norte;
- `grupoS`: informa qual o grupo está posicionado a sul;
- `grupoL`: informa qual o grupo está posicionado a leste;
- `grupoO`: informa qual o grupo está posicionado a oeste;
- `localN`: informa o nome do local posicionado ao norte do grupo atual;
- `localS`: informa o nome do local posicionado ao sul do grupo atual;
- `localL`: informa o nome do local posicionado ao leste do grupo atual;
- `localO`: informa o nome do local posicionado ao oeste do grupo atual;
- `info`: registra, caso houver, alguma informação pertinente ao local do grupo atual, como presença de escada, fim do piso tátil.

Com essas duas tabelas no banco de dados, uma contendo os vários *IDs* que um grupo pode conter e outra contendo as informações desses grupos, evita-se, então, a repetição das informações de um grupo para os diferentes *IDs*.

Para ilustrar a construção de um banco de dados, a Figura 20 mostra um mapeamento de um trecho de uma galeria de compras no qual contém sete lojas e 4 grupos de *tags*. A tabela `grupoInfos` referente a esse mapeamento é apresentada na Figura 21.

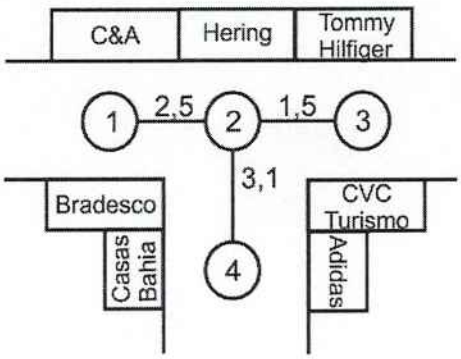


Figura 20: exemplo de de mapeamento e suas informações

RecNo	grupo	pesoN	pesoS	pesoL	pesoO	grupoN	grupoS	grupoL	grupoO	localN	localS	localL	localO	info
1	1	<null>	<null>	2.5	<null>	<null>	<null>	2	<null>	C&A	Bradesco	<null>	<null>	<null>
2	2	<null>	3.1	1.5	2.5	<null>	4	3	1	Hering	<null>	<null>	<null>	<null>
3	3	<null>	<null>	<null>	1.5	<null>	<null>	<null>	2	Tommy Hilfiger	CVC Turismo	<null>	<null>	<null>
4	4	3.1	<null>	<null>	<null>	2	<null>	<null>	<null>	<null>	<null>	Adidas	Casas Bahia	<null>

Figura 21: tabela `grupoInfos` do banco de dados

A tabela `destinos` tem como objetivo criar variações nos nomes dos destinos que serão reconhecidos pelo sistema de reconhecimento de voz e reduzir a probabilidade de falha na detecção do destino. Essa tabela possui os seguintes campos:

- `nome`: informa o nome oficial do destino;
- `var1`: informa uma primeira variação para o nome;
- `var2`: informa uma segunda variação para o nome;
- `var3`: informa uma terceira variação para o nome;
- `tts`: informa o texto que produzirá a síntese de voz do nome de forma mais fiel;
- `grupo`: informa o grupo de *tags* ao qual o destino está associado;
- `dir`: informa a direção que o destino está posicionado em relação ao grupo de *tags*.

RecNo	nome	var1	var2	var3	tts	grupo	dir
1	AA	A A	tam	ar	A A	1	N
2	AB	A B	a b c	ac dc	A B	2	N
3	AC	A C	hace	asse	A C	3	N
4	AN	A N	tam m	capemi	A N	5	S
5	CF	C F	cpf	se f	C F	30	O
6	Polícia Federal	Policia federal	policia federal 1	policia federal ce	Polícia Federal	15	N
7	Hering	having	ele	herring	ééringui	16	S
8	Fatto a Mano	fato a mano	fato a manno	fattoamano	fàto a mãno	10	L

Figura 22: Tabela `destinos`

Os campos var1, var2 e var2 têm como objetivo informar variações para o nome do destino de contendo valores prováveis de resultados do reconhecimento de voz para o destino em questão.

6.3. Reconhecimento do local de destino

O destino é detectado através do reconhecimento da fala do usuário. Para isso, é chamada a aplicação de reconhecimento de voz presente no sistema operacional Android através do `RecognizerIntent`. Essa aplicação grava o trecho da fala do usuário, o resultado do reconhecimento da fala provém do servidor da aplicação, portanto, é necessário estar conectado à internet para que esse recurso funcione. Ao fim da execução dessa aplicação, uma lista de `Strings` é recebida como resultados do reconhecimento da fala, sendo que cada `String` é uma possível palavra ou frase dita pelo usuário. Essa lista é utilizada como parâmetro da função `achaDestino(ArrayList<String>)` da classe `dbWrapper`, que busca o destino no banco de dados do mapeamento (na tabela `destinos` do banco de dados).

A Figura 23 e a Figura 24 mostram uma implementação de um aplicativo teste para visualização gráfica dos resultados do reconhecimento de voz. Inicialmente é exibida uma caixa de diálogo solicitando que o usuário fale o que deseja ser reconhecido. A Figura 24 mostra a lista de palavras contendo os resultados do reconhecimento e, após o campo "Grupo destino", apresenta o número do grupo do destino no banco de dados obtido pela função `achaDestino(ArrayList<String>)` da classe `dbWrapper`.

Figura 23: Aplicação para reconhecimento de fala

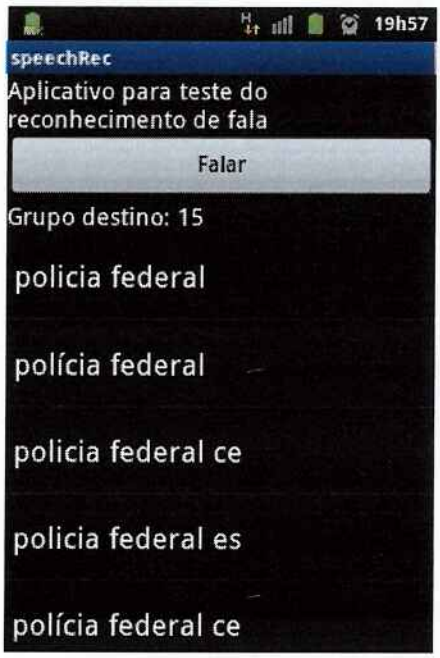


Figura 24: Resultado do reconhecimento de fala

6.4. Acesso ao banco de dados

O banco de dados gerado pelo *SQLite Expert Personal* é acessado utilizando as rotinas da classe `SQLiteDatabase` do API de banco de dados do Android. Essa classe fornece funções para criar, excluir, executar comandos SQL, assim como outras tarefas comuns de banco de dados. No entanto, a utilização direta dessas funções necessita a passagem de diversos parâmetros e, por outro lado, esse projeto necessita poucos recursos da API (porém, com freqüência). Portanto, para facilitar a manipulação do banco de dados, foi criado a classe `dbWrapper`, que abre o banco de dados e que contém funções que auxiliam na busca das informações, além de gerar o grafo do mapeamento do ambiente a partir das informações da tabela `grupoInfos`.

Foi criada a classe `grupoInfo` para encapsular todas as informações da tabela `grupoInfos` de um determinado grupo. O diagrama de classes do `dbWrapper` está representado na Figura 25.

