UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA
FAET – FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

MICHEL CALHEIROS DA ROCHA

SISTEMA DE LOCALIZAÇÃO PARA DEFICIENTES VISUAIS:

Sistema de localização para deficientes visuais utilizando algoritmos de deslocamento e precisão

BRASÍLIA/DF
DEZEMBRO DE 2007

MICHEL CALHEIROS DA ROCHA

SISTEMA DE LOCALIZAÇÃO PARA DEFICIENTES VISUAIS:

Sistema de localização para deficientes visuais utilizando algoritmos de deslocamento e precisão

Monografia apresentada ao Curso de Engenharia da Computação, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Prof. Luís Cláudio Lopes de Araújo

BRASÍLIA/DF

2º SEMESTRE DE 2007

Resumo

Os receptores GPS de navegação são capazes de calcular e armazenar as posições, no entanto, não são capazes de transmitir as informações sonoras a deficientes visuais. O objetivo deste trabalho foi desenvolver uma solução conjunta entre um receptor GPS e aplicações de softwares com o objetivo de prover essas informações a um deficiente visual. Informações estas interpretadas pelo software, aplicados algoritmos de deslocamento e precisão capaz de um deficiente se localizar diante de um ambiente físico.

Palavras chave: GPS, Cálculos com Coordenadas, Localização, Algoritmos de Roteamento, Velocidade, Distância, Tempo Previsto, Banco de Dados.

Abstract

The GPS receivers for navigation are able to calculate and store the positions, however, they are not capable of transmitting the information through sound to a visually impaired person. The objective of this work has been to develop a joint solution between a GPS receiver and applications of software in order to provide such information to the visually impaired. This information are interpreted by software, displacement and precision algorithms are applied and are able to guide a blind front of a physical environment.

Keywords: GPS, Coordinates with Math, Location, Routing Algorithms, Speed, Distance, Time Set, Database.

Agradecimentos

O agradecimento é como compartilhar informações, satisfaz a ambos, o elogio também funciona de forma semelhante servindo de alimento para a alma. Mas as palavras de agradecimento têm que ser sinceras para ter efeito.

Por dias pensei naqueles que me incentivaram nesse longo e árduo caminho, as memórias vinham e eu viajava em minhas lembranças, por vezes me peguei com o sorriso estampado no rosto olhando para o nada.

Em algumas palavras gostaria de agradecer àqueles que se fizeram importantes no decorrer desses cinco anos e seis meses de luta.

Primeiramente ao único conhecedor da verdade absoluta, Deus.

Sou bastante grato aos mestres que tive a felicidade de encontrar pelo caminho, por suas dedicações, em especial aos professores Francisco Javier, Antônio Gonçalves, Maria Marony, Cláudio Penedo, Carmo Gonçalves e Fabiano Mariath.

Aos amigos de faculdade, Gustavo de Faria(Capitão), Rodrigo Benin(Xamba), Diogo Curto(Diogro), Leonardo Rocha(Frangolino), Leandro Almeida(Cabide), Wagner Bastos(Wagnão), Ricardo Rebelo(Careca), Daniel Neto, Roberta Nedder e aos demais.

Grato aos amigos que sempre me deram força nos momentos difíceis, Luiz Fernando(Troncho), Luiz Miamoto(Magrissa), Hélio Sant'Ana(Peçanha), Hugo Fagner; Fabrício de Amorin, por ter me ameaçado quando comentei que pensava em largar o curso e voltar para minha cidade natal (Maceió).

Aos meus familiares, sou abençoado por ter uma família tão maravilhosa sempre disposta a dar o máximo uns pelos outros, principalmente a meus Padrinhos Hayton e Magdala, sem eles não seria possível dar continuidade a um sonho juvenil, eternamente grato.

Em especial, agradeço a meu orientador, amigo e professor, Luís Cláudio, exemplo de dedicação, competência e amizade, sabe falar as palavras corretas nas horas apropriadas e também de tem o dom de incentivar, cobrar e punir quando necessário.

“Só eu sei as esquinas por que passei
Só eu sei os desertos que atravessei
Sabe lá o que é não ter e ter que ter pra dar
Sabe lá o que é morrer de sede em frente ao mar
E quem será na correnteza do amor que vai saber se guiar
A nave em breve ao vento vaga de leve e trás toda a paz
que um dia o desejo levou”

Djavan

Índice

1. INTRODUÇÃO 12
1.1. Contextualização do Problema 13
1.2. Objetivo do Projeto 13
1.3. Motivação 14
1.4. Estrutura do Trabalho 15
2. CONCEITOS BÁSICOS SOBRE O FUNCIONAMENTO DO GPS 16
2.1. O Sistema de Posicionamento Global (GPS) 16
2.2. Histórico 17
2.3. Segmentos do sistema 18
2.3.1. Segmento espacial 18
2.3.2. Segmento de controle 22
2.3.3. Segmento de usuários 23
2.4. Funcionamento do sistema 24
2.4.1. Fundamentação matemática 26
2.4.2. Coordenadas geográficas de um ponto no espaço 30
2.4.3. Exemplo real de triangulação 32
2.5. Receptores GPS 35
2.6. Precisão do Posicionamento GPS 37
2.6.1. Fontes de Erros 38
2.6.1.1. Erros por Satélites 38
2.6.1.2. Erros pelos receptores 39
2.6.1.3. Erros pelo meio de transmissão 40
3. COORDENADAS GEOGRÁFICAS 41
3.1. Localização 41
3.2. Sistema de Coordenadas Geográficas 43
3.3. Distância entre dois pontos de coordenadas dadas 45
3.3.1. Encontrando o raio da Terra 47
3.3.2. Trigonometria 49
3.3.2.1. Trigonometria plana 50
3.3.2.2. Trigonometria esférica 52
3.4. Ângulo de deslocamento para melhor rota 54

3.5. Velocidade de deslocamento 58
3.6. Tempo de chegada 59
4. CONSTRUÇÃO DO PROTÓTIPO 61
4.1. Hardware 61
4.1.1. Diagrama de Blocos 61
4.1.2. O Receptor GPS 62
4.1.3. Porta Serial 63
4.1.3.1. Comunicação com RS-232 63
4.1.3.2. Conversor USB - Serial 64
4.1.4. O Notebook 65
4.2. Software 66
4.2.1. Sistema Operacional 66
4.2.2. Framework 66
4.2.3. MYSQL 67
4.2.4. Microsoft Agent 2.0 67
4.2.5. Alfabeto Fonético L&H 67
4.2.6. Visual Studio 2005 68
4.2.7. DBDesigner 4.0 68
5. SISTEMA DE LOCALIZAÇÃO 69
5.1. Camada de Entidades 69
5.2. Camada de Persistência 70
5.3. Camada de Negócio 71
5.4. Camada de Apresentação 73
5.4.1. Navegabilidade 73
5.4.2. Gerenciamento de Categorias 74
5.4.3. Gerenciamento de Pontos 76
5.4.4. Pesquisa 77
5.5. Base de Dados 77
5.5.1. Caso Performático 78
6. CONSIDERAÇÕES FINAIS 79
6.1. Dificuldades Encontradas 79
6.2. Resultados Obtidos 80
6.3. Conclusões 81

6.4. Sugestões de Trabalhos Futuros 82
Índice Remissivo 83
REFERÊNCIAS BIBLIOGRÁFICAS 85
ANEXO A 86

Índice de Imagens

Figura 1 - Protótipo 14
Figura 2 - Segmentações do Sistema de Posicionamento Global 18
Figura 3 - Plano de Satélites do Sistema 19
Figura 4 - Tipos de Satélite de GPS 20
Figura 5 - Transmissão do sinal dos satélites de GPS 22
Figura 6 - Estações de Monitoramento 23
Figura 7 - Aplicações do GPS 24
Figura 8 - Triangulação com um satélite 25
Figura 9 - Triangulação com dois satélites 26
Figura 10 - Triangulação com três satélites 26
Figura 11 - Sistema de coordenadas (modelo matemático) 31
Figura 12 - Representação do conceito de triangulação 34
Figura 13 - Diagrama de funcionamento do receptor de GPS 35
Figura 14 - Aparelhos receptores de GPS 37
Figura 15 - Precisão na distância entre satélites 38
Figura 16 - Volume do espaço criado pelos satélites 38
Figura 17 - Erro de Multi-Trajeto 39
Figura 18 - Erro devido ao meio de transmissão 40
Figura 19 - Meridianos 42
Figura 20 - Paralelos 43
Figura 21 - Latitude 43
Figura 22 - Longitude 44
Figura 23 - Determinação da distância entre dois pontos 45
Figura 24 - Raio da Terra 46
Figura 25 - Teodolito: aparelho usado para medir ângulos 47
Figura 26 - Achando o raio da Terra 48
Figura 27 - Relações de trigonometria plana 50
Figura 28 - Demonstração de trigonometria 51
Figura 29 - Triângulo Esférico A B C 52
Figura 30 - Representação plana do triângulo esférico 53
Figura 31 - Determinação do ângulo de deslocamento 55

Figura 32 - Sentido de deslocamento 56
Figura 33 - Determinação de reta 57
Figura 34 - Calculo do sentido de deslocamento 58
Figura 35 - Velocidade de deslocamento 59
Figura 36 - Tempo de Chegada 60
Figura 37 - Diagrama de blocos e fluxo de dados 61
Figura 38 - Garmin eTrex Legend 62
Figura 39 - Conversor USB - Serial 65
Figura 40 – Notebook 65
Figura 41 - Modelo de classes, Entidades. 70
Figura 42 - Modelo de classes, Persistência. 71
Figura 43 - Modelo de classes, Negócio. 72
Figura 44 - Tela Principal do SpeakPosition 74
Figura 45 - Tela Principal de Categorias 75
Figura 46 - Tela de detalhamento de categorias 75
Figura 47 - Gerenciamento de Pontos 76
Figura 48 - Detalhamento de pontos 77
Figura 49 - Modelo de Dados 78
Figura 50 - Ambiente de teste 80

Índice de Tabelas

Tabela 1 - Situação atual dos satélites que compõem o sistema GPS 21
Tabela 2 - Exemplo real (efemérides) 33
Tabela 3 - Exemplo real (tempos de atraso) 33
Tabela 4 - Especificações Técnicas do Aparelho GPS 63
Tabela 5 - Pinos de comunicação serial..... 63
Tabela 6 - Especificação técnicas do notebook 66
Tabela 7 - Mapeamento de Navegação 74

Índice de Equações

Equação 1 - Interseção entre os planos $S_1 \cap S_2$, $S_1 \cap S_3$ e $S_1 \cap S_4$ t 28
Equação 2 - Latitude no ponto 31
Equação 3 - Longitude no ponto 32
Equação 4 - Raio da Terra 49
Equação 5 - Lei dos co-senos para triângulos esféricos 54
Equação 6 - Lei dos Co-senos 56
Equação 7 - Velocidade média 58
Equação 8 - Velocidade instantânea 58
Equação 9 - Tempo de chegada 60

Glossário de Termos e Abreviaturas

GPS - Global Positioning System (Sistema de Posicionamento Global)

SVN - Space Vehicle Number (Número do veículo espacial)

NAVSTAR - Navigation System with Time and Ranging (Systema de Navegação com Tempo e Variação)

PRN - Pseudo-Random-Noise (Ruído falsamente aleatório)

SVID (Space Vehicle Identification identificação do veículo espacial)

DoD - Department of Defense (Departamento de Defesa dos Estados Unidos da América)

FOC - Full Operational Capability (Capacidade Operacional Máxima)

USB - Universal Serial Bus

HD – Hard Disk (Disco Rígido)

TTS - Text-to-Speech (Texto para Fala)

SQL - Structured Query Language (Linguagem de Consulta Estruturada)

Lista de Símbolos

ϕ – Latitude de um ponto

λ - Longitude de um ponto

α – Ângulo entre dois pontos em relação ao centro da Terra

N – Pólo Norte

S – Distância entre dois pontos.

t – Medida de tempo

β – Ângulo interno entre o sentido de deslocamento e o sentido em linha reta para o ponto desejado.

γ - Ângulo externo entre o sentido de deslocamento e o sentido em linha reta para o ponto desejado

1. INTRODUÇÃO

A engenharia da computação é baseada em uma sólida formação técnico-científica e profissional geral capaz de absorver e desenvolver novas tecnologias, estimulando a sua atuação crítica e criativa na identificação e resolução de problemas, considerando seus aspectos político-econômicos, sociais, ambientais e culturais, com visão ética e humanística, em atendimento às demandas da sociedade.

O Engenheiro de Computação é um profissional com formação plena em Engenharia, preparado em assuntos de computação para especificar, conceber, desenvolver, implementar, adaptar, produzir, industrializar, instalar e manter sistemas computacionais, bem como perfazer a integração de recursos físicos e lógicos necessários para o atendimento das necessidades informacionais, computacionais e da automação de organizações em geral.

Um dos diversos campos de atuação na área de Engenharia da Computação é o de prover informações cada vez mais rápidas e precisas, requerendo uma gama de componentes e soluções disponibilizados nas mais diversas aplicações. Outro campo seria as aplicações de localização e mapeamento de regiões.

Como toda evolução deve ser acompanhada, por todos que estão ligados, diretamente ou indiretamente a área quer seja ele estudante, técnico ou engenheiro e por ser também a informação um ramo imenso e de intenso trabalho, além do conhecimento que será adquirido, o aluno autor optou em desenvolver um projeto final ligado a esta área tendo como gratificação maior a elaboração de uma aplicação voltada às pessoas menos favorecidas de uma sociedade, os deficientes visuais.

O projeto parte da idéia de utilizar um receptor GPS, trabalhar em cima dos dados de latitude e logitude extraídos do mesmo e envialos a um notebook, interpretar e gerar várias informações para retransmitir ao usuário de forma sonora e concisa.

1.1. Contextualização do Problema

Um dos maiores problemas para um deficiente visual se dá na locomoção em uma cidade, seja ela grande ou pequena. Com o grande número de lugares e grades distâncias, um deficiente visual tem bastante dificuldade em armazenar tantas informações.

Para se locomover, um deficiente, quando sozinho, fica perguntando a diversas pessoas qual sua localidade, a distância até onde ele gostaria de ir e isso leva ao deficiente se sentir incapacitado de realizar simples atividades corriqueiras.

1.2. Objetivo do Projeto

Criar um sistema utilizando uma tecnologia já existente que é o GPS (Global Positioning System - Sistema de Posicionamento Global) aplicando respostas sonoras, que auxiliará pessoas com serias deficiências visuais, facilitando de forma simples de operação sua locomoção principalmente em cidades. O mesmo indicará a localização da pessoa assim como o melhor caminho para chegar a seu destino, distância entre pontos, destino e outras funcionalidades.

O sistema utilizará algoritmos de deslocamento, velocidade e posicionamento tendo como foco principal a precisão do local onde os deficientes desejam ir, assim como a rota a ser seguida. Com isso, aumenta sua qualidade de vida, dando-os uma nova maneira de reconhecer que o problema de locomoção pode ser minimizado. Tornando menos dependentes e mais confiantes para conquistar seus outros objetivos de vida. A Figura 1 mostra a arquitetura de hardware do protótipo desenvolvido neste projeto.

Figura 1 - Protótipo

1.3. Motivação

> *"Ao explicar direções para uma pessoa cega, seja o mais claro e específico possível. Não se esqueça de indicar os obstáculos que existem no caminho que ela vai seguir. Como algumas pessoas cegas não têm memória visual, não se esqueça de indicar as distâncias em metros (por exemplo: "uns vinte metros para a frente"). Mas se você não sabe corretamente como direcionar uma pessoa cega, diga algo como "eu gostaria de lhe ajudar, mas como é que devo descrever as coisas?", ele(a) dirá"* Texto extraído do portal da Associação de Cegos Louis Braille.

A sociedade sempre teve a postura de manter preconceito contra os deficientes, de uma maneira geral estes são tratados de incapacitados. Eles precisão encontrar uma forma de contornar esses preconceitos, encontrar uma maneira que não os deixem desanimar para os problemas impostos por sua distinção social. O objetivo é ajudar os deficientes visuais a mudar a situação atual e provar que é possível ver através da escuridão.

É com o intuito de desenvolver algo a sociedade menos favorecida que vem a motivação de fazer alguma utilidade para os mais necessitados, aplicar todo o conhecimento adquirido de forma ética, proveitosa, e para o próximo, traz bastante satisfação e sensação de tarefa cumprida.

1.4. Estrutura do Trabalho

Além deste capítulo introdutório, este trabalho está estruturado em seis capítulos assim distribuídos:

No **Capítulo 2** é apresentada uma introdução teórica sobre funcionamento do GPS, onde são abordados os conceitos essenciais para o entendimento do funcionamento do sistema e, principalmente, do software desenvolvido.