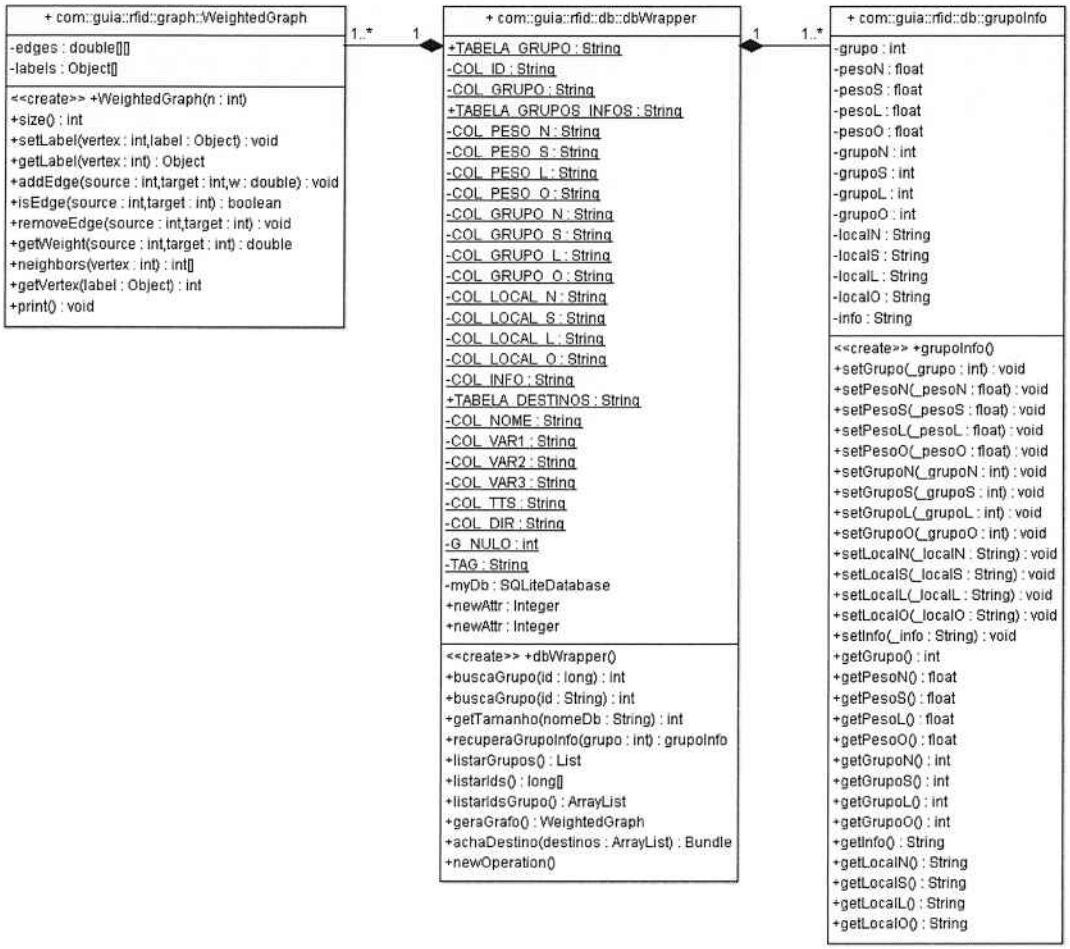


Figura 25: diagrama de classes do dbWrapper

6.5. Determinação do melhor caminho

Uma das questões fundamentais nesse projeto é a determinação do melhor caminho entre a posição atual do usuário até o destino desejado. A posição atual do usuário é determinada através da detecção das tags pelo caminho, ou seja, através da leitura dos IDs das tags; o destino desejado é informado previamente pelo usuário por voz.

Para resolver o problema da determinação do melhor caminho, o mapeamento do local é interpretado como um grafo com pesos. Cada vértice do grafo é gerado a partir dos grupos da tabela grupoInfos do banco de dados e os pesos são definidos pelas distâncias entre os grupos (colunas pesoN, pesoS, pesoL e pesoO da tabela grupoInfos). Distância será o critério de escolha do melhor caminho, portanto, com o grafo gerado a partir apenas dos grupos e distâncias (pesos) do mapeamento, é

possível determinar o melhor caminho até o destino (ou seja, o caminho de menor distância). Com o grafo em mãos e sabendo-se o grupo referente à posição do usuário e o grupo referente ao destino, o melhor caminho é determinado através do clássico algoritmo de Dijkstra (Apêndice I).

6.6. Sistema de roteamento

O sistema de roteamento possui a função de guiar o usuário (dar instruções por áudio, e assim orientar o percurso da rota correta), à medida que ele se desloque pela trilha de piso tátil.

6.6.1. Variáveis importantes

O algoritmo de roteamento utiliza variáveis importantes para o seu funcionamento: no_ant (grupo lido anteriormente de referência para o usuário), no_atual (grupo de referência para o usuário), no_novo (grupo lido atual), no_prox (próximo grupo do vetor caminho[ ]), classe direção (armazena o sentido de locomoção do usuário) e o vetor caminho[ ] de grupos (menor caminho) dado pelo Algoritmo de Dijkstra.

6.6.2. Funções

O algoritmo de roteamento faz uso de diversas funções:

- Acesso ao banco de dados;
- Algoritmo de Dikstra;
- *TextToSpeech*;
- achaDireção;
- daInstrução.

O acesso ao banco de dados e o Algoritmo de Dijkstra já foram explicados anteriormente, e possuem grande importância no algoritmo de roteamento.

O *TextToSpeech* possui a função de dar instruções ao usuário por uma interface de áudio.

A função achaDireção utiliza como entrada os grupos no_ant e no_atual, e o acesso ao banco de dados para achar o sentido do usuário. De maneira recursiva, a função calcula e retorna o sentido (Norte, Sul, Leste ou Oeste) de locomoção do usuário, atualizando a variável direção.

A função daInstrução utiliza o *TextToSpeech* e o acesso ao banco de dados para passar as diversas instruções possíveis ao usuário (no banco de dados temos as direções relativas entre cada grupo: norte, sul, leste e oeste). Suas entradas são: no_ant (grupo anterior), no_atual (grupo atual), destino (local de destino escolhido pelo usuário), direção (sentido do usuário) e o vetor de grupos dado pelo Algoritmo de Dikstra. E a partir disso, essa função verifica o sentido do usuário, o vetor de grupos e as direções relativas a cada grupo; assim é possível calcular para onde o usuário deve ir, e retorna as possíveis mensagens de voz (instruções), como: "Siga em frente", "Vire à esquerda", "Vire à direita", "O destino se encontra a sua esquerda/direita", e diversas outras mensagens que ainda serão listadas com o andamento do projeto, como mensagens de emergência, serviços (banheiro), entre outras.

6.6.3. Cenários e análises

A rotina do algoritmo de roteamento se baseia na análise de três possíveis cenários de ocorrência:

- O primeiro cenário: a rota que o usuário percorre é a mesma rota dada pelo Algoritmo de Dijkstra, ou seja, o grupo lido confere com o grupo da menor rota calculada por Dijkstra. Neste caso, as instruções dadas orientam o usuário até seu destino. E em adição a isso, foi implementado uma correção inteligente do sistema em que se o grupo lido for diferente do no_prox, porém ele está contido em caminho[ ], o roteamento redireciona o usuário e atualiza sua posição, dando novas instruções ;
- As duas condições para o segundo cenário: a rota que o usuário percorre difere da rota dada pelo Algoritmo de Dijkstra e o grupo lido pelo leitor é o mesmo grupo lido anteriormente, ou seja, o usuário ficou parado ou pode ter ido e voltado (e deste modo, não há como saber o sentido de locomoção do usuário). Neste caso, a instrução dada ao usuário é: "Ande pela trilha." (até

que um novo grupo de *tags* seja lido, e assim pode ser calculado o sentido de locomoção do usuário).

- As duas condições para o terceiro cenário: a rota que o usuário percorre difere da rota dada pelo Algoritmo de Dijkstra e o grupo lido pelo leitor é diferente do lido anteriormente, ou seja, o caminho percorrido pelo usuário difere da rota dada inicialmente por Dijkstra, porém agora sabemos o sentido de locomoção do usuário. Neste caso, após achar o sentido do usuário, o Algoritmo de Dijkstra é chamado novamente para a correção de rota. Assim uma nova rota é calculada, e novas instruções são dadas ao usuário.

6.6.4. Funcionamento do algoritmo de roteamento

Inicialmente, o usuário escolhe o destino e o Algoritmo de Dijkstra retorna a possível menor rota (a partir do grupo inicial até o destino). E a partir deste momento, o algoritmo de roteamento entra em funcionamento.

É considerado que o usuário anda pela trilha de piso tátil.

É definido o no_atual como início (o primeiro grupo lido) para a primeira iteração do algoritmo de roteamento. A partir daí, o usuário começa a receber instruções e caminhar pela trilha. Para cada grupo (*tag ID*) lido pelo leitor RFID, entramos na rotina do algoritmo de roteamento: avaliação dos possíveis cenários analisados anteriormente, e as variáveis no_ant, no_atual, no_prox direção e caminho[ ] são atualizadas (no_ant recebe o no_atual, e o no_atual recebe o grupo lido pelo leitor RFID).

No primeiro cenário: há chamada das funções achaDireção e daInstrução, e como neste caso, o usuário percorre de maneira correta. Assim somente a variável direção é atualizada. E lembrando que neste caso há correção inteligente (caso o usuário pule um grupo de caminho[ ], mas continua na rota correta.

No segundo cenário: somente a função daInstrução é chamada, e como o usuário, de alguma maneira, voltou ao mesmo grupo lido, nenhuma variável é atualizada, e a instrução "Ande pela trilha" é dita ao usuário, com o intuito de atualizar a variável direção com a leitura do próximo *tag* (grupo).

No terceiro cenário: há chamada das funções achaDireção, do Algoritmo de Dijkstra e daInstrução. Pois neste caso, devemos calcular novamente o sentido e atualizar a variável direção; achar a nova rota (menor caminho), e posteriormente dar novas instruções ao usuário.

Esses três cenários podem ser observados no fluxograma abaixo, onde os losangos apresentam as decisões e os quadriláteros apresentam as ações do algoritmo de roteamento:

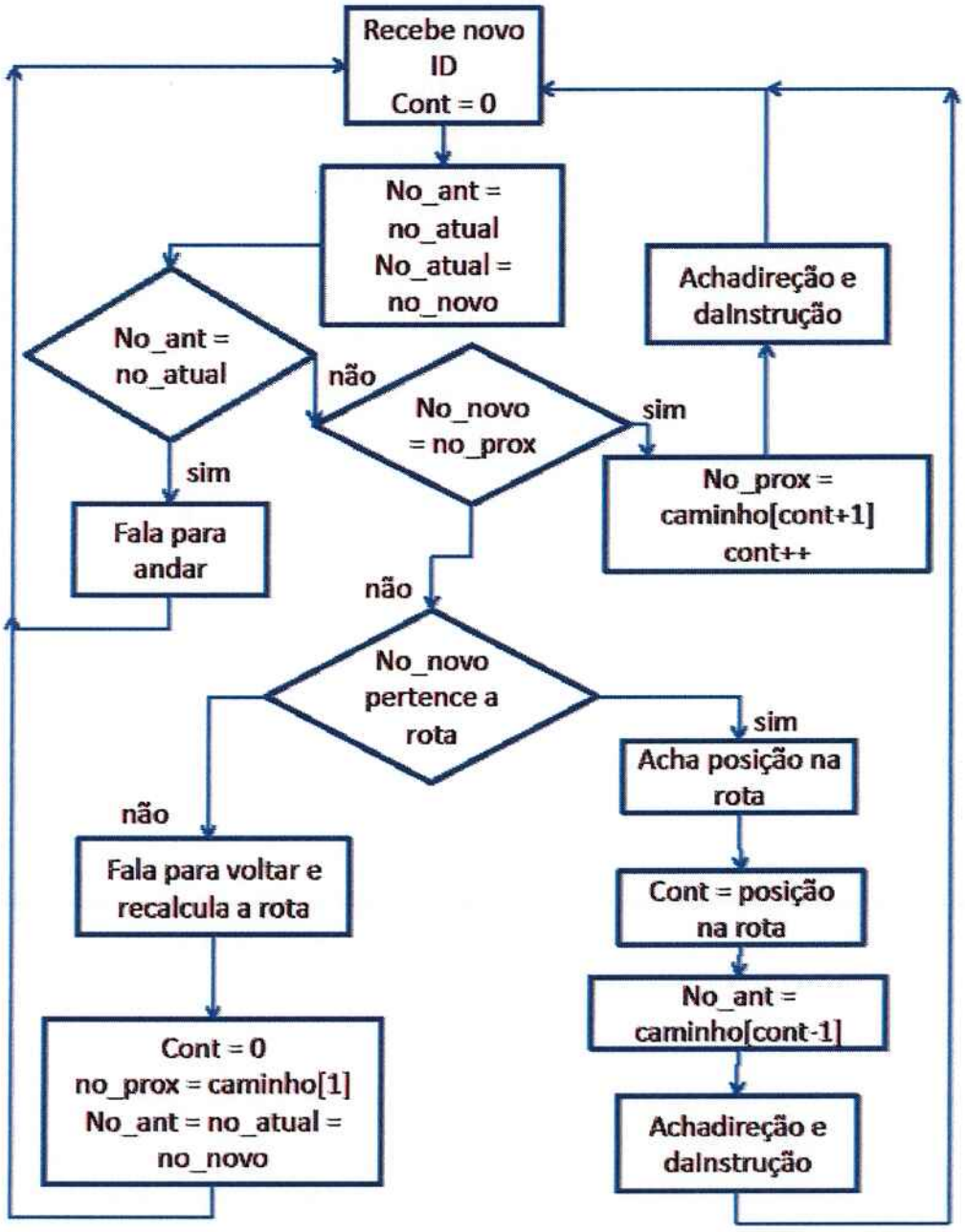


Figura 26: Fluxograma do Algoritmo de Roteamento

Primeiro, o sistema de roteamento recebe um novo ID (que é associado a algum grupo), e as variáveis no_ant e no_atual são atualizadas. Caso eles sejam iguais, o segundo cenário é abordado. Caso contrário, há a comparação entre o grupo lido pelo leitor e o próximo grupo do vetor caminho[ ]. Se essa comparação for verdadeira, o primeiro cenário é abordado (tudo está saindo como planejado). E caso contrário, há entrada em uma outra decisão.

Se o no_novo pertence a rota, entramos no primeiro cenário, na parte da correção inteligente (explicada anteriormente). Caso contrário, a instrução de "Dê meia volta" é dada, e o terceiro cenário é abordado.

E assim o algoritmo de roteamento fica em *loop* nesse rotina, até o usuário chegar ao destino desejado. E quando ele chega a esse destino, um novo destino deve ser escolhido (para dar continuidade ao programa).

6.6.5. Observações

Algumas observações podem ser feitas: com a implementação do projeto, novos cenário e análises podem surgir, bem como novas instruções que são dadas ao usuário; e se houve tempo suficiente, será possível implementar algumas opções sobre preferências do usuário, propagandas (uma mensagem por exemplo: "A loja X está próxima"), abortagem de rota (escolher algum novo destino ou por causa de alguma emergência), entre outros.

6.6.6. Lista de instruções

Seguem abaixo algumas instruções em voz que são úteis para guiar o usuário (que podem ser implementadas):

- Vire à esquerda;
- Vire à direita;
- Siga em frente;
- Dê meia volta;
- O destino desejado se encontra à direita (esquerda ou em frente);
- Cuidado com o degrau (e outras mensagens de perigo iminente);

- Serviços: sanitário masculino/feminino (fraudário, elevador entre outros) à direita/esquerda;
- Preferência de loja X na proximidade;
- Para abortar o destino, pressione o botão Y;
- Escolha seu destino;

6.7. Aplicativo principal

O aplicativo principal integra todas as funcionalidades desenvolvidas nesse projeto (comunicação com o Arduino, reconhecimento de voz, acesso ao banco de dados, roteamento, síntese de voz, etc.). A Figura 27 resume a sequência de tarefas realizadas: conexão com o hardware do projeto via Bluetooth, reconhecimento do destino utilizando voz e o roteamento do usuário até o destino.

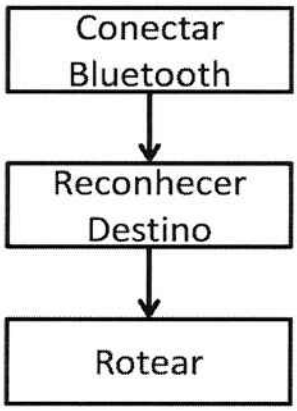


Figura 27: Sequência das tarefas realizadas pelo aplicativo principal

A Figura 28 apresenta um diagrama de classes simplificado da aplicação principal. A classe MainApplication integra as principais classes utilizadas no software do projeto e mantém um estado global para os diversos segmentos da aplicação.

A classe `BluetoothSerialService` possui as rotinas para a comunicação com o dispositivo Bluetooth, como as rotinas para procura de dispositivos, estabelecimento da conexão, recebimento dos valores dos IDs, etc.