No **Capítulo 3** é apresentado como funciona o sistema de coordenadas e os cálculos com latitude e longitude, informações estas vindas do GPS. São detalhadas o calculo de distância entre dois pontos, o ângulo de deslocamento para melhor rota, a velocidade de deslocamento e o de tempo de chegada.

No **Capítulo 4** é detalhada a construção do protótipo, descrevendo as etapas e justificativas de montagem. São também caracterizados os principais componentes que compõem o protótipo e o papel de cada um no sistema como um todo assim como especificadas as tecnologias utilizadas e os motivos de escolha de cada uma.

No **Capítulo 5** é apresentada a construção do sistema de deslocamento, responsável por guiar e dar as mais diversas informações. Este capítulo contém ainda a descrição das funcionalidades principais dos módulos da aplicação.

Por fim, no **Capítulo 6** são apresentadas as considerações finais sobre o trabalho, contendo as principais conclusões, os resultados obtidos, as dificuldades encontradas e as sugestões para trabalhos futuros.

2. CONCEITOS BÁSICOS SOBRE O FUNCIONAMENTO DO GPS

Neste capítulo são abordados conceitos teóricos sobre o funcionamento do GPS, informações estas pertinentes ao entendimento do projeto desenvolvido. Embora alguns dos conceitos citados envolvam desenvolvimentos matemáticos, demandando um detalhamento mais extenso do tema para sua completa compreensão, apenas as características principais serão apresentadas, visando manter o foco de interesse do presente trabalho que, embora sirva-se destes conceitos, não tem seu foco principal centrado neles.

2.1. O Sistema de Posicionamento Global (GPS)

O Sistema GPS é um sistema de posicionamento global que utiliza a tecnologia de satélites para determinar a localização de um ponto em qualquer parte do planeta e a qualquer momento.

Sem dúvida a utilização de GPS irá dominar a navegação do futuro, pois seus objetivos na essência visam atender a todas as necessidades de deslocamento, sem um alto custo para isso. Oferece uma navegação em tempo real, que são informações corrigidas a todo o instante, alta imunidade à interferências causadas pelos meios de propagação das ondas e uma cobertura em todo o globo terrestre, sem a necessidade de sintonização de estações de rádio (o sistema possui uma busca por satélites, sintonizando-os automaticamente). Em resumo, baixo custo de

aquisição, alta precisão (margem de erro muito pequena), fácil operação, são as facilidades que cumprem o objetivo do sistema.

2.2. Histórico

Em 1957 os russos lançaram um satélite chamado Sputinik I, marcando o início da utilização de satélites para o posicionamento geodésico. Os americanos em 1958 lançaram o Vanguard tendo assim o início do desenvolvimento do sistema Navstar (Navigation satellite with Timing and Ranging) e por volta de 1967 é que foi liberado para uso civil, o sistema denominado Navy Navigation Satellite System (NNSS) também chamado de Transit.

Em 1973 iniciou-se o desenvolvimento do Global Positioning System (GPS), o sistema foi projetado pelo Departamento de Defesa dos Estados Unidos da América (EUA) fornecendo a posição instantânea, assim como a velocidade e o horário de qualquer ponto existente na superfície da terra (LETHAM, 1996).

Esse processo de desenvolvimento se deu em 3 fases;

A primeira fase com a autorização e o inicio do desenvolvimento. Essa fase durou seis anos e teve como objetivo os estudos iniciais, estudos de desempenho do sistema e o de viabilidade do projeto,

A segunda fase marca o desenvolvimento e testes de equipamentos GPS durando até meados de 1985.

A terceira e última fase caracteriza-se pela produção de receptores GPS e a implementação da rede de satélites, finalizando com 24 satélites e 6 órbitas e proporcionando cobertura completa em todo o globo terrestre. Conhecido como FOC (Full Operational Capability), pelo fato de todos os satélites estarem funcionando simultaneamente.

O propósito inicial do sistema de navegação era apenas para o uso das forças armadas americanas resultante da uma união das forças da Marinha e da Força Aérea. Em função da grande eficiência e utilidade proporcionada pelos receptores de GPS, uma grande comunidade usuária emergiu dos mais variados segmentos da comunidade civil (navegação, posicionamento geodésico, agricultura, controle de frotas etc.) (MONICO, 2000).

Hoje em dia seus objetivos vão além do simples uso em missões de guerra, pode-se dizer que esse sistema difundiu-se tanto em outras áreas que seu uso faz parte de um cotidiano regular.

2.3. Segmentos do sistema

O Sistema de posicionamento Global (GPS) divide-se em três segmentos; O segmento espacial, segmento de controle e segmento de usuário como mostra a Figura 2.

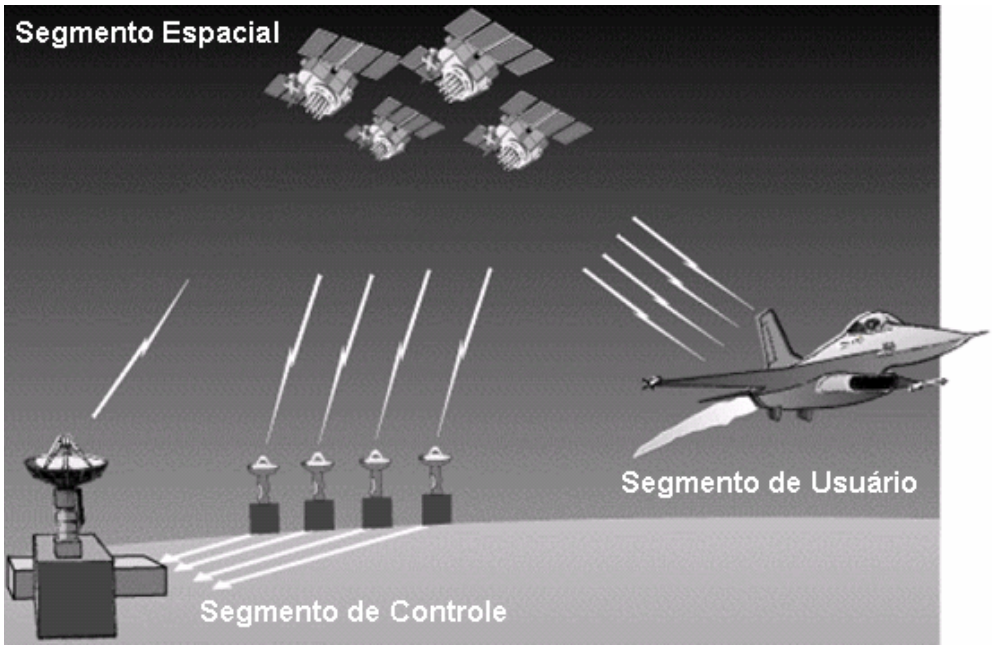


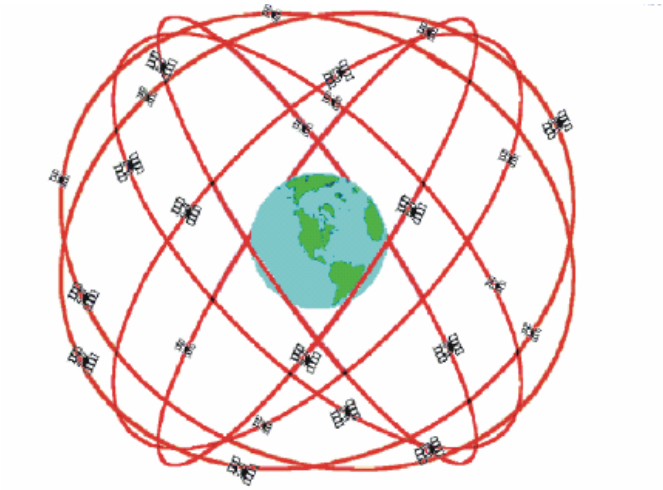
Figura 2 - Segmentações do Sistema de Posicionamento Global

2.3.1. Segmento espacial

O segmento espacial consiste em satélites que circulam ao redor da Terra distribuídos em seis órbitas distintas, a uma altitude de 20.200 km[1], cada órbita pos-

[1] Uma distância pouco menor que o dobro do diâmetro da Terra.

sui quatro satélites. Os planos orbitais[2] estão com uma inclinação de 55º em relação ao Equador e o período de revolução de 12 horas siderais e de 11 horas e 58 minutos terrestres, ilustrado na Figura 3. Dessa forma, a posição dos satélites se repete, a cada dia, 4 minutos antes que a do dia anterior.



Bloco de 24 (+4) satélites
6 Planos Orbitais
Altitude Média de 20.200 km
Inclinação Orbital de 55º (Equador)
Período de 11h57'58.3" (tempo sideral)

Figura 3 - Plano de Satélites do Sistema

Essa configuração garante que, no mínimo, quatro satélites GPS sejam visíveis em qualquer local da superfície terrestre ou acima dela, a qualquer hora do dia (MONICO, 2000).

O projeto NAVSTAR-GPS possui quatro blocos de satélites denominados Bloco I, II, IIA, IIR e IIF. Os satélites do Bloco I foram desativados em 1995, e os satélites dos Blocos II e IIA, são compostos por 28 satélites, os quais se referem, respectivamente, a primeira e segunda geração de satélites GPS (MONICO, 2000).

Os satélites do Bloco IIA, têm comunicação recíproca e uma maior capacidade de armazenamento de dados de navegação. Já estão sendo substituídos pelos

[2] Plano imaginário que contém a órbita do satélite. Este plano forma um ângulo de 55º com o plano que contém o Equador.

do Bloco IIR, que pertencem à terceira geração de satélites, cujas características principais, que os diferem dos Blocos anteriores, é que são capazes de medir distâncias entre eles e ainda calcular os erros no próprio satélite, transmitindo essas informações entre os satélites e para o sistema de controle em Terra (SEEBER, 1993).

Uma quarta geração irá substituir os satélites do Bloco IIR. Estes satélites denominados de Bloco IIF será composto por 33 satélites que deverão incorporar a modernização do sistema. Os satélites do grupo IIR carregam padrões de freqüência altamente estáveis, oriundos de osciladores atômicos de césio e rubídio e os do Bloco IIF poderão carregar osciladores Maser de hidrogênio, considerados até o momento os melhores osciladores (MONICO, 2000). A Figura 4 mostra os satélites dos blocos I, II, IIA e IIR.

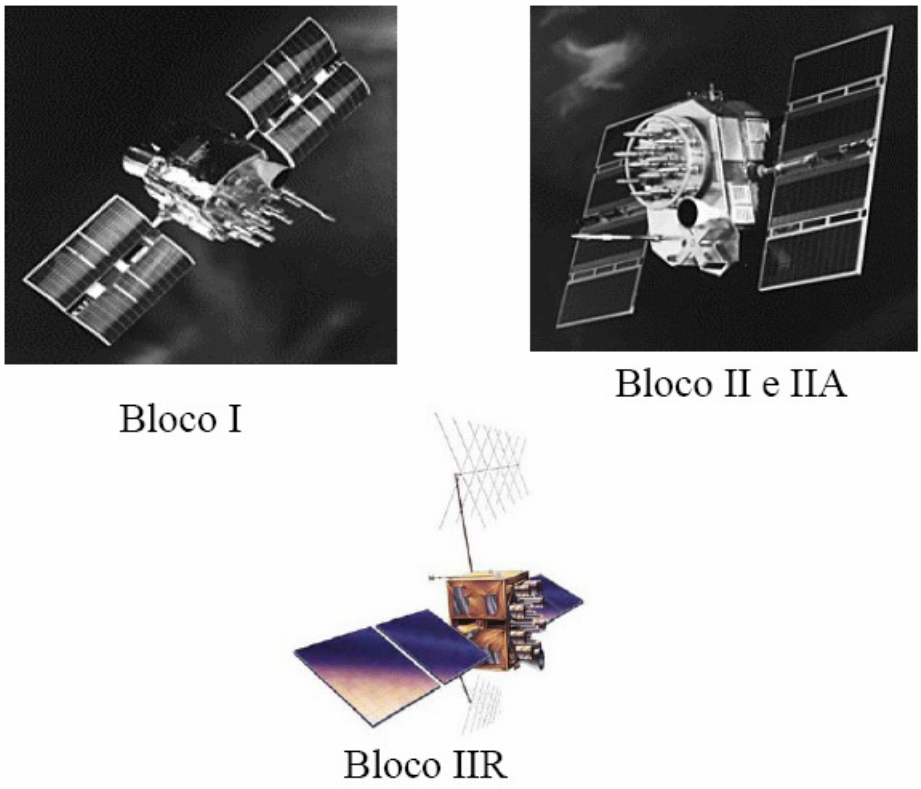


Figura 4 - Tipos de Satélite de GPS

A situação atual dos satélites que compõem o sistema é mostrada na Tabela 1.

Plano/Órbita	1	2	3	4	5(reservas)

A	IIA-21 (39)[9]	IIA-12 (25)[25]	IIA-28 (38)[8]	IIA-15 (27)[27]	II-4 (19)[19]
B	IIA-18 (22)[22]	IIA-27 (30)[30]	II-2 (13)[2]	IIA-22 (35)[5]	IIR-5 (44)[28]
C	IIA-24 (36)[6]	IIA-25 (33)[3]	IIA-19 (31)[31]	IIA-20 (37)[7]	
D	IIA-11 (24)[24]	IIR-3 (46)[11]	II-5 (17)[17]	IIA-23 (34)[4]	II-9 (15)[15]
E	IIR-4 (51)[20]	II-8 (21)[21]	IIA-26 (40)[10]	IIR-7 (54)[18]	IIA-10 (23)[23]
F	IIR-6 (41)[14]	IIA-14 (26)[26]	IIR-2 (43)[13]	IIA-16 (32)[1]	IIA-17 (29)[29]

Tabela 1 - Situação atual dos satélites que compõem o sistema GPS

Fonte:<http://www.spaceandtech.com/spacedata/constellations/navstar-gps_consum.shtml> em 20/10/2007

Obs: Bloco - Seqüência de lançamento, (SVN)-Space Vehicle Number, número do veículo espacial, [PRN]-Pseudo-Random-Noise, ruído falsamente aleatório.

Da Tabela 1 apresentada, 3 satélites são do Bloco II, 16 são do Bloco IIA e 5 são do Bloco IIR, esta é a situação na data de 20/10/2007.

Os satélites são identificados por várias maneiras, o SVN, ou número NAVSTAR, o PRN ou SVID (Space Vehicle Identification identificação do veículo espacial) e número da posição orbital.

Os satélites do sistema GPS emitem ondas portadoras de 1575,42 Mhz e comprimento de onda $\lambda \cong 19$ cm (L1) e 1227,60 Mhz e comprimento de onda $\lambda \cong 24$ cm (L2). Modulados em fase portadoras, sendo os códigos +1 e –1, emitidos a freqüências de 1,023 Mhz (código C/A) e 10,23 Mhz (código P). O código C/A (Coarse/Acquisition Code) se repete a cada milissegundo, enquanto o código P (precision code), a cada 267 dias. Além desses códigos, existe ainda o código Y, sendo gerado, entretanto a partir de uma equação secreta (anti-spoofing). A portadora L1 é modulada com os códigos C/A e P (ou Y), enquanto L2 apenas com o código P (ou Y) (SEGANTINE, 1999). Segue abaixo na Figura 5 o esquema de envio de sinal.

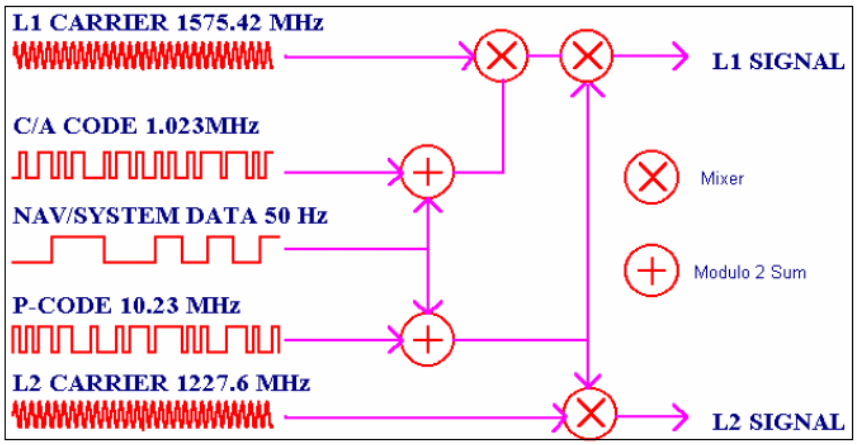


Figura 5 - Transmissão do sinal dos satélites de GPS

As pseudo-distâncias bem como as fases de ondas portadoras ou diferenças de fase medidas pelos sinais emitidos dos satélites são consideradas as observáveis básicas do GPS (SEEBER, 1993). Através das observáveis os receptores GPS convertem em posicionamento, velocidade e tempo estes sinais eletromagnéticos oriundos dos satélites, interpretando-os e convertendo em dados calculados no receptor de GPS.