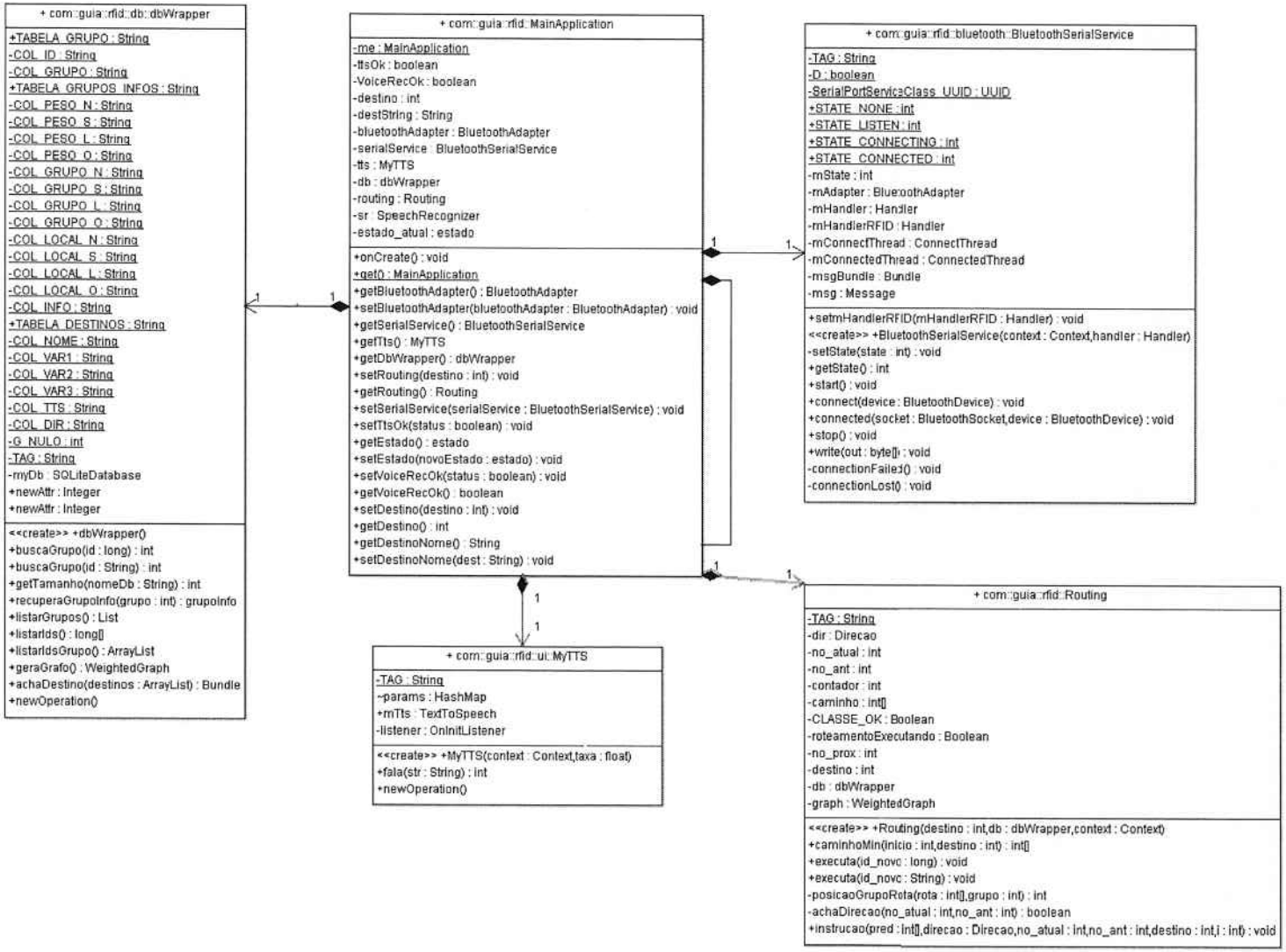


Figura 28: Diagrama de classes simplificado do aplicativo principal

O banco de dados é acessado através de uma única instância da classe `dbWrapper`, ou seja, todas as outras classes que possuem acesso ao banco de dados, o faz através da função `getDbWrapper()` implementada na classe `MainApplication`.

Da mesma forma, uma única instância da classe `MyTts` (para síntese de voz) é utilizada em todo o aplicativo. As classes que necessitam executar síntese de voz, acessam a instância do `MyTts` através do `getMyTts()` implementada no `MainApplication`.

Como a interação com o usuário ocorre principalmente por voz (reconhecimento e síntese), a interface gráfica do aplicativo é apenas uma tela preta sem nenhum objeto gráfico.

O núcleo do controle do software ocorre em segundo plano na classe `GuiaRfidService`, uma extensão da classe `Service` do Android API. O controle de todas as atividades do software é feito nessa classe, iniciando pelo controle da conexão com o dispositivo Bluetooth, chamada da aplicação para o reconhecimento do destino e o início e execução do roteamento.

7. TESTES

Ao se finalizar cada subsistema, é importante a realização de testes para verificar o seu correto funcionamento. A seguir serão descritos os procedimentos que foram adotados na verificação dos subsistemas mais críticos.

7.1. Teste do cálculo do melhor caminho

Para testar o funcionamento do algoritmo de determinação do melhor caminho, foi criado um aplicativo em que se pode escolher um grupo para o início da trajetória e outro grupo para o destino. Além disso, foi criado o banco de dados referente ao mapeamento ilustrado na Figura 29. Como o objetivo é testar a determinação da melhor rota, apenas as informações do grupo e os pesos são relevantes nesse banco de dados. A Figura 30 mostra a captura de tela da interface do aplicativo criado, onde foi escolhido o grupo 1 como ponto inicial e o grupo 30 como grupo destino.

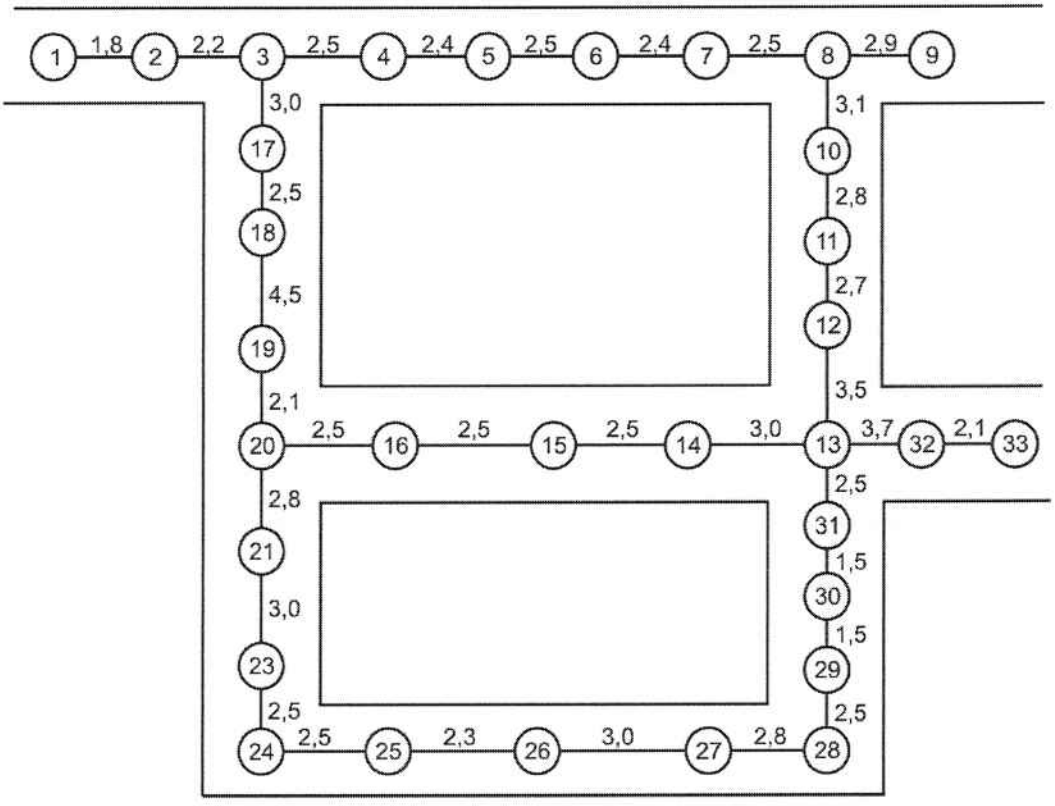


Figura 29: mapeamento para teste do do cálculo do melhor caminho

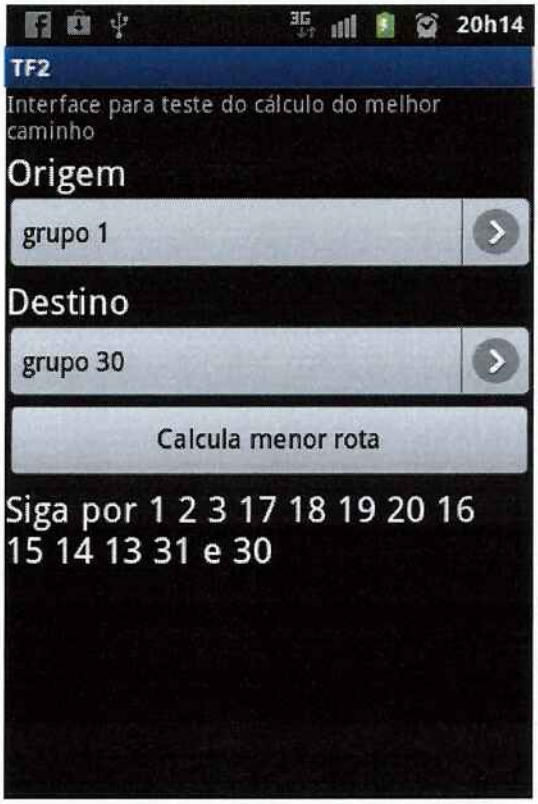


Figura 30: aplicativo para teste do cálculo do melhor caminho

Os testes realizados mostraram que o sistema reagia de forma coerente com o esperado, apresentando como resposta sempre o caminho de menor soma de pesos em direção ao destino indicado.

7.2. Teste do sistema de roteamento

Os testes de roteamento apresentam como entrada uma sequência de grupos percorridos pelo usuário e como saída as instruções dadas ao usuário pelo *TextToSpeech*.

Dado o mapeamento representado na Figura 31 e o grupo inicial (posição inicial do usuário), para cada sequência de grupos, obtêm-se instruções em voz do roteamento. Neste caso, o grupo 1 foi considerado como a posição inicial e o grupo 16 como destino.

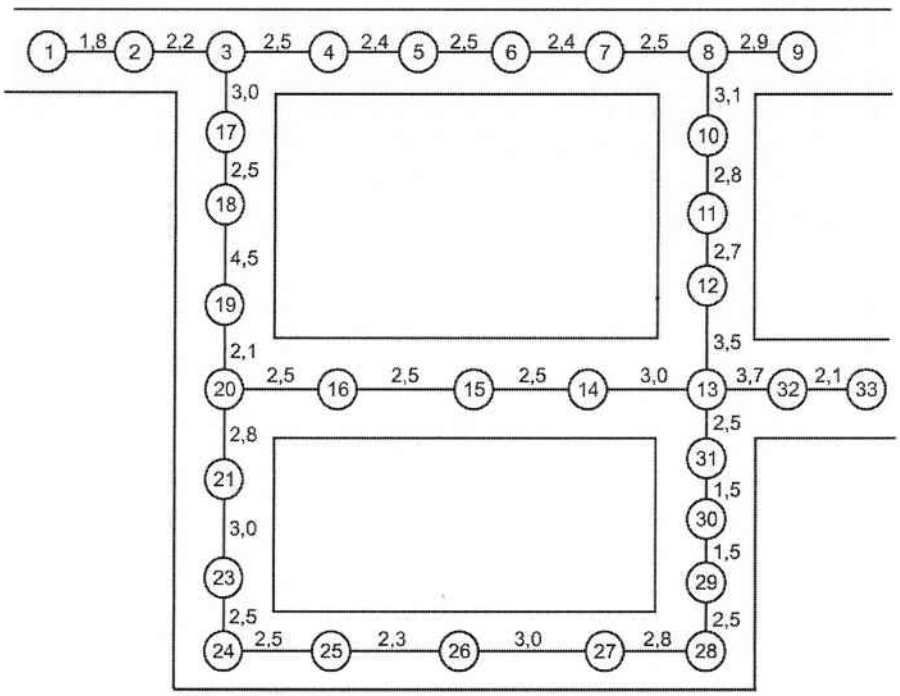


Figura 31: Mapeamento exemplo para testes

- Teste 1: Sequência de grupos: [1 1 2 3 17 18 19 20 16]

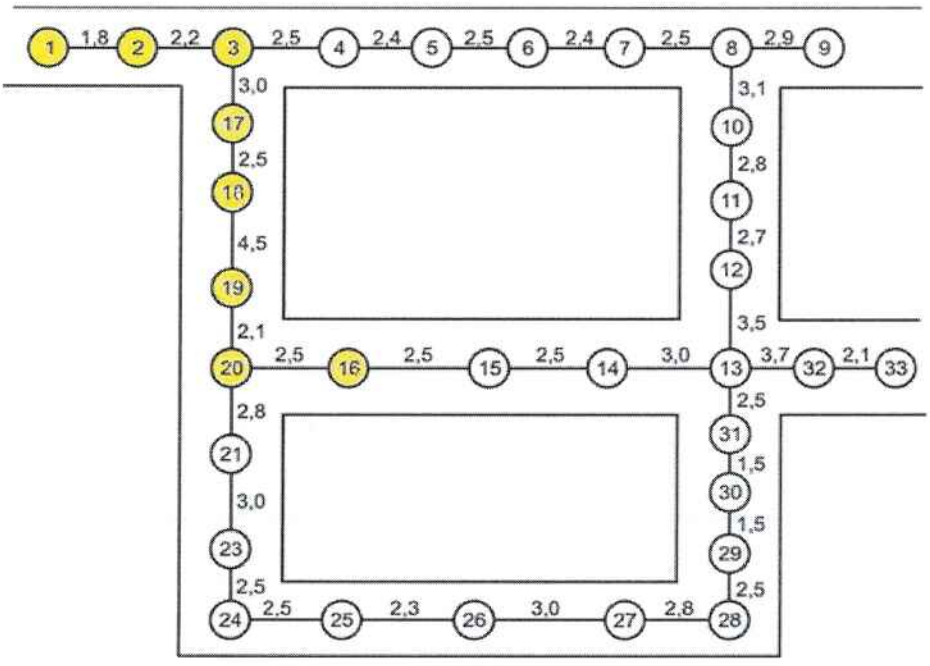


Figura 32: Teste para a sequência de grupos [1 1 2 3 17 18 19 20 16]

Tabela 3: Instruções recebidas para uma sequência de leituras de grupos contínuas e dentro da rota

Grupo	Instrução
1	Comece a andar pela trilha de piso tátil
1	Ande pela trilha
2	Siga em frente
3	Vire à direita
17	Siga em frente
18	Siga em frente
19	Siga em frente
20	Vire à esquerda
16	Você chegou ao seu destino

O teste 1 realiza um caso em que o usuário permanece dentro da rota mínima, mas que ocorre a leitura de um mesmo grupo duas vezes. Quando o mesmo grupo é lido duas vezes, o sistema de roteamento entende que o usuário não andou, então é dada a instrução "Ande pela trilha".

- Teste 2: sequência de grupos: [1 2 3 18 20 16]

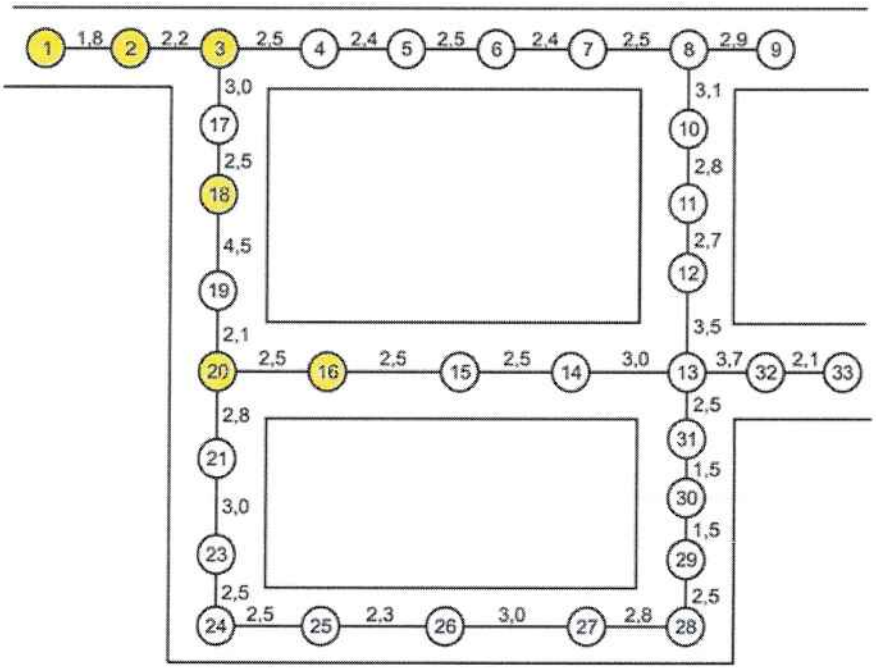


Figura 33: Teste para a sequência de grupos [1 2 3 18 20 16]

Tabela 4: Sequência de instruções para teste de salto de grupo

Grupo	Instrução
1	Comece a andar pela trilha de piso tátil
2	Siga em frente
3	Vire à direita
18	Siga em frente
20	Vire à esquerda
16	Você chegou ao seu destino

No teste 2 ocorre do usuário andar sob o caminho de rota mínima, porém saltando alguns grupos (os grupos 17 e 19 não foram lidos durante o roteamento). Apesar de não ter havido a leitura de todos os grupos pertencentes à rota que o usuário deve seguir desde o ponto inicial até o ponto final, o sistema de roteamento considera que o usuário está andando corretamente até o destino e fornece as instruções normalmente como se não houvesse pulado grupos.

- Teste 3: sequência de grupos: [1 2 4 5 4 3 17 18 19 20 16]

Tabela 5: Sequência de instruções do teste de quando o usuário sai da rota

Grupo	Instrução
1	Comece a andar pela trilha de piso tátil
2	Siga em frente
4*	Dê meia volta
5**	Dê meia volta
4	Siga em frente
3	Vire à esquerda
17	Siga em frente
18	Siga em frente
19	Siga em frente
20	Vire à esquerda
16	Você chegou ao seu destino

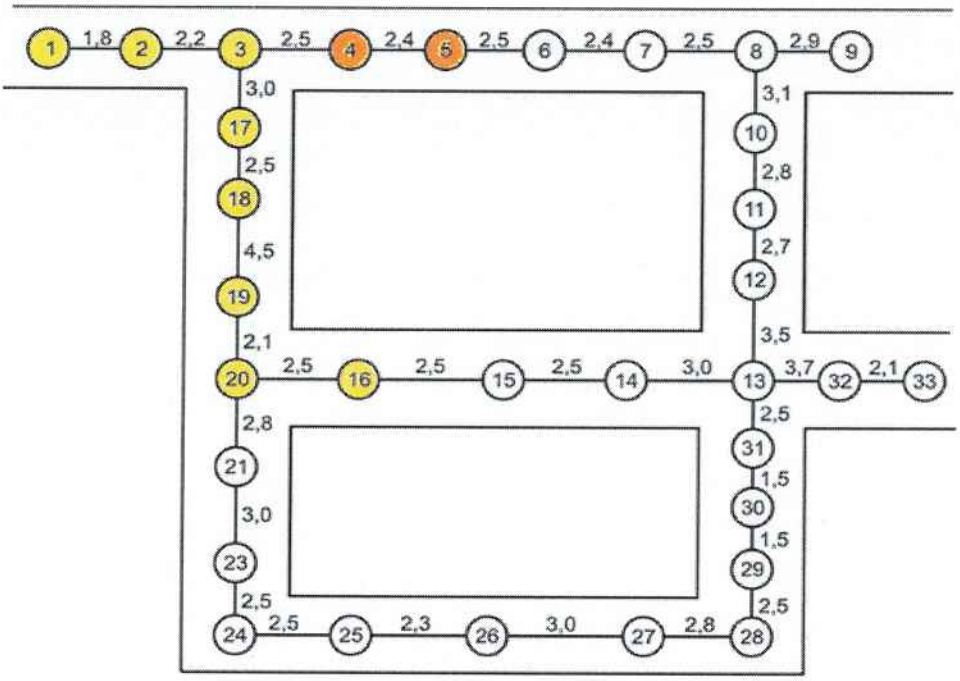


Figura 34: Teste de quando o usuário sai da rota

O teste 3 refere-se a um caso em que o usuário sai da rota mínima, nesse caso, o usuário vai para o grupo 4 após o grupo 3. No grupo 4* o roteamento não sabe a direção do usuário, assim a instrução de dar a meia volta tem o objetivo de descobrir a essa direção; em 5** já há o cálculo da menor rota pelo Algoritmo de Dijkstra e a direção do usuário é atualizada e a instrução "Dê meia volta" é dada ao usuário com o intuito de percorrer a menor rota (diferente na leitura do grupo 4*).