2.3.2. Segmento de controle

As principais tarefas do sistema de controle são:

- Controlar continuamente o sistema de satélites
- Determinar o tempo GPS
- Calcular as correções dos relógios dos satélites
- Predizer as efemérides[3] dos satélites
- Atualizar periodicamente as mensagens de navegação de cada satélite

[3] Efemérides, referente a satélites de GPS, consiste em uma seqüência de dados transmitidos dos satélites para a Terra com o objetivo de informar aos usuários sobre a via útil e posição dos satélites.

Esse sistema é composto por cinco estações de monitoramento mundial localizadas nos locais: Hawai (EUA), Atol Kwajalein (Oceano Pacífico Norte), Ilha de Ascension (Oceano Atlântico Sul), Ilha de Diego Garcia (Oceano Indico Sul) e Colorado Springs (EUA); três delas com capacidade de transmitir os dados para os satélites (Ilha Ascension, Ilha de Diego Garcia e Atol de Kwajalein); e uma estação de controle central (Master Control Station) localizada em Colorado Springs. Segue na Figura 6 que mostra suas localidades no planeta.

As cinco estações de monitoramento pertencem à American Air Force (ver Figura 6). Em conjunto com as sete estações do National Imagery and Mapping Agency, compõem as estações monitoras GPS do DoD. Todas são equipadas com relógios atômicos, receptores de dupla freqüência e múltiplos canais. Tendo a capacidade de produzir efemérides com precisão da ordem de poucos centímetros em cada uma das coordenadas do satélite, permitindo assim atender à maioria das aplicações que exijam alta precisão.

Figura 6 - Estações de Monitoramento

Fonte: <http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html>

2.3.3. Segmento de usuários

O segmento de usuários consiste nos receptores GPS utilizados para as mais diversas atividades. Este segmento pode ser dividido em civil e militar, sendo que para o uso civil existe restrição quanto à precisão. Os militares americanos fa-

zem uso dos receptores GPS para cálculos de posições e deslocamentos quando realizam manobras de combate e de treino.

Existem equipamentos para usuários autorizados, que permitem a obtenção de uma precisão da ordem de milímetros, tanto na horizontal quanto na vertical. Estes receptores são utilizados em estudos geodésicos. Abaixo segue alguns exemplos de segmentos de usuários civis e militares e ilustrados na Figura 7.

- Navegação para aviões, carros, navios e outros meios de transporte.
- Posicionamento de objetos e dados no espaço
- Movimentos de placas tectônicas
- Esportes
- Correção geométrica de fotos aéreas e imagens de satélite

Figura 7 - Aplicações do GPS

2.4. Funcionamento do sistema

Os satélites GPS emitem dois sinais nas portadoras (L1 e L2). Uma consiste na mensagem de navegação, bits de dados que contêm os valores das órbitas, dados para correção da propagação do sinal na atmosfera, valores para correção de

erros dos relógios dos satélites, etc. Essas informações contidas nos sinais são determinadas pelo segmento de controle do GPS em terra.

O segundo tipo de informação é um conjunto de ruídos aleatórios (PRN), impulsos digitais em um padrão inconfundível. Os valores de transmissão não são informações relevantes dados no sentido de transmissão de informações. Os códigos são enviados para permitir que a unidade que está recebendo o sinal possa medir o tempo exato em que o sinal atrasa para chegar ao receptor. Cada sinal enviado é diferente uns dos outros, possuem uma assinatura digital específica, permitindo assim existir distinção dos satélites transmissores.

Os receptores possuem por volta de 12 canais de recepção. Cada sinal de código PRN é alocado em um canal, o receptor tem a capacidade então de receber simultaneamente 12 sinais de satélites distintos.

O receptor tem a função de calcular a distância entre o receptor e o satélite transmissor do sinal, esse calculo e feito pelo tempo em que o sinal leva para sair do transmissor no satélite e chegar até o receptor.

Considerando um único satélite, o ponto a ser encontrado pode estar em qualquer lugar da superfície de uma esfera imaginária onde o raio se dá pelo tempo de atraso do sinal multiplicado pela velocidade da luz[4], mostrado na Figura 8.

Raio = distancia do satélite até o receptor de GPS.

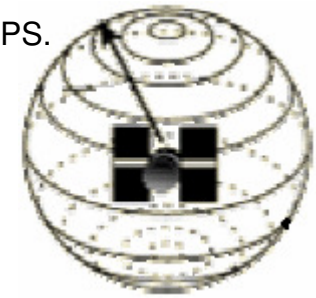


Figura 8 - Triangulação com um satélite

Com dois satélites o ponto a ser encontrado pode estar na linha de interseção das duas esferas geradas pelas distâncias dos satélites, ver Figura 9.

---

[4] A velocidade da luz é igual a $2{,}99792458.10^{8}$ metros por segundo. Fonte: Revista do Professor de Matemática 59, 2006 pag. 25.

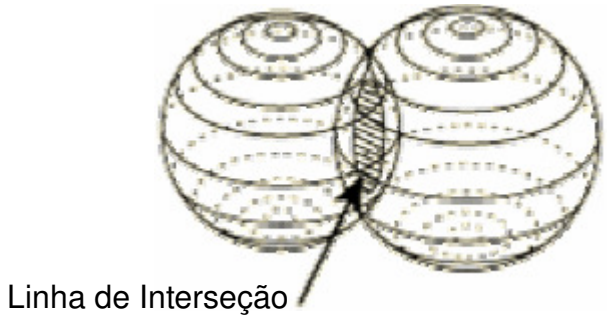


Figura 9 - Triangulação com dois satélites

Por fim com três satélites é possível determinar dois pontos na interseção dos três, para definir entre esses dois é necessário um quarto satélite, mas normalmente um desses pontos é impossível (longe da Terra), quatro satélites são necessários para encontrar o ponto de recepção do GPS, ver a intersecção de três satélites na Figura 10. Será mostrado matematicamente e dado alguns exemplos na seção seguinte.

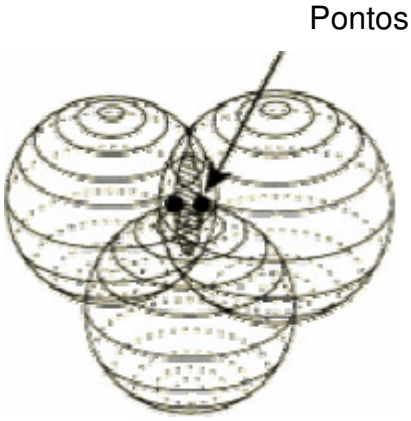


Figura 10 - Triangulação com três satélites

2.4.1. Fundamentação matemática

Como dito anteriormente, as distâncias entre satélites de transmissão e os receptores de GPS são calculadas com referência ao tempo em que o sinal leva entre a emissão e a recepção desse sinal. Considerando que a distância seja o raio de uma superfície esférica e o satélite transmissor esteja no centro da mesma é possível determinar um único ponto de interseção.

Teorema 2.1: Se quatro superfícies esféricas se intersectam e seus centros não são coplanares, então essa interseção consiste em um único ponto.

Demonstração (Revista do Professor de Matemática 59, 2006):

Tem-se então quatro superfícies esféricas $S_1, S_2, S_3 \text{ e } S_4$ e com centros $C_1, C_2, C_3 \text{ e } C_4$. O objetivo é determinar como encontrar o ponto de interseção $P \in S_1 \cap S_2 \cap S_3 \cap S_4$ considerando $C_1, C_2, C_3 \text{ e } C_4$ não coplanares.

Considerando

$$x^2 + y^2 + z^2 + a_j x + b_j y + c_j z + d_j = 0 \quad \text{(i)}$$

equações gerais de S_j e que $j = 1,2,3,4$ tem que um ponto (x, y, z) estará sobre as quatro esferas se

$$\begin{cases} x^2 + y^2 + z^2 + a_1 x + b_1 y + c_1 z + d_1 = 0 \\ x^2 + y^2 + z^2 + a_2 x + b_2 y + c_2 z + d_2 = 0 \\ x^2 + y^2 + z^2 + a_3 x + b_3 y + c_3 z + d_3 = 0 \\ x^2 + y^2 + z^2 + a_4 x + b_4 y + c_4 z + d_4 = 0 \end{cases}$$

Utilizando o sistema acima e subtraindo a primeira linha da segunda, a primeira da terceira e a primeira da quarta, elimina-se $x^2, y^2 \text{ e } z^2$ e passa-se a ter;

$$\begin{cases} (a_1 - a_2)x + (b_1 - b_2)y + (c_1 - c_2)z + (d_1 - d_2) = 0 \\ (a_1 - a_3)x + (b_1 - b_3)y + (c_1 - c_3)z + (d_1 - d_3) = 0 \\ (a_1 - a_4)x + (b_1 - b_4)y + (c_1 - c_4)z + (d_1 - d_4) = 0 \end{cases}$$

Um ponto que esteja ao mesmo tempo sobre as quatro esferas devem satisfazer simultaneamente as três equações acima. Cada equação separadamente tem o seguinte significado:

a) a primeira equação foi obtida subtraindo as equações da primeira esfera da segunda e assim ela mostra a equação do plano $S_1 \cap S_2$ que contém a interseção da primeira esfera com a segunda.
b) a segunda equação foi obtida subtraindo as equações da primeira esfera da terceira e assim ela mostra a equação do plano $S_1 \cap S_3$ que contém a interseção da primeira esfera com a terceira.
c) a terceira equação foi obtida subtraindo as equações da primeira esfera da quarta e assim ela mostra a equação do plano $S_1 \cap S_4$ que contém a interseção da primeira esfera com a quarta.

Considerando os planos $S_1 \cap S_2$, $S_1 \cap S_3$ e $S_1 \cap S_4$ se tem que, se $P(x, y, z)$ está em $S_1 \cap S_2 \cap S_3 \cap S_4$, então (x, y, z) é a solução do sistema apresentado abaixo.

$$\begin{cases} (a_1 - a_2)x + (b_1 - b_2)y + (c_1 - c_2)z = d_2 - d_1 \\ (a_1 - a_3)x + (b_1 - b_3)y + (c_1 - c_3)z = d_3 - d_1 \\ (a_1 - a_4)x + (b_1 - b_4)y + (c_1 - c_4)z = d_4 - d_1 \end{cases} \quad \text{(ii)}$$

Equação 1 - Interseção entre os planos $S_1 \cap S_2$, $S_1 \cap S_3$ e $S_1 \cap S_4$ t

Para provar o teorema é necessário ter uma única solução para o sistema. Caso contrário na interseção $S_1 \cap S_2 \cap S_3 \cap S_4$ existirá dois ou mais pontos distintos acarretando em duas ou mais soluções distintas do sistema.

Escrevendo o último sistema na forma matricial se tem

$$\begin{bmatrix} a_1 - a_2 & b_1 - b_2 & c_1 - c_2 \\ a_1 - a_3 & b_1 - b_3 & c_1 - c_3 \\ a_1 - a_4 & b_1 - b_4 & c_1 - c_4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_2 - d_1 \\ d_3 - d_1 \\ d_4 - d_1 \end{bmatrix}$$

Para mostrar que a interseção das esferas é um único ponto, basta mostrar que a matriz dos coeficientes acima possui determinante não nulo.

Sendo r um número real positivo e C um ponto fixado, o conjunto dos pontos do espaço cuja distância a C é igual a r é chamado superfície esférica S de centro C e raio r. Se $C_j = (u_j, v_j, w_j)$, então S_j é descrita como o conjunto dos pontos $P = (x, y, z)$ tais que

$$(x - u_j)^2 + (y - v_j)^2 + (z - w_j)^2 = r^2 \quad \text{(iii)}$$

A equação (iii) é denominada equação reduzida da esfera S_j. Assim, por exemplo, $(x+1)^2 + (y-2)^2 + (z)^2 = 4$ é a equação reduzida da superfície esférica de centro $C = (-1,2,0)$ e raio $r = \sqrt{4} = 2$.

Desenvolvendo os quadrados em (i), se obtém

$$x^2 - 2u_j x + u_j{}^2 + y_j{}^2 - 2yv_j + v_j{}^2 + z^2 - 2zw_j + w_j{}^2 = r^2 \quad \text{(iv)}$$

e reposicionando os termos se tem

$$x^2 + y^2 + z^2 - 2u_j x - 2v_j y - 2w_j z + u_j{}^2 + v_j{}^2 + w_j{}^2 = r_j{}^2$$

Sendo $C_j = (u_j, v_j, w_j)$ o centro de S_j, $j = 1,2,3,4$, então fica-se com

$$\begin{cases} x^2+y^2+z^2-2u_1x-2v_1y-2w_1z+{u_1}^2+{v_1}^2+{w_1}^2={r_1}^2 \\ x^2+y^2+z^2-2u_2x-2v_2y-2w_2z+{u_2}^2+{v_2}^2+{w_2}^2={r_2}^2 \\ x^2+y^2+z^2-2u_3x-2v_3y-2w_3z+{u_3}^2+{v_2}^2+{w_3}^2={r_3}^2 \\ x^2+y^2+z^2-2u_4x-2v_4y-2w_4z+{u_4}^2+{v_4}^2+{w_4}^2={r_4}^2 \end{cases}$$

Assim como antes, o objetivo é encontrar a diferença entre a primeira e a segunda, a primeira e a terceira e a primeira e a quarta linha do sistema. Se obtém

$$\begin{cases} (-2u_1+2u_2)x+(-2v_1+2v_2)y+(-2w_1+2w_2)z+{u_1}^2-{u_2}^2+{v_1}^2-{v_2}^2+{w_1}^2-{w_2}^2+{r_1}^2-{r_2}^2=0 \\ (-2u_1+2u_3)x+(-2v_1+2v_3)y+(-2w_1+2w_3)z+{u_1}^2-{u_3}^2+{v_1}^2-{v_3}^2+{w_1}^2-{w_3}^2+{r_1}^2-{r_3}^2=0 \\ (-2u_1+2u_4)x+(-2v_1+2v_4)y+(-2w_1+2w_4)z+{u_1}^2-{u_3}^2+{v_1}^2-{v_4}^2+{w_1}^2-{w_4}^2+{r_1}^2-{r_4}^2=0 \end{cases}$$

e reposicionando os termos se tem

$$\begin{cases} 2(u_2-u_1)x+2(v_1-v_1)y+2(w_2-w_1)z=-{u_1}^2+{u_2}^2-{v_1}^2+{v_2}^2-{w_1}^2+{w_2}^2+{r_2}^2-{r_1}^2 \\ 2(u_3-u_1)x+2(v_3-v_1)y+2(w_3-w_1)z=-{u_1}^2+{u_3}^2-{v_1}^2+{v_3}^2-{w_1}^2+{w_3}^2+{r_3}^2-{r_1}^2 \\ 2(u_4-u_1)x+2(v_4-v_1)y+2(w_4-w_1)z=-{u_1}^2+{u_3}^2-{v_1}^2+{v_4}^2-{w_1}^2+{w_4}^2+{r_4}^2-{r_1}^2 \end{cases} \quad \text{(v)}$$

comparando as equações (i) e (iv), tem-se $a_j=-2u_j, b_j=-2v_j, c_j=-2w_j$ de modo que os sistemas (v) e (ii), possuem a mesma matriz dos coeficientes e deste modo

$$\begin{vmatrix} a_1-a_2 & b_1-b_2 & c_1-c_2 \\ a_1-a_3 & b_1-b_3 & c_1-c_3 \\ a_1-a_4 & b_1-b_4 & c_1-c_4 \end{vmatrix} = \begin{vmatrix} 2(u_2-u_1) & 2(v_2-v_1) & 2(w_2-w_1) \\ 2(u_3-u_1) & 2(v_3-v_1) & 2(w_3-w_1) \\ 2(u_4-u_1) & 2(v_4-v_1) & 2(w_4-w_1) \end{vmatrix}$$

Onde $|.|$ significa determinante de (.). Como ao multiplicar uma linha por uma constante este fica multiplicado por esta constante resulta em

$$\begin{vmatrix} a_1-a_2 & b_1-b_2 & c_1-c_2 \\ a_1-a_3 & b_1-b_3 & c_1-c_3 \\ a_1-a_4 & b_1-b_4 & c_1-c_4 \end{vmatrix} = 2\begin{vmatrix} u_2-u_1 & v_2-v_1 & w_2-w_1 \\ 2(u_3-u_1) & 2(v_3-v_1) & 2(w_3-w_1) \\ 2(u_4-u_1) & 2(v_4-v_1) & 2(w_4-w_1) \end{vmatrix}$$

$$= 4\begin{vmatrix} u_2-u_1 & v_2-v_1 & w_2-w_1 \\ u_3-u_1 & v_3-v_1 & w_3-w_1 \\ 2(u_4-u_1) & 2(v_4-v_1) & 2(w_4-w_1) \end{vmatrix}$$

$$= 8\begin{vmatrix} u_2-u_1 & v_2-v_1 & w_2-w_1 \\ u_3-u_1 & v_3-v_1 & w_3-w_1 \\ u_4-u_1 & v_4-v_1 & w_4-w_1 \end{vmatrix}$$

Como C_1, C_2, C_3 e C_4 são não coplanares, segue que o último determinante é não nulo e, portanto, a Equação 1, vista em (ii) é um sistema linear com determinante não nulo, tendo assim uma única solução.