7.3. Testes de leitura de *tags*

Foram realizados testes do alcance de leitura do módulo RFID com as *tags* 125 kHz que são utilizadas nesse projeto, com a antena em duas situações: ao ar livre e dentro de um encapsulamento de folha de madeira. A Tabela 6 a seguir mostra o resultado desses testes.

Tabela 6: Testes do alcance de leitura das tags

Antena	Distância máxima (cm)
Ao ar livre	4,5
Encapsulada	4,8

Observou-se que quando uma tag passa rapidamente pela região de alcance de leitura, o modulo não consegue realizar a leitura e que, apesar dos resultados semelhantes, houve uma variação considerável entre as distâncias cada vez que o teste foi realizado.

7.4. Teste de comunicação

Para verificar se o módulo RDM630 está se comunicando corretamente com o microcontrolador, após a conexão das portas seriais do módulo nas portar UART do Arduino, foi monitorado a recepção dos dados recebidos pelo Arduino através da ferramenta *Serial Monitor.*

O envio dos dados do Arduino para o *smartphone* foi testado através da criação de um aplicativo no Android que mostra a recepção de dados do acessório (no caso desse projeto, valores de IDs).

7.5. Demonstração

Pretende-se criar um ambiente para a demonstração do projeto, conforme descrido no Apêndice III. Para isso, deverá ser criada uma trilha com etiquetas RFID fixadas. O sistema deverá ser capaz de, dado o local de destino, guiar o usuário através dessa trilha utilizando instruções em voz.

8. CONCLUSÃO

O projeto foi especificado de forma a desenvolver, durante um semestre letivo, um sistema de roteamento para deficientes visuais de maneira viável economicamente. Para evitar um alto custo, foi definida a utilização da tecnologia RFID em 125 kHz e, para que se viabilizasse a implementação do projeto num prezo de quatro meses, foi definido a utilização do Arduino Mega ADK utilizando módulos com interfaces compatíveis. O módulo RDM6300 e o módulo Bluetooth Serial Mate com suas interfaces seriais foram diretamente compatíveis com as portas de comunicação seriais do Arduino, bastando fazer as devidas configurações e programação via software no microcontrolador.

A utilização da plataforma Android proporcionou a implementação eficiente de uma interface com o usuário intuitiva através de seus recursos de reconhecimento e síntese de voz. A necessidade de que o *smartphone* esteja conectado à internet para o funcionamento do sistema de reconhecimento de voz é uma desvantagem dessa plataforma.

A tecnologia RFID em 125 kHz, apesar de ser a de menor custo, mostrou-se ineficiente em relação ao alcance de leitura das *tags*. Devido ao seu baixo alcance de leitura e ao tempo de mínimo necessário para que uma *tag* seja lida pelo módulo RFID, é necessário que a antena esteja a uma distância máxima de três centímetros do chão e que o usuário não ande muito rápido durante o roteamento. Esse problema pode ser resolvido pela substituição do módulo RDM6300 por módulos que trabalham em freqüências mais elevadas, como em 13,56 MHz ou 915 MHz. No entanto, quanto maior a freqüência de operação dos módulos e etiquetas RFID e seus alcances de leitura, maior é o custo dos dispositivos. Por exemplo, o módulo RDM880 (mesmo fabricante do módulo adotado nesse projeto e mesmo layout de pinos), que opera em 13,56 MHz e possui um alcance de leitura de 3 a 10 cm e preços variando de 3 a 5 vezes o preço do módulo de 125 kHz utilizado. As *tags* 13.56 MHz e 915 MHz também são mais caras (*tags* 13,56 MHz custam o dobro do preço de *tags* 125 kHz em média).

Para módulos em 13,56 MHz, o sistema desse projeto é diretamente adaptável, pois o alcance de leitura desses módulos são tipicamente em torno de 7 cm, o que

ainda manteria a característica de ser possível localizar o usuário com uma precisão satisfatória apenas com a leitura de uma única *tag*. Portanto, pode-se considerar que esse trabalho prova o conceito proposto de guiar deficientes visuais através de uma interface intuitiva em locais mapeados com etiquetas RFID. Trabalhos posteriores podem chegar num produto eficiente e viável utilizando essa tecnologia.

REFERÊNCIAS

BAL, E.; A.S, P. O. **An Rfid application for the disabled:** Path Finder. 1st Annual RFID Eurasia. Istanbul: [s.n.]. 2007.

BEN-HAIM, N. Google Mobile Blog, 13 July 2011. Disponivel em: <http://googlemobile.blogspot.com/2011/07/live-traffic-information-for-13.html>. Acesso em: 02 fev. 2012.

BOUET, M.; DOS SANTOS, A. L. **RFID Tags:** Positioning Principles and Localization Techniques. Wireless Days. [S.l.]: [s.n.]. 2008.

BOUET, M.; DOS SANTOS, A. L. **RFID tags:** Positioning principles and localization techniques. Wireless Days. [S.l.]: [s.n.]. 2008. p. 1-5.

BUIKEMA, B. High-Speed Blog, 13 August 2010. Disponivel em: <http://blog.brianbuikema.com/2010/08/android-development-part-1-using-googles-places-api-to-develop-compelling-location-based-mobile-applications/>. Acesso em: 02 fev. 2012.

CASSOL CENTER LAR. **Cassol Center Lar**. Disponivel em: <http://www.cassol.com.br/blog/calcada-segura/>. Acesso em: 19 jun. 2012.

EYBEN, F.; WÖLLMER, M.; SCHULLER, B. **Talking Car and Virtual Companion Technical Report WP2: Use-Cases for Emotion Recognition in the Car**. Munich, Germany.

FACILITÁ. Disponivel em: <http://pisotatil.com/curitiba/>. Acesso em: Novembro 2012.

FUNDAÇÃO DORINA NORWILL PARA CEGOS. **Fundação Dorina Norwill para cegos**. Disponivel em: <http://www.fundacaodorina.org.br/deficiencia-visual>. Acesso em: 14 mar. 2012.

GOUVEIA, J. Android Market, April 2011. Disponivel em: <https://market.android.com/details?id=com.gouveia.eventscheduler&feature=search_result>. Acesso em: 05 fev. 2011.

GREGORY, D. R. et al. Poli Cidadã, 2011. Disponivel em: <http://www.policidada.poli.usp.br/projetos/projeto/view/55/>. Acesso em: 19 jun. 2012.

GUIO SYSTEMS. Guio Solid Step. **Guio Systems**. Disponivel em: <http://www.guio.pt/wp/?page_id=50>. Acesso em: 20 jun. 2012.

INSTITUTO DE RESPONSABILIDADE E INCLUSÃO SOCIAL. **Instituto de Responsabilidade e Inclusão Social**. Disponivel em: <http://www.iris.org.br/treinamento_habilidades.asp>. Acesso em: 14 mar. 2012.

ITEAD Studio. **http:** //iteadstudio.com/produce/play-rdm630-with-arduino/.

ITEAD Studio. **http:** //iteadstudio.com/application-note/use-the-rdm880-to-readwrite-the-iso14443-type-a-rfid-cardtag/.

KUDO, J. M. et al. Poli Cidadã, 2011. Disponivel em: <http://www.policidada.poli.usp.br/projetos/projeto/view/56/>. Acesso em: 19 jun. 2012.

LEAL, J. P., 1999. Disponivel em: <http://www.dcc.fc.up.pt/~zp/aulas/9899/me/trabalhos/alunos/Algoritmos_do_Caminho_Minimo_em_Grafos/ddijk.html>. Acesso em: 25 set. 2012.

LIM, C.-H. et al. A real-time indoor WiFi localization system utilizing smart antennas. **IEEE Transactions on Consumer Electronics**, v. 53, p. 618-622, Maio 2007.

MADE-IN-CHINA.COM. Disponivel em: <http://rfid-nfc.en.made-in-china.com/>. Acesso em: 15 abr. 2012.

MADE-IN-CHINA.COM. Disponivel em: <http://rfid-nfc.en.made-in-china.com/productimage/IbPJZOAVkMWd-2f0j00vsLabCBgAGkn/China-UHF-RFID-Reader-Module-NFC-9802M-.html>. Acesso em: Novembro 2012.

MADE-IN-CHINA.COM. Disponivel em: <http://cnfidelity.en.made-in-china.com/productimage/hBYnZRLcYJkI-2f0j00IZcaPlGqsukg/China-UHF-RFID-Laundry-Tag-FDY-008A-.html>. Acesso em: Novembro 2012.

MARIANI, A. C. Departamento de Informática e de Estatística. **Universidade Federal de SantaCatarina**. Disponivel em: <http://www.inf.ufsc.br/grafos/temas/custo-minimo/dijkstra.html>. Acesso em: 25 set. 2012.

MASSAROPPE, L.; REBUCCI, P. B. Poli Cidadã, 2011. Disponivel em: <http://www.policidada.poli.usp.br/projetos/projeto/view/53/>. Acesso em: 19 jun. 2012.

NONATO, L. G. Instituto de Ciências Matemáticas e de Computação. **Departamento de Computacao e Estatística**, 2004. Disponivel em: <http://www.lcad.icmc.usp.br/~nonato/ED/Dijkstra/node84.html>. Acesso em: 25 set. 2012.

O'CONNOR, M. C. Health care news. **RFID Journal**, 2009. Disponivel em: <http://www.rfidjournal.com/article/view/5214>. Acesso em: 6 Junho 2012.

RDM630 Specification. **Datasheet**. Disponivel em: <http://iteadstudio.com/store/images/produce/RFID/125KReader_U/RDM630-Spec.pdf>. Acesso em: out. 2012.

RFID Shop. Disponivel em: <http://www.rfidshop.net/index.php?main_page=product_info&cPath=&products_id=238>. Acesso em: Novembro 2012.

RFID Shop, 2012. Disponivel em: <http://www.rfidshop.net/index.php?main_page=product_info&cPath=&products_id=238>. Acesso em: Novembro.

SESAMONET. European Commission's Joint Research Centre. **Joint Research Centre**, 2007. Disponivel em: <http://ec.europa.eu/dgs/jrc/index.cfm?id=4210>. Acesso em: 19 jun. 2012.

SWISS NATIONAL SCIENCE FOUNDATION. **Molible information and communication systems**, 25 ago. 2005. Disponivel em: <http://www.mics.org/Annexes/report-ip6-y4.pdf>. Acesso em: 15 abr. 2012.

WEINSTEIN, R. RFID: A Technical Overview and Its Application to the Enterprise. **IT Professional**, v. 7, p. 27-33, 2005.

YELAMARTHI, K. et al. RFID and GPS integrated navigation system for the visually impaired. In: 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS). Seattle: [s.n.]. 2010. p. 1149 - 1152.

ZHOU, J.; SHI, J. RFID localization algorithms and applications - a review. **Journal of Intelligent Manufacturing**, v. 20, n. 6, p. 695-707, 2009.

APÊNDICE I

Algoritmo de Dijkstra

O algoritmo de Dijkstra é o mais famoso e o mais empregado dos algoritmos para a determinação do caminho de menos custo entre dois vértices de um grafo com pesos (NONATO, 2004).

Dado um vértice como origem da busca, o algoritmo calcula o custo mínimo deste vértice em relação a todos os outros vértices do grafo. Partindo de uma estimativa inicial para o custo mínimo, ele vai ajustando essa estimativa sucessivamente (MARIANI).

Seja $G(V, A)$ um grafo orientado, onde:

V– conjunto não vazio dos vértices e

A – conjunto de pares ordenados $a = (v, w)$ onde $v, w \in V$ são as arestas do grafo e v é pai de w (grafo orientado).

O algoritmo de Dijkstra pode ser descrito, segundo (LEAL, 1999): seja s um vértice de G, denominado vértice inicial. Inicialmente, considera-se a distância ao próprio s como valento 0, ou seja, $dist(s) = 0$ e $S = \{s\}$. Para todos os outros vértices, considera-se a distância como valendo infinito, ou seja, $dist(s) = \infty$. A introdução em S de um vértice v_m não pertencente a S faz-se:

1. Qualquer que seja v_m não pertencente a S tal que v foi o último nó a entrar em S e $(v, v_m) \in V$, se $dist(v_m) > dist(v)$ + distância associada a (v, v_m), então:
   $dist(v_m) = dist(v)$ + distância associada a (v, v_m)
2. Determinar qualquer que seja v_m não pertencente a S o menor de entre os valores de $dist(v_m)$. Seja $dist(v_j)$ esse valor;
3. Fazer $v_n = v_j$, $dist(v_n) = dist(v_j)$ e v_n passa a ser o novo elemento de S
4. Atribua valor zero à estimativa do custo mínimo do vértice s (a raiz da busca) e infinito às demais estimativas;

Se o vértice v_n coincidir com o vértice final então $dist(v_n)$ é a menor distância entre s e v_n, e parar a execução. Se não coincidir, voltam-se a repetir os passos 1, 2 e 3.

Quando todos os vértices tiverem sido fechados, os valores obtidos serão os custos mínimos dos caminhos que partem do vértice tomado como raiz da busca até os demais vértices do grafo. O caminho propriamente dito é obtido a partir dos vértices chamados acima de precedentes.

APÊNDICE II

Piso tátil

O piso tátil é uma estrutura que utiliza uma textura (relevo) e cor diferenciada dos pisos comuns. Sua função é alertar e guiar pessoas com deficiência visual e baixa capacidade de visão através do sentido do tato. Essas diferenças de texturas e cores proporcionam aos usuários segurança, orientação e autonomia.

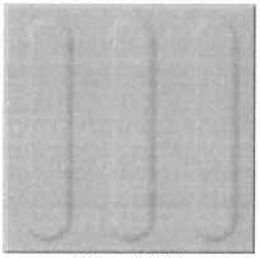
De acordo com a norma brasileira NBR 9050 existem dois tipos de pisos tátil: o de alerta e o direcional.

O primeiro apresenta uma estrutura antiderrapante com várias fileiras de bolinhas, que possui a função de alertar o usuário algum tipo de perigo e obstáculo iminente ou mudança de direção (sentido) da trilha.

O segundo apresenta normalmente uma estrutura com 3 fileiras contínuas, que é usado para guiar e dar direção (sentido) da caminhada.

Ambos são utilizados em espaços públicos: ruas, *shopping centers*, escolas, parques, entre outros.

Seguem abaixo as figuras piso tátil direcional e piso tátil de alerta, respectivamente:



(Facilitá)

(Facilitá)

Neste projeto, o piso tátil é utilizado como trilha para a implementação e fixação das etiquetas RfId, de maneira a facilitar o roteamento do usuário até o destino desejado.

APÊNDICE III

Implementação de um mapeamento teste

O mapeamento será feito no segundo andar do Bloco B do prédio da Engenharia Elétrica: os lugares de acesso e destino são as salas de aulas B2-01 a B2-12, a cantina, e as rampas que dão acesso ao bloco A e ao bloco C.

A figura abaixo mostra como o mapeamento foi feito: as linhas contínuas azuis representam onde as trilhas de etiquetas e o piso tátil serão colocados, e as bolinhas azuis são as localizações dos grupos de etiquetas. No total há 46 grupos de etiquetas, serão utilizadas cercas de 100 etiquetas. E para cada grupo, 2 etiquetas serão utilizadas para os grupos 'comuns' e 3 etiquetas serão utilizadas para os grupos 'bifurcações'. As medidas de distâncias e os números dos grupos foram omitidos, para não poluir visualmente a imagem.