■

Evidentemente, o simples fato de o sistema linear ter uma única solução, o que equivale a dizer que os centros são não coplanares, não acarreta necessariamente que a intersecção das quatro superfícies esféricas consiste em um único ponto P. Em outras palavras, a hipótese $S_1 \cap S_2 \cap S_3 \cap S_4 \neq \phi$ é essencial para a validade do teorema.

Observar na situação real do receptor de GPS, essa hipótese é comprovada pela existência do próprio usuário e sua localização.

Exemplo 1: Considere, as superfícies esféricas:

$$S1 : \text{centro}(0,0,1) \text{ e raio } \sqrt{2};$$
$$S2 : \text{centro}(0,3,0) \text{ e raio } \sqrt{10};$$
$$S3 : \text{centro}(2,0,0) \text{ e raio } 1;$$
$$S4 : \text{centro}(0,0,0) \text{ e raio } 1;$$

Seus centros são não coplanares e o sistema linear, neste caso,

$$\begin{cases} 6y - 2z = 0 \\ 4x - 2z - 4 = 0 \\ -2z = 0 \end{cases},$$

tem como única solução $x = 1, y = 0 \text{ e } z = 0$

Uma verificação simples mostra que $P = (1,0,0)$ pertence simultaneamente à $S_1, S_2, S_3 \text{ e } S_4$ de modo que $S_1 \cap S_2 \cap S_3 \cap S_4 = \{(1,0,0)\}$

2.4.2. Coordenadas geográficas de um ponto no espaço

Fixando um sistema de coordenadas tendo como origem o centro da Terra O, o Eixo N apontando em direção ao pólo Norte, o plano Oxy o plano do equador com o eixo Ox positivo cortando o meridiano de Greenwich e o eixo Oy positivo cortando o meridiano de longitude de 90ºE.

Dado um ponto $P = (x, y, z)$ do espaço, sejam ϕ e λ as medidas dos ângulos representados na Figura 11:

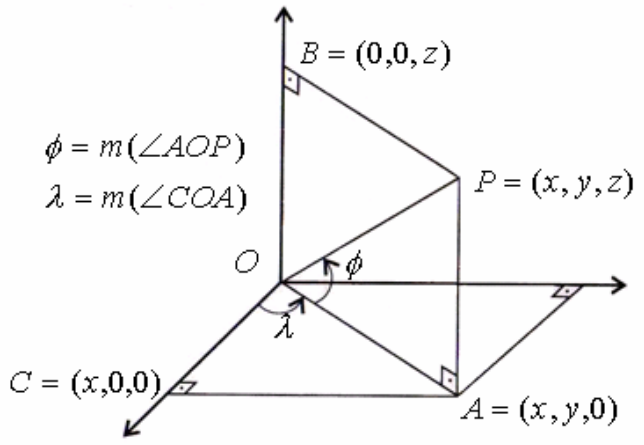


Figura 11 - Sistema de coordenadas (modelo matemático)

Quando P se encontra na superfície terrestre os ângulos ϕ e λ representam a latitude e longitude do ponto, respectivamente.

A distância entre $OP = d(OP) = \sqrt{x^2 + y^2 + z^2}$ é o que determina o raio da Terra, chamado de elevação do ponto.

A latitude, longitude e elevação que são as coordenadas geográficas serão relacionadas de diversas formas como mostra em seguida, analisando primeiramente o triângulo retângulo ΔOPB, obtém

$$\cos(90-\phi) = \frac{OB}{OP} = \frac{z}{\sqrt{x^2 + y^2 + z^2}} \text{ e,}$$

como $\cos(90-\phi) = \sin(\phi)$, segue que

$$\sin(\phi) = \frac{z}{\sqrt{x^2 + y^2 + z^2}}$$

Equação 2 - Latitude no ponto

.

Essa expressão limita o valor de ϕ um único valor entre 0 e 90 quando $z > 0$ e um único valor entre -90 e 0 quando $z < 0$. Quando positivo a latitude de P é $\phi^o N$ (norte), e quando negativo P é $\phi^o S$ (sul). Analisando agora o triângulo ΔOAC,

$$\sin\lambda = \frac{AC}{OA} = \frac{y}{\sqrt{x^2+y^2}} \text{ e } \cos\lambda = \frac{OC}{OA} = \frac{x}{\sqrt{x^2+y^2}}$$

Equação 3 - Longitude no ponto

.

Definindo um único λ entre 0 e 180, quando $y>0$ significa que P é $\lambda^o E$ (leste). Quando $y<0$, λ assume um valor único entre -180 e 0, e nisso a longitude de P é $(-\lambda^o)W$ (oeste).

Exemplo 2: Determinar as coordenadas geográficas de um ponto $P=\left(3\sqrt{3}.10^6,-3.10^6,6\sqrt{3}.10^6\right)$

Solução:

$$x^2+y^2+z^2 = 27.10^{12}+9.10^{12}+108.10^{12} = 144.10^{12}$$

e

$$x^2+y^2 = 27.10^{12}+9.10^{12} = 36.10^{12}.$$

Logo, $\sin(\phi) = \frac{6\sqrt{3}.10^6}{12.10^6} = \frac{\sqrt{3}}{2}$; portanto $\phi = 60$. Como

$$\sin\lambda = -\frac{3.10^6}{6.10^6} = -\frac{1}{2} \text{ e } \cos\lambda = \frac{3\sqrt{3}.10^6}{6.10^6} = \frac{\sqrt{3}}{2},$$

se obtém que $\lambda = -30$. Assim, as coordenadas de P são $\phi = 60^o N$ e $\lambda = 30^o W$.

Considerando o raio da Terra igual a $6{,}4.10^6$ metros, a elevação de P mede $12.10^6 - 6{,}4.10^6 = 5{,}6.10^6$ metros.

■

2.4.3. Exemplo real de triangulação

Aqui segue um exemplo real onde o usuário do GPS é visto por 4 satélites. Todas as informações transmitidas dos satélites ou interpretadas pelo receptor de GPS, requer uma precisão de dez ou mais casa decimais, pois como as distâncias são grandes, os erros seriam significativos após os cálculos efetuados para localização.

Exemplo 3: Considere a seguinte situação. Em princípio mostra-se uma tabela, Tabela 2 - Exemplo real (efemérides), com as efemérides enviadas pelos satélites tomados em metros ao sistema de coordenadas cartesianas.

	x	y	z
Satélite 1	$1{,}877191188.10^{6}$	$-1{,}064608026.10^{7}$	$2{,}428036099.10^{7}$
Satélite 2	$1{,}098145713.10^{7}$	$-1{,}308719098.10^{7}$	$2{,}036005484.10^{7}$
Satélite 3	$2{,}459587359.10^{7}$	$-4{,}336916128.10^{6}$	$9{,}090267461.10^{6}$
Satélite 4	$3{,}855818937.10^{6}$	$7{,}251740720.10^{6}$	$2{,}527733606.10^{7}$

Tabela 2 - Exemplo real (efemérides)

Encontrar a posição do ponto sobre a superfície terrestre a partir destes dados.

Solução: O GPS calcula e armazena também o tempo necessário (em segundo) entre a transmissão e a recepção dos sinais de cada satélite, ver Tabela 3.

Satélite 1	Satélite 2	Satélite 3	Satélite 4
0.08251731391	0.07718558331	0.06890629029	0.07815826940

Tabela 3 - Exemplo real (tempos de atraso)

Multiplicando cada atraso contido na Tabela 3 com a velocidade da luz ($2{,}99792458.10^{8}$ metros por segundo), se tem a distância entre o receptor e cada satélite, isso permite escrever as equações reduzidas das superfícies esféricas imaginárias e raios iguais às distâncias calculadas.

$$S_1 : (x-1{,}8.10^{6})^2 + (y+10{,}6.10^{6})^2 + (z-24{,}2.10^{6})^2 = 611{,}9.10^{12}$$
$$S_2 : (x-10{,}9.10^{6})^2 + (y+13.10^{6})^2 + (z-20{,}3.10^{6})^2 = 535{,}4.10^{12}$$
$$S_3 : (x-24{,}5.10^{6})^2 + (y+4{,}3.10^{6})^2 + (z-9.10^{6})^2 = 426{,}7.10^{12}$$
$$S_4 : (x-3{,}8.10^{6})^2 + (y-7{,}2.10^{6})^2 + (z-25{,}2.10^{6})^2 = 549{,}10.10^{12}$$

Desenvolvendo os quadrados, se tem as equações gerais e o sistema linear é dado por:

$$\begin{cases} 18{,}2x - 4{,}88y - 7{,}84 - 76{,}52.10^6 = 0 \\ 45{,}43x + 12{,}61y - 30{,}38 - 185{,}23.10^6 = 0 \\ 3{,}95x + 35{,}79y + 1{,}99 - 62{,}95.10^6 = 0 \end{cases},$$

tendo como solução única $x = 0{,}5660.10^7$, $y = 0{,}0978.10^7$ e $z = 0{,}2775.10^7$.

■

O ponto P com essas coordenadas cartesianas pertence simultaneamente às quatro superfícies esféricas imaginárias. Calculando como no item anterior, são

Latitude: $\phi = 26^\circ N$; Longitude: $\lambda = 10^\circ E$; Elevação: $919{,}71$ metros. Representado pela cidade de Djanet, localizada nos Montes Tássili, na fronteira entre a Argélia e a Líbia.

Uma ilustração de como ocorre a localização de um ponto na superfície terrestre pode ser visto na Figura 12.

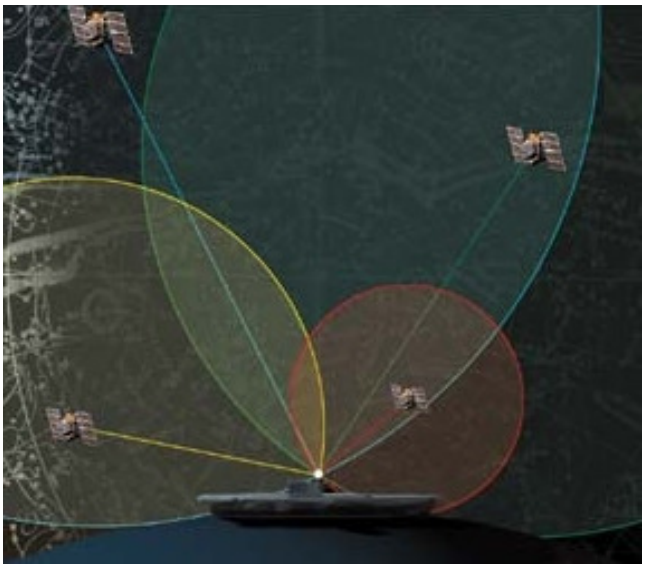


Figura 12 - Representação do conceito de triangulação

Fonte: <http://www.geodesia.org>

Um dos grandes desafios é implantar nos receptores relógios precisos, como não existem relógios perfeitos, os receptores GPS necessitam também identificar uma quarta incógnita: a diferença entre o relógio de baixo custo no implantado no receptor e o horário na rede GPS. O horário da rede é controlado com precisão até um bilionésimo de segundo por relógio atômico, mas o relógio do receptor pode estar

sujeito a erros significativos, o erro de milésimos de segundos já resulta em grandes distâncias na localização do ponto.

2.5. Receptores GPS

Os receptores GPS são os equipamentos responsável por receber os sinais dos satélites, calculam o tempo de retardo dos sinais dos satélites determinando as pseudo-distâncias, como dito anteriormente, determinando dessa forma a localização do mesmo. As diversas funcionalidade variam nos mais diversos modelos de receptores de GPS mas o princípio básico é o calculo de posição tridimensional, latitude, longitude e altitude.

A Figura 13 mostra o modelo simples e utilizado por diversos receptores de GPS.

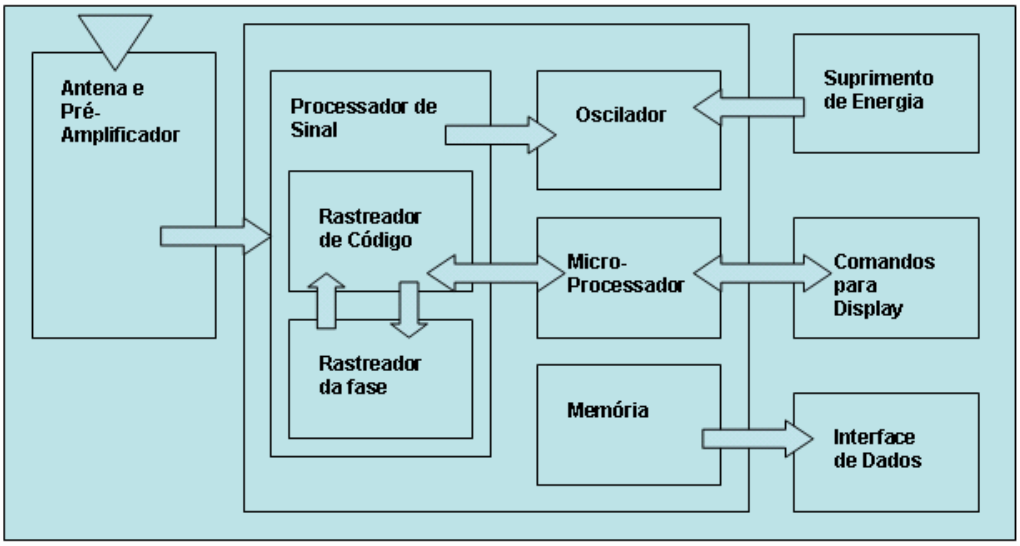


Figura 13 - Diagrama de funcionamento do receptor de GPS

A antena recebe ondas eletromagnéticas transmitidas pelos satélites, converte a energia em sinais elétricos, amplifica o sinal e o envia para a parte eletrônica do receptor, o processador de sinais.

Os sinais que entram no receptor são convertidos para uma freqüência mais baixa, chamada de freqüência intermediária (FI). Esta redução de freqüência é feita pela combinação da onda com um sinal senoidal gerado pelo oscilador sendo utilizado normalmente de quartzo.

Após isso são alocados em canais de recepção, canais são considerados as unidades eletrônicas primordiais, sendo divididos em multicanais (canais dedicados), seqüenciais e multiplexados. Cada canal rastreia continuamente um dos satélites visíveis, sendo no mínimo necessários quatro canais para se obter posição e correção do relógio em tempo real.

O microprocessador é necessário para processar o sinal, interpretar e decodificar a mensagem de navegação, assim como calcular posições e velocidades. A informação obtida é transferida para o display onde o usuário pode fazer visualizar todas as funções do receptor.

A interface com o usuário é feita por meio de comandos, teclas, botões e do display que proporcionam as interações entre ambas as partes. A maioria dos receptores dispõe de um padrão de operação preestabelecido, não requerendo intervenção direta do usuário.

Memória para armazenamento também são implantadas nos receptores, responsáveis por armazenar dados sobre as posições, distância e outros dados, que são capazes de serem descarregados através de uma porta serial do tipo RS-232, ou outra forma de interface com o exterior.

A energia para os receptores foi considerada um fator muito importante, mas com o avanço e a diminuição do consumo de energia dos receptores isso tornou um fator não muito relevante. Alguns aparelhos funcionam com pilhas, e alguns com baterias internas recarregáveis.

Os receptores GPS podem ser classificados de três formas segundo critérios específicos: para uso da comunidade usuária militar ou civil; para aplicação em navegação, geodésia[5] e uso direto em Sistemas de Informações Geográficas e, por último, segundo os diferentes tipos de receptores e dados proporcionados como é o caso dos receptores com código C/A; código C/A e portadora L1; código C/A e portadoras L1 e L2; código C/A e P e portadoras L1 e L2; portadora L1 e portadoras L1 e L2. Estas divisões ajudam os usuários na identificação do receptor adequado às suas necessidades, independentemente da classificação adotada. A Figura 14 mostra alguns tipos de receptores de GPS.

---

[5] Geodésia é a ciência que se ocupa da determinação da forma, das dimensões e do campo de gravidade da Terra. Fonte: <www.ibge.gov.br>

Figura 14 - Aparelhos receptores de GPS

2.6. Precisão do Posicionamento GPS

Os erros referentes à localização do ponto dependem do numero e da geometria dos satélites vistos no momento da captura do sinal e também a precisão da pseudo-distância do receptor ao satélite.

Supondo que quatro satélites estejam muito próximos, a medição da distância pode resultar em um grande erro para posição calculada. Mas se observarmos diversos satélites espalhados no momento da medição, o erro certamente será bem pequeno.

Para perceber melhor o efeito da geometria dos satélites na precisão do posicionamento a Figura 15 ilustram esses efeitos, logo depois um tetraedro é formado por linhas que ligam o receptor a cada satélite usado ilustrado na Figura 16.

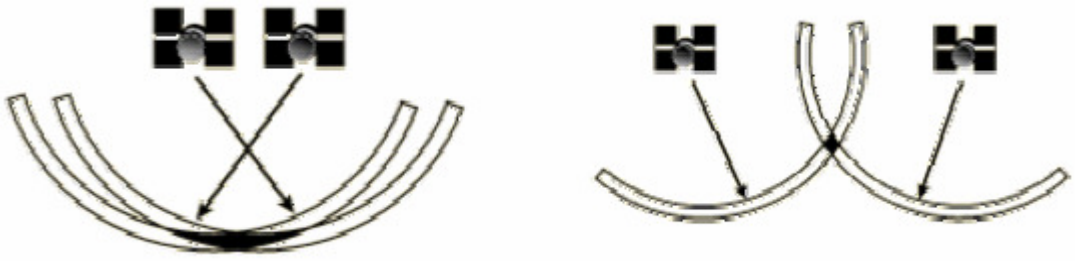


Figura 15 - Precisão na distância entre satélites

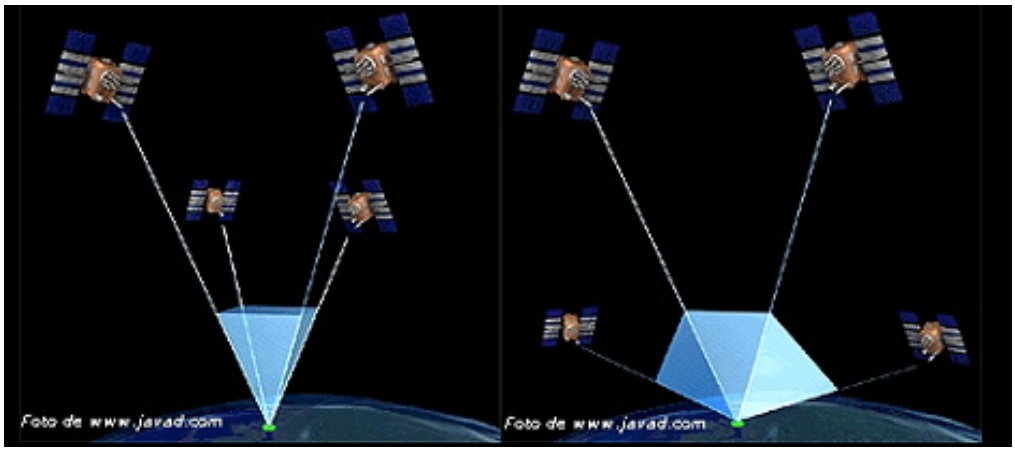


Figura 16 - Volume do espaço criado pelos satélites

Fonte: < http://www.malima.com.br/satelite/blog_commento.asp?blog_id=9 >

2.6.1. Fontes de Erros

Existem vários fatores que influenciam a erros de posição nos receptores de GPS, alguns são citados e descritos abaixo: os erros são agrupados por fatores causadores do erro; satélites, receptores e o meio de propagação.

2.6.1.1. Erros por Satélites

Erros nos relógios dos satélites: Os relógios dos satélites são muito precisos (cada satélite contém quatro relógios atômicos, dois de rubídio e dois de césio), mesmo assim esses não são perfeitos. Para se ter idéia da magnitude do erro tendo

um nano segundo de erro, ou seja, 0,000000001 s, o erro de posição na Terra chega a ser cerca de 30cm na medição da pseudo-distância.

Erros nas efemérides: A precisão da posição em Terra está relacionada com a precisão da localização dos satélites (pontos de referência). Apesar de os satélites serem colocados em órbitas precisas devido ao insuficiente conhecimento do campo gravitacional, forças gravitacionais da Lua e do Sol e até ao atrito remanescente da atmosfera terrestre bem como a pressão das radiações solares nos satélites provoca variações nas suas órbitas, essas posições são constantemente monitoradas pelos centros de controle.

2.6.1.2. Erros pelos receptores

Erros nos relógios dos receptores: similar aos provocados pelos relógios dos satélites.

Multi-Trajeto: Provocado devido a reflexão em objetos anteriormente a chegada do sinal ao receptor. Um exemplo é a reflexão do sinal em prédios (Figura 17).

Figura 17 - Erro de Multi-Trajeto

Fonte:<http://www.geodesia.org/>

Erros causados pela variação do centro de fase da antena: Causado em função da construção do aparelho. Devido a características da antena e o ângulo da direção.

Ruído do Receptor: Limitações do receptor, limitação baseada no erro causado pelo desvio patrão de cada medição.

2.6.1.3. Erros pelo meio de transmissão

Atrasos ionosféricos: Todo o calculo é feito com base no atraso do sinal com relação à velocidade da luz. Entretanto, essa velocidade varia sob condições atmosféricas distintas. A camada mais alta da atmosfera, a ionosfera, contém partículas "carregadas" que atrasam o código e adiantam a fase.

Atraso troposférico: na camada mais baixa, a troposfera, o sinal também sofre um atraso na fase e no código. O atraso é causado por duas componentes (seca e úmida). O problema está diretamente relacionado com a umidade presente no ar mo momento da leitura.

Essas camadas podem ser observadas na Figura18.

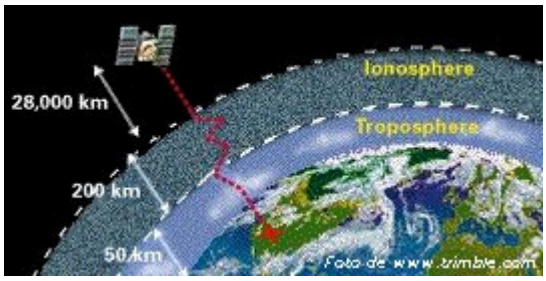


Figura 18 - Erro devido ao meio de transmissão

Fonte:<http://www.geodesia.org/>

3. COORDENADAS GEOGRÁFICAS

Este capítulo apresenta como a Terra é definida para melhor localização, é apresentado o sistema de coordenadas geográficas e também os cálculos com latitudes e longitudes apresentados no projeto, cálculos estes como distância entre dois pontos, ângulo de deslocamento, velocidade de deslocamento e tempo de chegada.

3.1. Localização

A Terra possui um movimento de rotação em torno de um eixo imaginário. Os pontos destes eixos que interceptam a superfície terrestre são chamados de pólos geográficos, norte e sul. Imaginando a Terra um círculo perpendicular ao eixo de rotação e que divide a Terra em duas metades iguais ou hemisférios. O círculo máximo é conhecido como equador terrestre ou equador geográfico. Os demais círculos menores traçados imaginariamente na Terra, paralelos ao equador, são denominados paralelos de latitude terrestre ou geográfica. É possível também traçar outros círculos máximos (que dividem também a Terra em hemisférios), perpendiculares ao equador terrestre. Estes círculos são chamados de meridianos terrestres ou geográficos. Através destes círculos, pode-se determinar as coordenadas geográficas de um lugar.

Um sistema utilizando pontos de referência para determinar qualquer ponto na Terra resultou do desenvolvimento de técnicas cartográficas para a elaboração de mapas. A partir deste sistema é possivel determinar a posição absoluta de qualquer lugar da Terra.

O Sistema baseia-se em um sistema de coordenadas, latitudes e longitudes, considerando a Terra como uma esfera perfeita. Assim, os pólos do sistema foram determinados como os pontos de interseção do eixo de rotação da Terra com a sua superfície e a linha do equador como o raio máximo do planeta.

Para que qualquer ponto da superfície terrestre que possa ser localizado, existe um sistema de coordenadas (linhas imaginárias), que são representadas em um mapa ou carta.

Os meridianos são as linhas imaginárias que são traçadas através dos pólos e ao redor da Terra, ou seja, todos são círculos máximos da esfera cujos planos contêm o eixo de rotação ou eixo dos pólos. Decidiu-se que o ponto de partida para a numeração dos meridianos seria o meridiano que passa pelo observatório de Greenwich, na Inglaterra. Portanto, o meridiano de Greenwich é o meridiano principal Figura 19. A leste de Greenwich os meridianos são medidos por valores crescentes até 180º e, a oeste, suas medidas são decrescentes até o limite de – 180º.

Figura 19 – Meridianos

Fonte: (ROSA, 2004)

Já os paralelos são círculos da esfera perpendiculares ao eixo dos pólos. O Equador é o paralelo que divide a Terra em dois hemisférios. O 0º corresponde ao equador, o 90º ao pólo norte e o -90º ao pólo sul Figura 20.

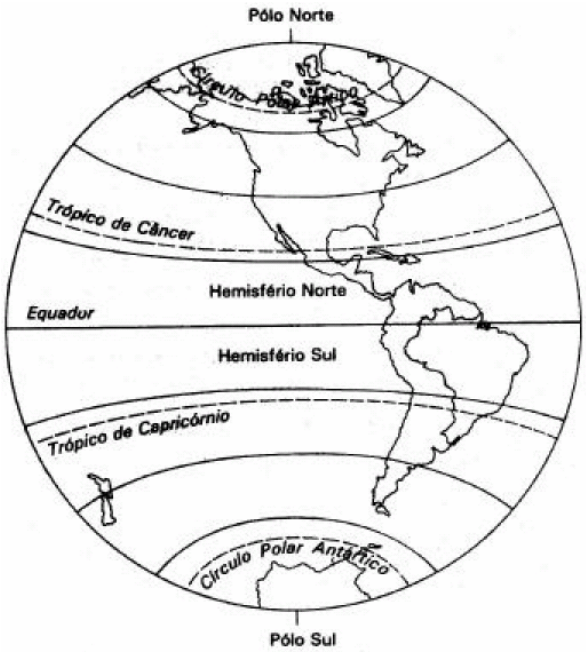


Figura 20 – Paralelos

Fonte: (ROSA, 2004)

3.2. Sistema de Coordenadas Geográficas

É sistema mais antigo de coordenadas. Cada ponto da superfície terrestre é localizado pela interseção de um meridiano com um paralelo. Suas coordenadas são a latitude e a longitude Figura 21 e Figura 22..

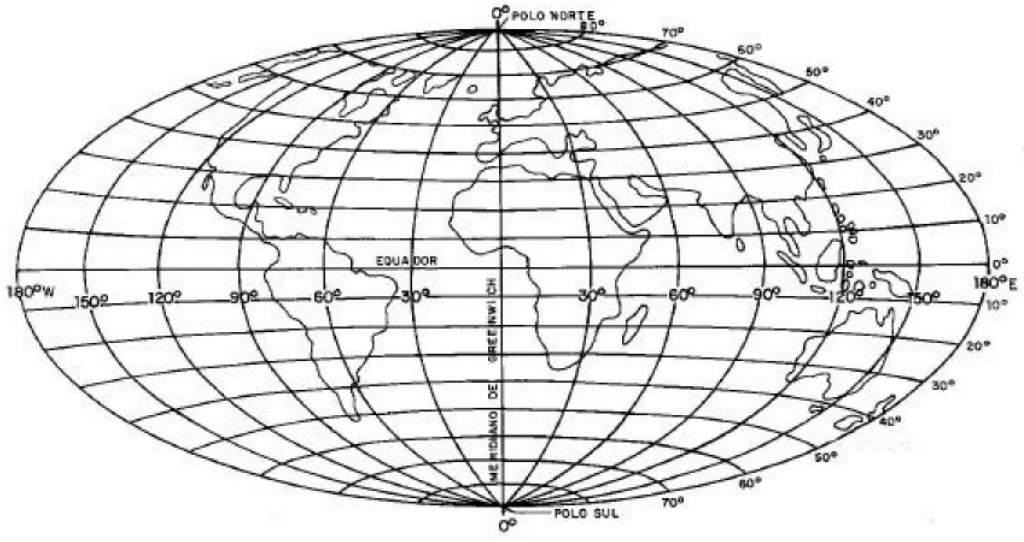


Figura 21 - Latitude

Fonte: (ROSA, 2004)

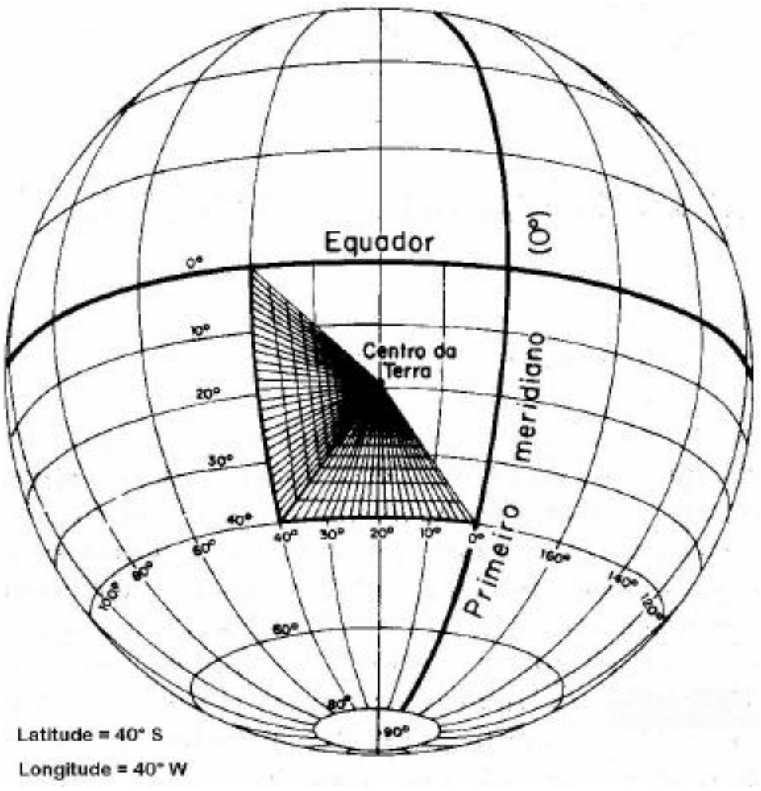


Figura 22 – Longitude

Fonte: (ROSA, 2004)

Longitude geográfica é o ângulo formado entre o meridiano que passa pelo lugar e o meridiano que passa pela cidade de Greenwich, Inglaterra. A longitude é medida pela variação de 0º a 180º, para leste ou para oeste de Greenwich. Por convenção, atribui-se sinais para as longitudes: negativo para oeste e positivo para leste.

Latitude geográfica é o ângulo ponto considerado e o marco 0º que é a linha do equador. Todos os pontos do equador possuem latitude geográfica igual à 0º. Pontos situados ao norte do equador têm latitudes maiores que 0º variando até 90º que é a latitude do pólo geográfico norte. Da mesma forma varia as latitudes ao sul do equador terrestre, desde 0º a 90º, latitude do pólo geográfico sul. Para se diferenciar os valores, atribui-se também similar a longitude convenção de sinal, positivo para as latitudes norte e negativo para as latitudes sul.

Assim, coordenadas geográficas são valores numéricos através dos quais é possível definir a posição de um ponto na superfície da Terra, tendo como ponto de origem para as latitudes o Equador e o meridiano de Greenwich para a origem das longitudes. [www.ibge.com.br]

3.3. Distância entre dois pontos de coordenadas dadas

Para determinar a distância aproximada entre dois pontos em coordenadas geográficas é necessário apenas conhecer a latitude e longitude destes pontos.