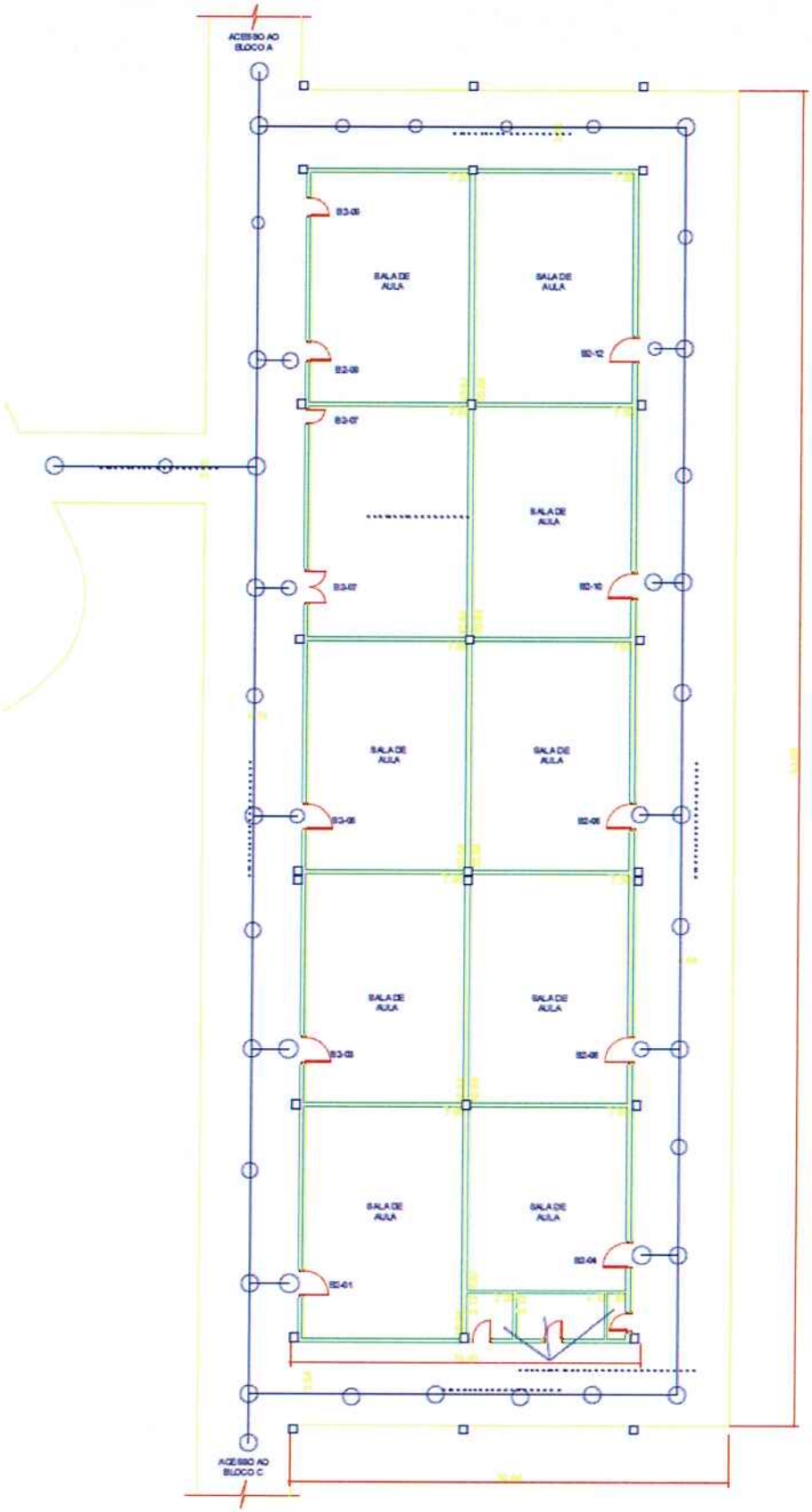
O planejamento do mapeamento se baseou em que as trilhas sejam colocadas no ponto médio do caminho (de maneira centralizada) e as etiquetas colocadas a cada 5~5.3 metros de distância, também há um grupo de etiquetas para cada sala do Bloco B2.

Durante a implementação, como não compramos o piso tátil por motivos financeiros, podemos utilizar fita isolante, ou qualquer outro material (ou combinação de materiais) para fixar as etiquetas e 'simular um piso tátil'.

Alguns cuidados/medidas devem ser adotados na etapa de implementação do mapeamento, como deixar algum aviso para as pessoas (universitários e funcionários) não danificar as trilhas, e ter permissão (com um aviso prévio da COD da Elétrica) para implementar o mapeamento no prédio da Engenharia Elétrica.

A implementação deve durar cerca de 1 a 2 dias, e planejamos obter permissão para utilizar o prédio da Elétrica como local de implementação por uma semana (validação e testes).

Segue abaixo a imagem do mapeamento do bloco B, que será utilizado na validação desse projeto:



APÊNDICE IV

Tabela ASCII

DEC	HEX	BIN	Símbolo	Descrição
0	0	0	NUL	Null char
1	1	1	SOH	Start of Heading
2	2	10	STX	Start of Text
3	3	11	ETX	End of Text
4	4	100	EOT	End of Transmission
5	5	101	ENQ	Enquiry
6	6	110	ACK	Acknowledgment
7	7	111	BEL	Bell
8	8	1000	BS	Back Space
9	9	1001	HT	Horizontal Tab
10	0A	1010	LF	Line Feed
11	0B	1011	VT	Vertical Tab
12	0C	1100	FF	Form Feed
13	0D	1101	CR	Carriage Return
14	0E	1110	SO	Shift Out / X-On
15	0F	1111	SI	Shift In / X-Off
16	10	10000	DLE	Data Line Escape
17	11	10001	DC1	Device Control 1 (oft. XON)
18	12	10010	DC2	Device Control 2
19	13	10011	DC3	Device Control 3 (oft. XOFF)
20	14	10100	DC4	Device Control 4
21	15	10101	NAK	Negative Acknowledgement
22	16	10110	SYN	Synchronous Idle
23	17	10111	ETB	End of Transmit Block
24	18	11000	CAN	Cancel
25	19	11001	EM	End of Medium
26	1A	11010	SUB	Substitute
27	1B	11011	ESC	Escape
28	1C	11100	FS	File Separator
29	1D	11101	GS	Group Separator
30	1E	11110	RS	Record Separator
31	1F	11111	US	Unit Separator
32	20	100000		Space
33	21	100001	!	Exclamation mark
34	22	100010	"	Double quotes (or speech marks)
35	23	100011	#	Number
36	24	100100	$	Dollar
37	25	100101	%	Procenttecken
38	26	100110	&	Ampersand

39	27	100111	'	Single quote
40	28	101000	(	Open parenthesis (or open bracket)
41	29	101001	)	Close parenthesis (or close bracket)
42	2A	101010	*	Asterisk
43	2B	101011	+	Plus
44	2C	101100	,	Comma
45	2D	101101	-	Hyphen
46	2E	101110	.	Period, dot or full stop
47	2F	101111	/	Slash or divide
48	30	110000	0	Zero
49	31	110001	1	One
50	32	110010	2	Two
51	33	110011	3	Three
52	34	110100	4	Four
53	35	110101	5	Five
54	36	110110	6	Six
55	37	110111	7	Seven
56	38	111000	8	Eight
57	39	111001	9	Nine
58	3A	111010	:	Colon
59	3B	111011	;	Semicolon
60	3C	111100	<	Less than (or open angled bracket)
61	3D	111101	=	Equals
62	3E	111110	>	Greater than (or close angled bracket)
63	3F	111111	?	Question mark
64	40	1000000	@	At symbol
65	41	1000001	A	Uppercase A
66	42	1000010	B	Uppercase B
67	43	1000011	C	Uppercase C
68	44	1000100	D	Uppercase D
69	45	1000101	E	Uppercase E
70	46	1000110	F	Uppercase F
71	47	1000111	G	Uppercase G
72	48	1001000	H	Uppercase H
73	49	1001001	I	Uppercase I
74	4A	1001010	J	Uppercase J
75	4B	1001011	K	Uppercase K
76	4C	1001100	L	Uppercase L
77	4D	1001101	M	Uppercase M
78	4E	1001110	N	Uppercase N
79	4F	1001111	O	Uppercase O
80	50	1010000	P	Uppercase P
81	51	1010001	Q	Uppercase Q
82	52	1010010	R	Uppercase R
83	53	1010011	S	Uppercase S

84	54	1010100	T	Uppercase T
85	55	1010101	U	Uppercase U
86	56	1010110	V	Uppercase V
87	57	1010111	W	Uppercase W
88	58	1011000	X	Uppercase X
89	59	1011001	Y	Uppercase Y
90	5A	1011010	Z	Uppercase Z
91	5B	1011011	[	Opening bracket
92	5C	1011100	\	Backslash
93	5D	1011101	]	Closing bracket
94	5E	1011110	^	Caret - circumflex
95	5F	1011111	_	Underscore
96	60	1100000	`	Grave accent
97	61	1100001	a	Lowercase a
98	62	1100010	b	Lowercase b
99	63	1100011	c	Lowercase c
100	64	1100100	d	Lowercase d
101	65	1100101	e	Lowercase e
102	66	1100110	f	Lowercase f
103	67	1100111	g	Lowercase g
104	68	1101000	h	Lowercase h
105	69	1101001	i	Lowercase i
106	6A	1101010	j	Lowercase j
107	6B	1101011	k	Lowercase k
108	6C	1101100	l	Lowercase l
109	6D	1101101	m	Lowercase m
110	6E	1101110	n	Lowercase n
111	6F	1101111	o	Lowercase o
112	70	1110000	p	Lowercase p
113	71	1110001	q	Lowercase q
114	72	1110010	r	Lowercase r
115	73	1110011	s	Lowercase s
116	74	1110100	t	Lowercase t
117	75	1110101	u	Lowercase u
118	76	1110110	v	Lowercase v
119	77	1110111	w	Lowercase w
120	78	1111000	x	Lowercase x
121	79	1111001	y	Lowercase y
122	7A	1111010	z	Lowercase z
123	7B	1111011	{	Opening brace
124	7C	1111100 |		Vertical bar
125	7D	1111101	}	Closing brace
126	7E	1111110	~	Equivalency sign - tilde
127	7F	1111111		Delete