O cálculo efetuado corresponde à distância em "linha reta" sob a superfície da Terra entre as duas localidades informadas, sem considerar os aspectos geodésicos[6] nem o relevo da superfície, ou seja, a Terra é tratada como uma superfície esférica ideal. Foi utilizado como referência para os cálculos de Trigonometria Esférica, o raio médio da Terra, e não o raio no Equador.

Como dito anteriormente para tratar a Terra como uma esfera, mostrado na Figura 23, com o Pólo Norte em N e a linha do Equador determinando a metade da esfera.

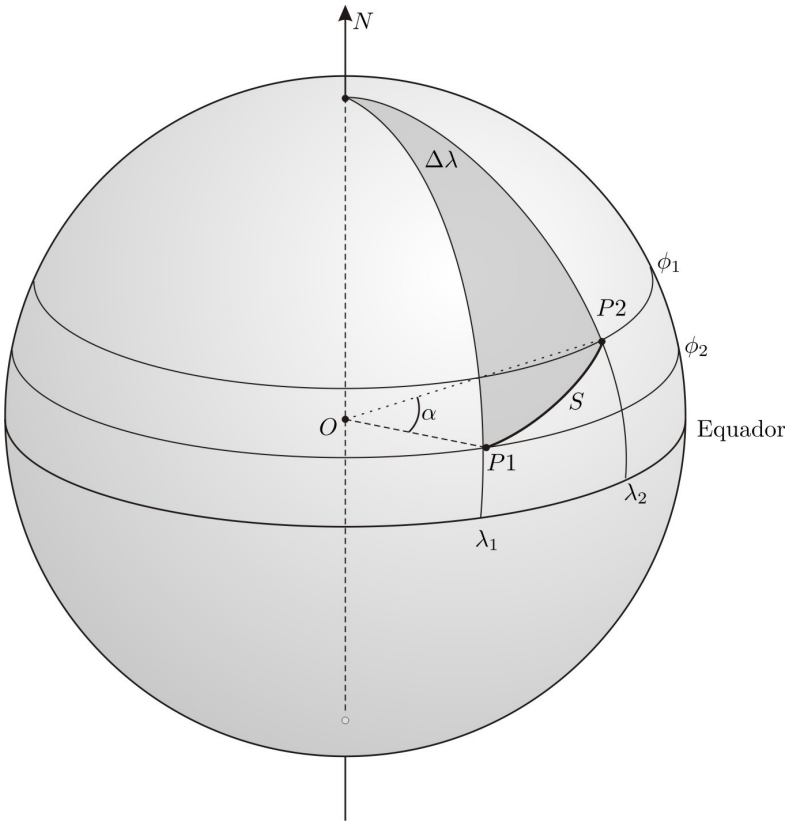


Figura 23 - Determinação da distância entre dois pontos

Fonte: <http://obsn3.on.br/~jlkm/geopath> Modificado pelo autor

[6] Geodésica é uma linha que une dois pontos com o menor tamanho possível, no plano esférico a geodésica é um arco do círculo composto pelos dois pontos e o centro da esfera.

Este cálculo supõe um triângulo esférico cujos 3 vértices são: N - Pólo Norte, P1 - Local 1 e P2 - Local 2.

A distância entre o Local 1 e Local 2, é o comprimento do arco S com centro O no centro da Terra e que passa pelos pontos P1 e P2. Observando novamente na Figura 23. Note que para calcular o comprimento S nosso idéia é concentrar no arco de circunferência com centro em O e que passa pelos pontos P1 e P2. A situação plana é como segue.

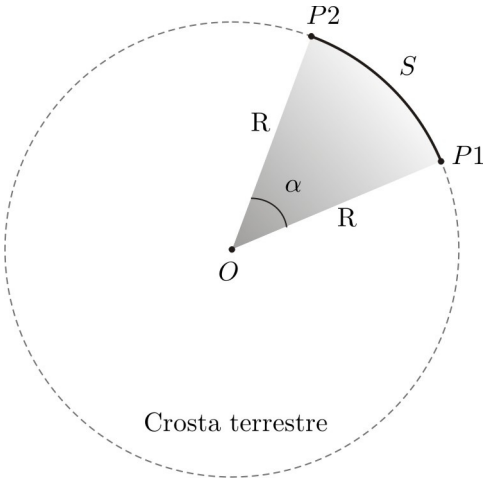


Figura 24 - Raio da Terra

Fonte: (AUTOR, 2007)

Na Figura 24, R é o raio da Terra e S é o comprimento que se quer encontrar. Sabe-se que para um arco de 2π radianos o comprimento é $2\pi R$. Pergunta-se então qual o comprimento do arco S determinado por um ângulo de α radianos. Naturalmente que a proporção dizer que

$$\frac{2\pi R}{2\pi} = \frac{S}{\alpha}$$

e deste modo, $S = \alpha.R$. O comprimento do arco S é então determinado pelo produto do raio da Terra e o ângulo que este arco determina com o centro da Terra.

3.3.1. Encontrando o raio da Terra

Um dos cálculos que mais fascinaram os matemáticos no passado era calcular o raio da Terra. O processo já era imaginado pelos gregos da Antiguidade, porém o que limitava eram os instrumentos de medidas precários.

Eratóstenes (284 – 192 a.C.), que era natural da cidade de Cirene visitou vários outros centros importantes, inclusive Atenas, na Grécia. Embora não tenha sido o primeiro em qualquer ramo do saber, era admirado e respeitado por vários de seus contemporâneos, dentre os quais Arquimedes (287 – 212 a.C.) Já em seu tempo ele estimou o raio da Terra em algo entre 6.369 km e 7.635 km. Hoje se sabe que o primeiro número é mais próximo do real. Além de Eratóstenes, há que se lembrar também de Posidônio (135 – 51 a.C.) que fez o cálculo da Terra com base na distância da cidade de Alexandria até a ilha de Rodes. Estas e outras informações podem ser encontradas em (Ávila, 2007, Capítulo 3).

Como se vê, calcular o raio da Terra não é algo novo. Uma forma de calcular a raio da Terra nos dias de hoje é com a ajuda de um teodolito de precisão (ver Figura 25).

Figura 25 - Teodolito: aparelho usado para medir ângulos

Para tal, é necessário que se tenha o horizonte de forma nítida e a melhor forma é estar em uma montanha perto do mar onde se saiba a altura da montanha. A própria altura da montanha pode ser calculada com o uso do teodolito usando trigonometria plana básica.

A idéia é criar uma situação favorável ao cálculo, situar-se no alto de um morro próximo ao mar, ver com nitidez a linha do horizonte. Suponha que esse morro tem uma altura conhecida h. Com o teodolito na posição 0^o estará em P olhando para Z. Descendo o teodolito até ele estar apontando para o horizonte onde o mar desaparece terá um ponto imaginário H. O ângulo que deverá considerar é o complemento do ângulo $Z\hat{P}H$. Considerando o ângulo μ, complemento do ângulo obtido na leitura, se tem uma situação como pode ser visto na Figura 26.

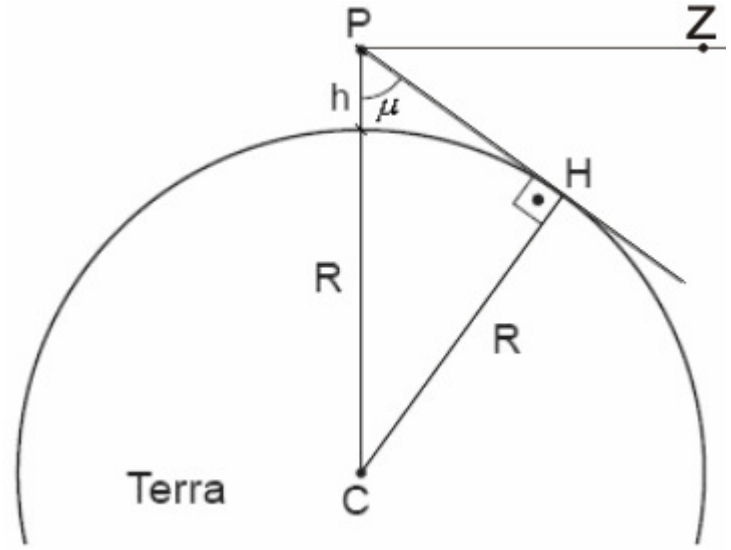


Figura 26 - Achando o raio da Terra

Fonte: (AUTOR, 2007)

Na Figura 26, considerando o ponto C, o centro da Terra e o ponto P é a pessoa que está situada a uma altura h em relação ao nível do mar. Para essa pessoa, o ponto H está na linha do horizonte e, como a reta PH é tangente à Terra, o ângulo PHC é reto. A altura h do morro é conhecida e o ângulo $\mu = CPH$ pode ser medido. Portanto, no triângulo CPH, o seno do ângulo μ é igual a $\frac{CH}{CP}$, ou seja, $\sin\mu = \frac{R}{h+R}$, em que R, o raio da Terra, é a nossa incógnita.

Então,

$$(h+H)\sin\mu = R$$
$$h\sin\mu + R\sin\mu = R$$
$$h\sin\mu = R - R\sin\mu \quad \text{ou seja,} \quad R = \frac{h\sin\mu}{1-\sin\mu}$$
$$h\sin\mu = R(1-\sin\mu)$$

Equação 4 - Raio da Terra

Note que a medida do raio da Terra está diretamente ligada à precisão da leitora de seu teodolito no cálculo da altura do morro (h) e o ângulo $Z\hat{P}H$. Conhecendo a altura h e o ângulo μ, pode-se calcular o raio da Terra usando essa fórmula, mas, na prática, existem dificuldades. A altura h será sempre muito pequena em relação ao raio da Terra. Para se obter R com precisão, é preciso medir o ângulo μ também com muita precisão, pois um pequeno erro na medida de μ acarretará um erro muito grande na medida de R. Hoje, existem instrumentos eletrônicos que medem ângulos com precisão de 1 milésimo de grau, e as calculadoras científicas fornecem os senos dos ângulos com a necessária exatidão.

Por exemplo, se a pessoa P está a uma altura de 2 km em relação ao nível do mar, o ângulo a será de 88,657 graus. Com uma calculadora científica, encontra-se o seno desse ângulo igual a 0,9996872 e o raio da Terra aproximadamente igual a 6390 km.

3.3.2. Trigonometria

A trigonometria plana é essencial para o entendimento de conceitos e resolução dos problemas referente à geometria plana e funciona muito bem quando se trata de regiões pequenas em relação à Terra. Para entender isto, basta imaginar o cálculo da distância de Brasília até Goiânia e a distância de Brasília até Moscou. De Brasília até Goiânia é possível imaginar que estas cidades estão em um plano, enquanto que a distância de Brasília até Moscou deve ser vista como um arco de circunferência.

A Trigonometria Esférica por outro lado trata de triângulos que são uma seção da superfície de uma esfera e resolução dos problemas de Navegação Astronômica.

3.3.2.1. Trigonometria plana

Algumas relações trigonométricas utilizada no item 3.3.2.2 são apresentadas neste item. Dado um ponto P sobre uma circunferência de raio 1, o co-seno do ângulo descrito pelo arco AP é a abscissa do ponto P e o seno deste mesmo ângulo é a ordenada deste mesmo ângulo é a ordenada do ponto P (ver Figura 27).

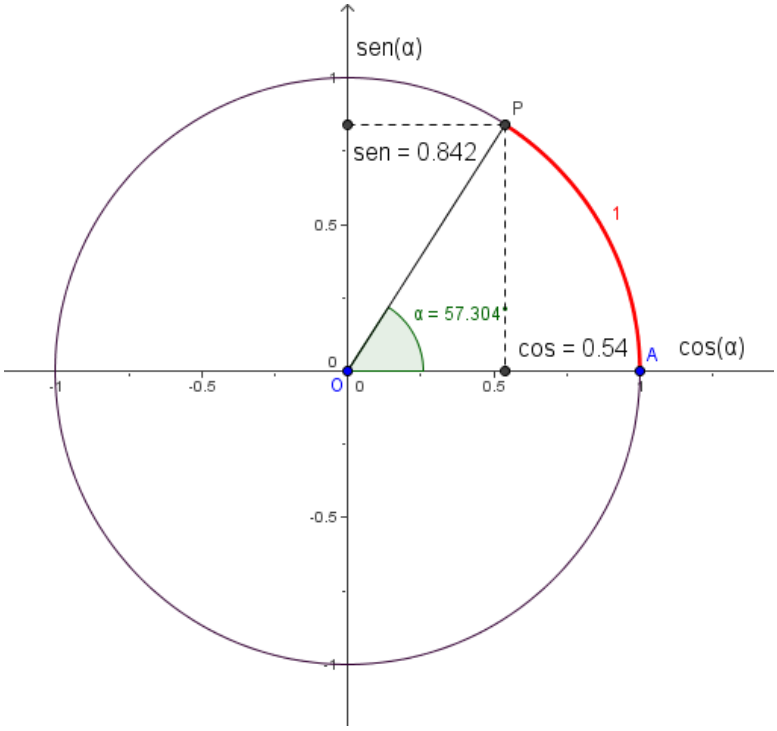


Figura 27 - Relações de trigonometria plana

Fonte: (AUTOR, 2007)

Isto pode ser resumido na seguinte propriedade;

$$\begin{cases} \sin(a) = x(P) \\ \cos(a) = y(P) \end{cases}$$

Isto é conhecido, em geral, por estudantes. Já a secante é menos comum. Lembrando que $\sec(\alpha) = \dfrac{1}{\cos(\alpha)}$, veja Figura 28.

Lema 3.1: Considere uma reta “b” que tangente à circunferência dada e seja “B” o ponto de interseção desta reta com o EixoX. A secante do ângulo α é igual a abscissa do ponto B.

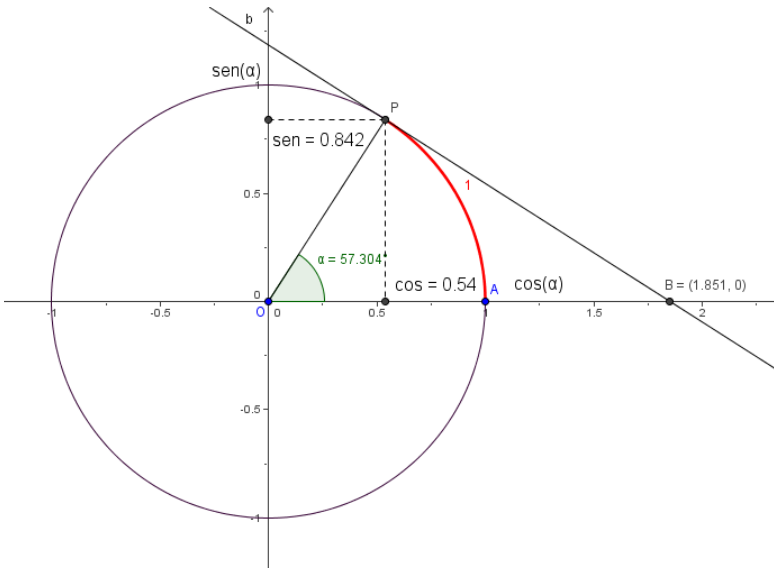


Figura 28 - Demonstração de trigonometria

Fonte: (AUTOR, 2007)

Demonstração: A prova deste resultado é simples, desde que se lembre que o ângulo $O\hat{P}B$ é reto e assim, tomando como referência o triângulo OPB, se tem que

$$\cos(\alpha) = \frac{med(OP)}{med(OB)}$$

e assim

$$med(OB) = \frac{med(OP)}{\cos(\alpha)} = \frac{1}{\cos(\alpha)} = \sec(\alpha)$$

■

Lema 3.3.2.1: Vale a seguinte identidade trigonométrica: $\tan^2(a) - \sec^2(a) = -1$.

De fato, como $sen^2(a) + \cos^2(a) = 1$, dividindo ambos os membros por $\cos^2(a)$ se tem

$$\frac{sen^2(a)}{\cos^2(a)} + \frac{\cos^2(a)}{\cos^2(a)} = \frac{1}{\cos^2(a)}$$

e daí,

$$\tan^2(a) + 1 = \sec^2(a)$$

e desta forma $\tan^2(a) - \sec^2(a) = -1$.

■

3.3.2.2. Trigonometria esférica

Triângulo esférico é a porção da superfície esférica compreendida entre três arcos, Figura 29. Os ângulos do triângulo esférico ABC são simbolizados com as letras A, B, C e os lados opostos, com as minúsculas respectivas: a, b, c e a medida de cada lado é igual à medida do respectivo ângulo central:

lado a = ângulo central BOC
lado b = ângulo central AOC
lado c = ângulo central AOB

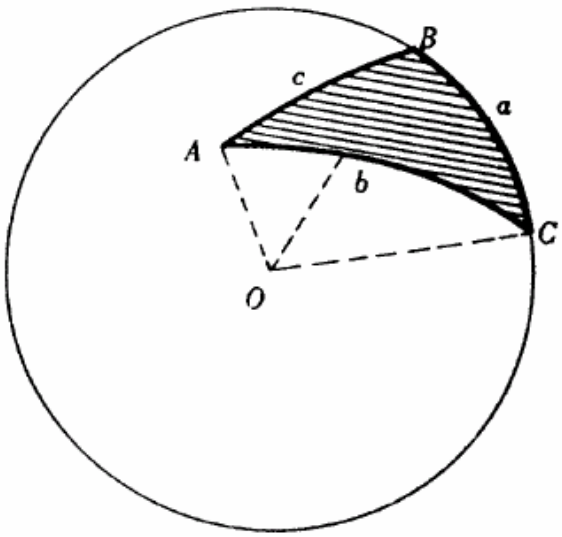


Figura 29 - Triângulo Esférico A B C

A título de curiosidade eis algumas propriedades para triângulos esféricos:

1 - A soma dos 3 lados de um triângulo esférico é maior que 0º e menor que 360º. $0º < a + b + c < 360º$

2 - A soma dos 3 ângulos de um triângulo esférico é maior que 2 retos e menor que 6 retos. $180º < A + B + C < 540º$

3 - Cada lado de um triângulo esférico é menor que a soma e maior que a diferença dos outros dois.

$$|b-c| < a < b+c$$
$$|c-a| < b < c+a$$
$$|a-b| < c < a+b$$

4 - Se 2 lados de um triângulo esférico são iguais, os ângulos opostos também são iguais. A recíproca é verdadeira. Se a = b, então A = B (e reciprocamente)

5 - Ao maior lado se opõe o maior ângulo e vice-versa.

6 - A soma de dois ângulos é menor que o terceiro acrescido de 180º e a diferença é menor que o suplemento do terceiro.

A + B < C + 180º

A – B < 180º– C

Combinando os lados e os ângulos da Figura 30, obtêm as seguintes relações:

$$\begin{aligned} \tan b &= AL \qquad \sec b = OL \\ \tan c &= AK \qquad \sec c = OK \end{aligned} \tag{3.1}$$

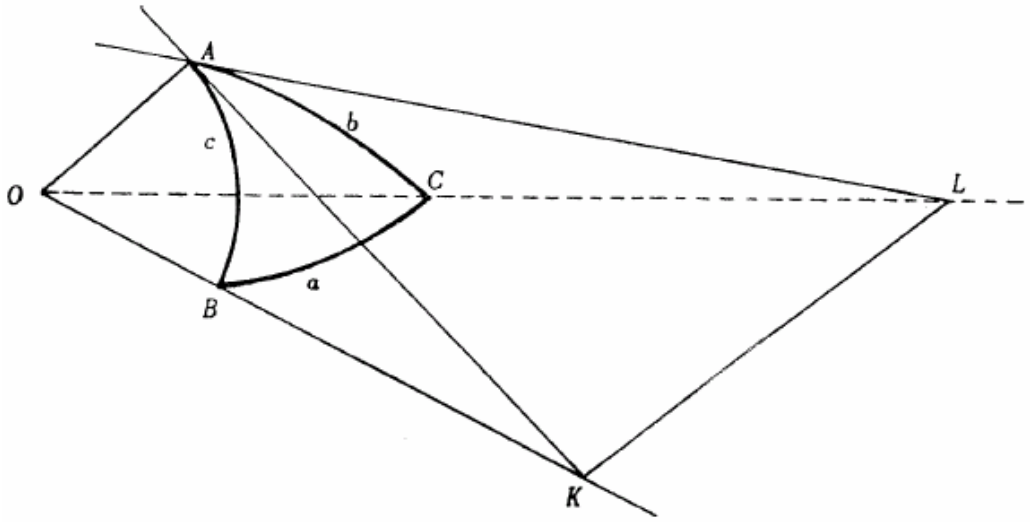


Figura 30 - Representação plana do triângulo esférico

Os triângulos KOL e KAL agora representados de forma plana podem ser escrito de forma:

$$\overline{KL^2} = \overline{OL^2} + \overline{OK^2} - 2.\overline{OL}.\overline{OK}.\cos a$$
$$\overline{KL^2} = \overline{AL^2} + \overline{AK^2} - 2.\overline{AL}.\overline{AK}.\cos A$$

Comparando estas relações tem-se

$$\overline{OL^2} + \overline{OK^2} - 2.\overline{OL}.\overline{OK}.\cos a = \overline{AL^2} + \overline{AK^2} - 2.\overline{AL}.\overline{AK}.\cos A$$

Substituindo usando (3.1) fica-se com

$$\sec^2(b) + \sec^2(c) - 2.\sec(b).\sec(c).\cos(a) = \tan^2(b) + \tan^2(c) - 2\tan(b).\tan(c).\cos(\hat{A}),$$

ou seja,

$$-2\sec(b).\sec(c).\cos(a) = \underbrace{\tan^2(b) - \sec^2(b)}_{-1} + \underbrace{\tan^2(c) - \sec^2(c)}_{-1} - 2\tan(b).\tan(c).\cos(\hat{A})$$

Dividindo os dois membros por (-2), tem-se:

$$\sec(b).\sec(c).\cos(a) = 1 + \tan(b).\tan(c).\cos(\hat{A})$$

Multiplicando por $\cos(b).\cos(c)$, ficará:

$$\underbrace{\frac{1}{\cos(b)}}_{\sec(b)}.\underbrace{\frac{1}{\cos(c)}}_{\sec(c)}.\cos(a).\cos(b).\cos(c) = \cos(b).\cos(c) + \underbrace{\frac{\sin(b)}{\cos(b)}}_{\tan(b)}.\underbrace{\frac{\sin(c)}{\cos(c)}}_{\tan(c)}.\cos(\hat{A}).\cos(b).\cos(c)$$

De onde segue a Equação 5:

$$\cos(a) = \cos(b).\cos(c) + \sin(b).\sin(c).\cos(\hat{A})$$

Equação 5 - Lei dos co-senos para triângulos esféricos

Ver (Função 1, p. 98).

3.4. Ângulo de deslocamento para melhor rota

O GPS por si só não possui uma bússola agregada, o apontamento de direção se dá por meio de um deslocamento mínimo existindo variações de latitude e longitude recebida pelos sinais dos satélites, assim é possível determinar a direção em que o aparelho receptor do sinal está seguindo.

Com base nessa idéia que foi desenvolvido o processo para determinar o ângulo de deslocamento de rota. Considere a seguinte situação. Uma pessoa está em um ponto A e quer ir a um ponto C (Figura 31). O sistema criado é capaz de dizer a

que distância a pessoa está do destino, mas precisa de um outro ponto de referência para poder informar a direção que deverá tomar. Imagine que esta pessoa faça um movimento pequeno em qualquer direção. Este pequeno movimento irá informar ao sistema um novo ponto B e com isto cria uma situação como a descrita na Figura 31.

Após certa variação de latitude e longitude, considerando um deslocamento razoável do receptor tem-se agora 3 pontos conhecidos: A - ponto de origem do deslocamento; B - ponto de destino do deslocamento e C - ponto de interesse onde se deseja chegar. Lembrado que cada ponto é um par de coordenadas, A(LatA, LonA), B(LatB, LonB), C(LatC, LonC).

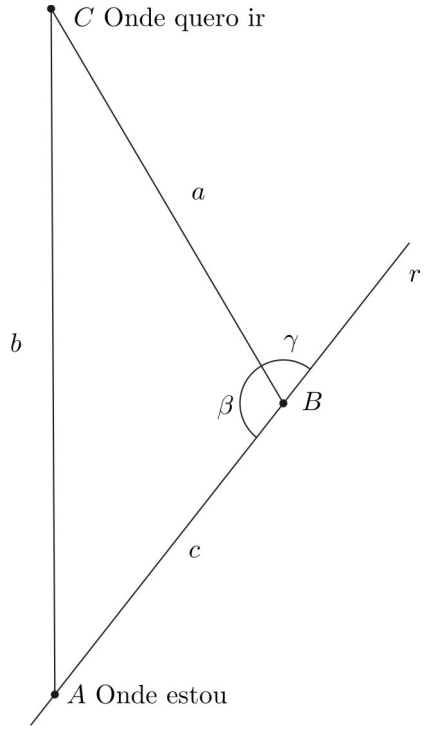


Figura 31 - Determinação do ângulo de deslocamento

O primeiro passo a fazer é determinar as distâncias entre os pontos, calcular a, b e c com base no item anterior. Como se trata do deslocamento de uma pessoa andando, não se faz necessário ver o triângulo criado como sendo um triângulo esférico. Para estes cálculos usa-se a geometria plana. A lei dos cossenos para triângulos planos diz que

$$b^2 = a^2 + c^2 - 2ac\cos(\beta)$$

de onde pode-se encontrar β usando a expressão

$$Cos(\beta) = \frac{a^2 + c^2 - b^2}{2ac}$$

Equação 6 - Lei dos Co-senos

de onde se determina

$$\beta = \arccos\left(\frac{a^2 + c^2 - b^2}{2ac}\right)$$

Entretanto, para um pedestre anda de A até B, o ângulo que deve ser informado é o γ mostrado na Figura 31 e com isso, β subtraído de 180º se tem λ que é o ângulo desejado. Ver (Função 2, p. 100).

Uma pergunta que pode surgir neste momento. Achar o ângulo de deslocamento foi demonstrado acima, mas se o calculo está relacionado apenas com as distâncias entre os pontos e não com os pontos de coordenadas, o usuário deverá girar λ graus no sentido horário ou no sentido anti-horário? A situação colocada na Figura 32 ilustra duas situações onde os pontos estão em posições diferentes porem a distância e o ângulo entre os pontos são iguais, apenas muda o sentido de deslocamento.

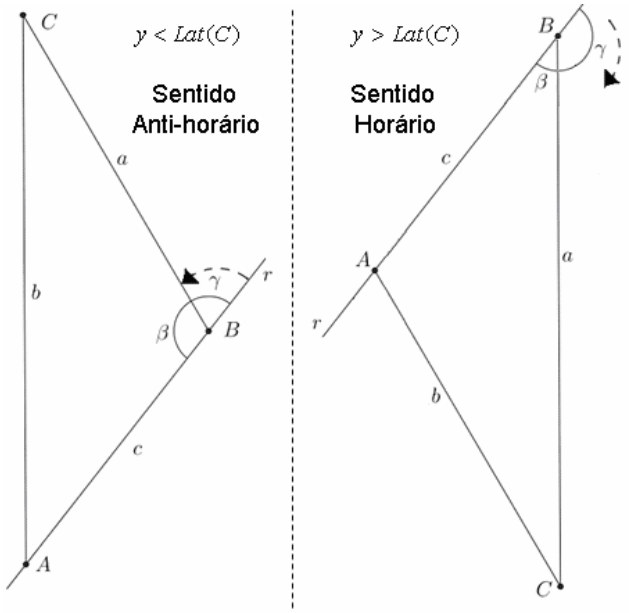


Figura 32 - Sentido de deslocamento

Para resolver este problema usa-se a equação da reta que passa pelos pontos A e B e determinar se o ponto C está acima desta reta ou abaixo. Se estiver acima o usuário deverá girar no sentido anti-horário e caso contrário no sentido horário.

Traça-se uma reta entre os pontos A e B, ver Figura 33. É sabido que a equação de uma reta que passa pelos pontos $A(x_0, y_0)$ e $B(x_1, y_1)$ é dada por

$$y - y_0 = m(x - x_0) \text{ onde } m = \frac{y_1 - y_0}{x_1 - x_0}$$

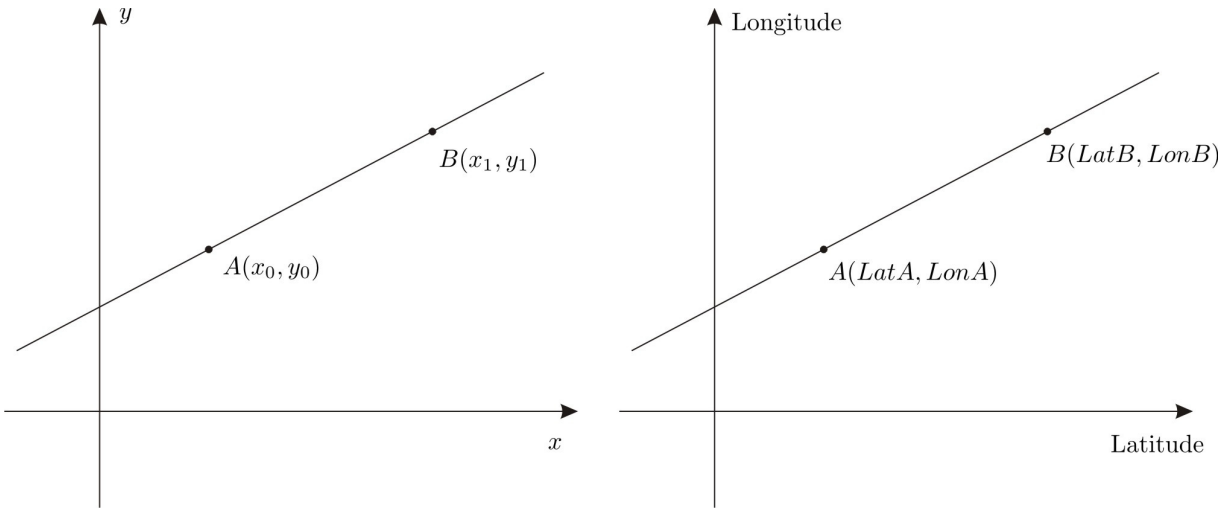


Figura 33 - Determinação de reta

Levando esta idéia para o sistema que envolve latitude e longitude ter-se-á

$$m = \frac{LatB - LatA}{LonB - LonA} \text{ e } y - LatA = +\frac{LatB - LatA}{LonB - LonA}(x - LonA)$$

A verificação se o ponto C está acima ou abaixo da reta se dá pela substituição de x pela Longitude do ponto C (LonC), o resultado y é verificado com a latitude do Ponto C (LatC), se $y > Lat(C)$ o sentido é anti-horário, se $y < Lat(C)$ o sentido é horário e se $y = 0$ o ponto de destino se encontra na reta de deslocamento, então verifica se ele se afasta ou se aproxima do ponto, o ângulo de deslocamento será 0º ou 180º, a Figura 34 ilustra bem a idéia, com isso está definido o sentido horário ou anti-horário em relação ao deslocamento de A para B para chegar em C e até se o sentido no deslocamento permanece ou se o sentido muda 180º. Ver (Função 3, p. 101).

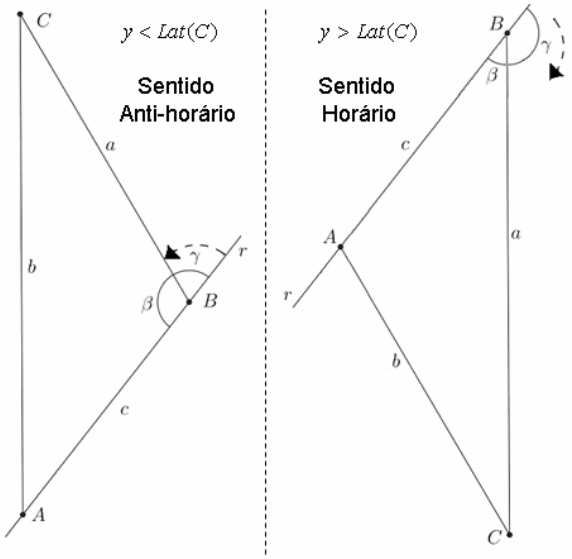


Figura 34 - Calculo do sentido de deslocamento

3.5. Velocidade de deslocamento

Em Física, velocidade (v) é a medida da rapidez com a qual um corpo altera sua posição. A velocidade média, que é uma medida da velocidade, é a razão entre um deslocamento e o intervalo de tempo levado para efetuar esse deslocamento. É matematicamente expressa por:

$$v = \frac{\Delta S}{\Delta t}$$

Equação 7 - Velocidade média

A velocidade instantânea é, portanto, definida como o limite da razão entre o espaço percorrido e o tempo gasto, quando o tempo tende a zero., ou seja

$$v_{inst} = \lim_{\Delta t \to 0} \frac{\Delta S}{\Delta t} = \frac{ds}{dt}$$

Equação 8 - Velocidade instantânea

Quando se considera um intervalo de tempo que não tende a 0, a velocidade é considerada média. A velocidade instantânea pode ser entendida como a veloci-

dade de um corpo no exato instante escolhido. No movimento retilíneo uniforme, a velocidade instantânea coincide com a média em todos os instantes.

O calculo de velocidade é bastante simples e é necessário apenas fixar um tempo para calcular a distância em que o receptor deslocou, ver Figura 35, dividir o deslocamento entre os dois pontos de medição encontrados pelo tempo pré-especificado, com isso tem-se a velocidade média do deslocamento do receptor de GPS.

O deslocamento é obtido pela demonstração do item 3.3.

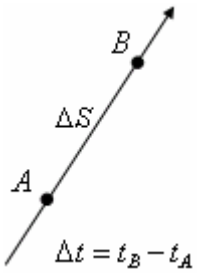


Figura 35 - Velocidade de deslocamento

A velocidade pode ser expressa em inúmeras unidades de medida, que relacionem o espaço (S) e o tempo (t) na forma $v = \frac{S}{t}$. O SI[7] tem como unidade padrão para velocidade o metro por segundo (m/s) que é a aplicada no projeto, mas unidades comumente usadas incluem o quilômetro por hora (km/h), centímetro por segundo (cm/s), milha por hora, etc. Ver (Função 4, p. 101)

3.6. Tempo de chegada

A previsão do tempo de chegada é uma forma inversa da velocidade, ilustrado na Figura 36, aplicando a mesma fórmula, Equação 7 - Velocidade média, é possível calcular o tempo previsto de chegada ao ponto de interesse.

[7] Sistema Internacional de Unidades (SI) – Conjunto de definições utilizadas e padronizadas em quase todo o mundo com o objetivo de uniformizar e facilitar as medições.

$$t = \frac{\Delta S}{v}$$

Equação 9 - Tempo de chegada

A velocidade utilizada é a calculada do item 3.5, o deslocamento se dá pelo calculo da distância do ponto B ao C visto anteriormente no item 3.3.

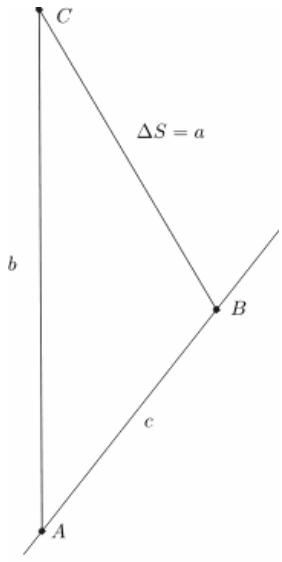


Figura 36 - Tempo de Chegada

A unidade de tempo apresentada no projeto é o segundo. Ver (Função 5, p. 101)

4. CONSTRUÇÃO DO PROTÓTIPO

Neste capítulo são abordados os procedimentos necessários à criação e montagem tanto de hardware quando de software do protótipo. É detalhada a construção do protótipo, descrevendo as etapas e justificativas de montagem. São também caracterizados os principais componentes que compõem o protótipo e o papel de cada um no sistema como um todo e por fim especificadas as tecnologias utilizadas e os motivos de escolha de cada uma.

4.1. Hardware

4.1.1. Diagrama de Blocos

O protótipo construído está ilustrado através do diagrama de blocos representado na Figura 37, mostrando os principais componentes e o fluxo de operação.

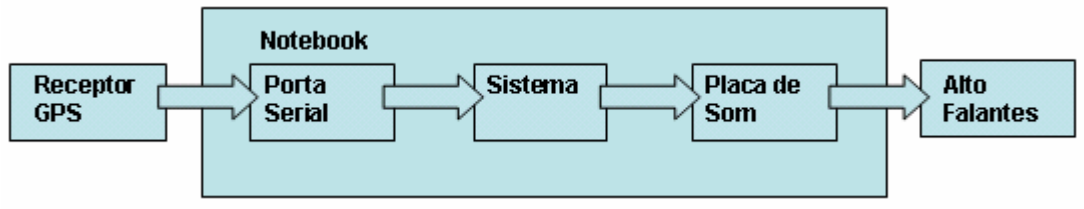


Figura 37 - Diagrama de blocos e fluxo de dados

4.1.2. O Receptor GPS

O receptor utilizado é um GPS do modelo eTrex Legend do fabricante Garmin, ver Figura 38. Mais utilizado para navegação em aventuras nas matas e florestas, possuindo mapas e rotas pré-instaladas.

Figura 38 - Garmin eTrex Legend

A maioria de suas funcionalidades não será utilizada no protótipo, as únicas informações e as principais são a de latitude e longitude do ponto atual de onde se encontra o GPS. Um fator importante à escolha do aparelho se dá na precisão que ele pode oferecer.

Não serão apresentadas as funcionalidades específicas a este aparelho e que não possui utilidade no propósito em questão, são apresentadas na Tabela 4 especificações técnicas relevantes ao poder de atualização e precisão do aparelho.

Formato da Posição	Latitude/Longitude
Receptor	Differential-ready, 12 canais paralelos
Tempo de Aquisição	quente/frio 15 seg./45 seg
Auto-Localização	2 minutos
Taxa de Atualização	1 segundo, contínua
Precisão na posição	15 metros, 95% típica
Velocidade	0.05 metros/seg
Precisão DGPS na posição	3 a 5 metros 95% típica

Interface	RS232 with NMEA 0183, RTCM 104 DGPS data format e formato GARMIN
Antena	Interna

Tabela 4 - Especificações Técnicas do Aparelho GPS

Fonte: <https://buy.garmin.com/shop/shop.do?pID=173&locale=en_US>

4.1.3. Porta Serial

A comunicação entre o computador e o GPS é realizada através de uma porta serial. O software é responsável por enviar os comandos de solicitação de coordenadas para o GPS e ler a resposta do mesmo para a interface de comunicação do notebook por esta porta.

4.1.3.1. Comunicação com RS-232

A interface serial mais comumente utilizada nos microcomputadores é a RS-232. Originalmente foi criada para facilitar a interconexão dos terminais e dos equipamentos de comunicação de dados (TAFNER, 1996).

Na interface RS-232 os pinos mais utilizados são três, sendo um com a função de enviar e outro com a função de receber os dados. Uns poucos pinos no conector são absolutamente previsíveis conforme mostrado na Tabela 5 (TAFNER, 1996).

Pino	**Função**
Pino 2	Pino para transmissão
Pino 3	Pino para recepção
Pino 5	Circuito comum

Tabela 5 - Pinos de comunicação serial

No que diz respeito às características elétricas, o padrão RS-232 define atualmente 4 níveis lógicos. As entradas têm definições diferentes dos dados. Para as saídas, o sinal é considerado na condição de estado “1”, quando a tensão no circuito de transferência, medida no ponto de interface é menor que –5V e maior que –15V, com relação ao circuito de referência (terra). O sinal é considerado na condição de estado “0”, quando a tensão for maior que +5V e menor que +15V, também com relação ao circuito de referência (terra) (TAFNER, 1996)..

Para as entradas, o sinal é considerado em condição de marca, ou estado “1”, quando a tensão no circuito de transferência, medida no ponto de interface, é menor que –3V e maior que –15V, com relação ao circuito terra. O sinal é considerado na condição de espaço ou estado “0”, quando a tensão for maior que +3V e menor que +15V, também com relação ao circuito terra. A região compreendida entre –3V e +3v é definida como região de transição (TAFNER, 1996)..

Durante a transmissão dos dados, a condição de marca é usada para discriminar o estado binário “1”, e a condição de espaço é usada para discriminar o estado binário “0” (TAFNER, 1996).

Muitas aplicações utilizam a conexão direta via cabo para trocar informações entre dois computadores. As utilidades vão desde o simples compartilhamento de arquivos sem a utilização de placas de rede até o jogo entre dois adversários em computadores diferentes. Cada computador dispõe de pelo menos uma porta serial, o conector pode ser um DB9 ou um DB25 (TAFNER, 1996).

4.1.3.2. Conversor USB - Serial

O Conversor USB-Serial tem o objetivo de conectar qualquer dispositivo serial na porta USB do micro, tornando os dispositivos Plug&Play permitindo até ganho de velocidades caso a limitação de velocidade venha a ser da porta serial e não do dispositivo ligado nesta.

A necessidade da implementação veio pelo motivo de que o notebook não possui portas serial e a interface do GPS se dá exclusivamente por esse meio.

Com relação à conexão tornou-se transparente esse link, para o protótipo, todas as funcionalidades são trabalhadas como se existisse a conexão com uma porta serial e não com uma USB. O conversor está apresentado na Figura 39.

Figura 39 - Conversor USB - Serial

4.1.4. O Notebook

O poder de processamento teve que ser avaliado minuciosamente, como o objetivo é fornecer dados com a maior precisão possível isso requer um hardware mais robusto. Outro fator importante e que requer recursos de hardware se dá ao fato dos dados de saída ser sonoras, consumindo uma parcela do processador e de memória RAM.

Figura 40 – Notebook

O Notebook, Figura 40, é composto com um hardware contendo as especificações, Tabela 6, necessárias ao funcionamento do protótipo:

Processador	AMD Semprom Mobile 3000+ com Clock de 1.80 GHz
Memória Principal	1 Gb de RAM DDR 333 MHz
Memória Secundária	HD 60 GB
Comunicação Externa	3 Portas USB 2.0
Placa de Som	Realtek

Tabela 6 - Especificação técnicas do notebook

4.2. Software

4.2.1. Sistema Operacional

O sistema operacional utilizado foi o Windows XP Professional com Service Pack 2, versão bastante estável e leve para o hardware que irá processar as informações.

O sistema é capaz de controlar o framework utilizado, Net 2.0; gerenciar o banco de dados utilizado, MySQL; pacote que faz o tratamento do som, Microsoft Windows Speech API; Alfabeto Fonético L&H para o português do Brasil; assim como todos os periféricos.

Não somente nesse ambiente que funcionará o protótipo como também fora desenvolvido o projeto, acrescentando o Visual Studio 2005.

4.2.2. Framework

Para a construção e funcionamento do software localizador foi utilizado o Microsoft Framework 2.0[8], linguagem VB.Net. A escolha deu-se pela familiaridade com

[8] Ver http://msdn2.microsoft.com/pt-br/netframework/default.aspx

o ambiente e pela facilidade que a linguagem traz para implementação de orientação a objeto, além de sua utilização e distribuição serem gratuitas.

4.2.3. MYSQL

O banco utilizado foi o MySql 4.1[9] para Windows NT, por ser um banco de dados gratuito, simples de configurar e administrar, um dos mais performáticos existentes no ambiente computacional e que se comunica com o .Net de forma eficiente.

4.2.4. Microsoft Agent 2.0

Microsoft Agent[10] é a tecnologia da Microsoft que trabalha com o recurso "Text-to-Speech", essa funcionalidade permite que o computador produza sons a partir de textos e também é responsável pelo gerenciamento das vozes instaladas no computador. Vários softwares que trabalham com voz requerem este pacote instalado.

4.2.5. Alfabeto Fonético L&H

Lernout & Hauspie® Portuguese TTS Engine TTS3000 é um plugin da Microsoft onde você poderá em alguns programas usar o recurso de ouvir um texto escrito. Ele pode lê em português do Brasil, com esta tecnologia TTS a velocidade da tradução de escrita para voz é feita de forma rápida e com um som aceitável.

---

[9] Ver http://dev.mysql.com/doc/refman/4.1/pt/index.html

[10] Ver http://www.microsoft.com/msagent/

4.2.6. Visual Studio 2005

O Microsoft Visual Studio[11] é um conjunto de ferramentas integradas para desenvolvimento de software, voltado para diversos públicos desde amadores a equipes corporativas. Construído para o desenvolvimento de aplicações rápidas e conectadas com experiências ricas.

4.2.7. DBDesigner 4.0

Para modelar o Banco de dados foi utilizado uma ferramenta gratuita e Open Source DBDesigner 4.0[12], desenvolvido pela fabForce.net. Utilizando tecnologia avançada, oferece uma ferramenta para modelagem e manutenção de diversos bancos de dados e como não poderia deixar de oferecer suporte, o próprio MySQL.

Dessa ferramenta também foi gerado o modelo de dados que é mostrado no capítulo seguinte.

[11] Ver http://www.msdnbrasil.com.br/visualstudio/
[12] Ver http://fabforce.net/dbdesigner4/index.php/

5. SISTEMA DE LOCALIZAÇÃO

Este capítulo detalha a construção do *software* localizador, que está dividido em quatro camadas: Entidades, Persistência, Negócio e Apresentação. Há ainda um tópico apresentado o banco de dados, tipo de tabela utilizada, model de dados e a ferramenta utilizada para modelagem dos dados e por último relacionando o funcionamento do .Net com o Microsoft Agent 2.0 e Alfabeto Fonético L&H. O código-fonte completo do software está no Apêndice 1.

5.1. Camada de Entidades

Nesta camada foi desenvolvido um conjunto de classes que simplesmente espelham o banco de dados, nela criados os objetos referenciados por todo o projeto, as classes são clsPonto, colPonto, clsCategoria e ctlCategoria.

A classe clsPonto refere-se a uma unidade de dado mapeado do banco de dados o qual já possui por volta de 700 pontos de todo o Distrito Federal com valores referentes a Latitude, Longitude, Descrição e Identificação da categoria que o ponto pertence. A classe colPonto é apenas uma coleção de dados da classe clsPonto.

A classe clsCategoria contem informações de identificação e descrição de Categorias que os pontos estão agrupados para melhor visualização e controle. Similar a colPonto a colCategoria é um classe que contem uma coleção de pontos. Abaixo na Figura 41 está apresentado o modelo de classes da camada de Entidades.

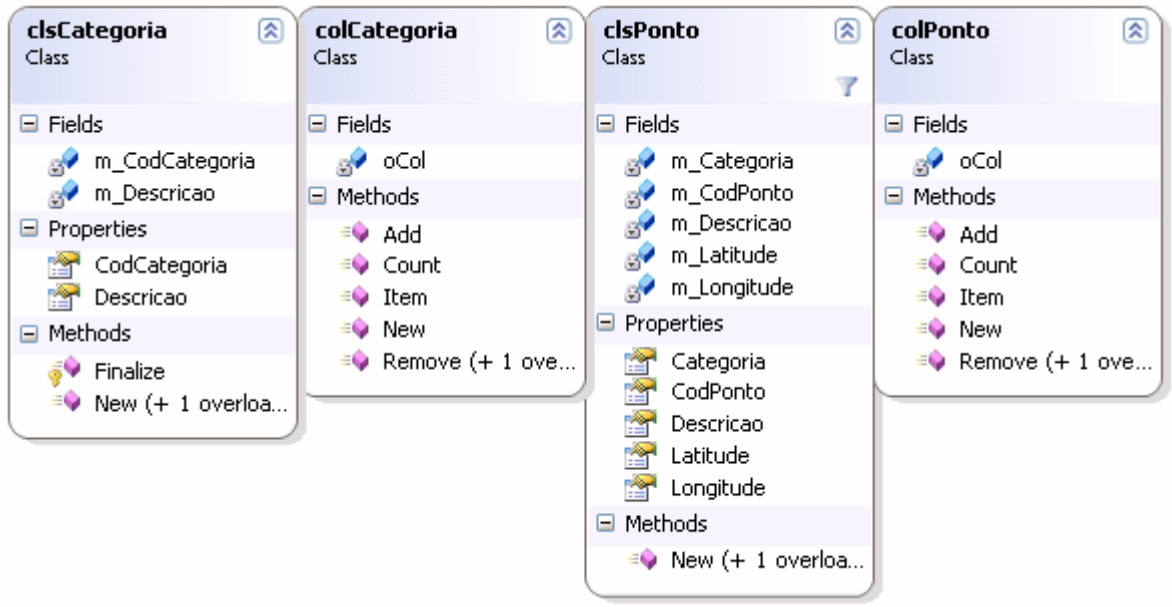


Figura 41 - Modelo de classes, Entidades.

5.2. Camada de Persistência

Nesta camada é que se encontram todos os métodos referentes ao acesso com o Banco de dados, uma classe de acesso ao banco foi criada para gerenciar qualquer transação efetuada com o banco, a classe DBConnection, responsável por criar a conexão com o Banco de dados, realizar consultas, atualizações e exclusões de dados diretamente na base.

Outras duas classes chamadas ctlPonto e ctlCategoria foram desenvolvidas para conter principalmente as instruções SQL, instruções estas interpretadas pelo banco de dados, e criar ligações entre a DBConnection e as classes de negócio. Segue o modelo de classes de persistência e seus atributos na Figura 42.

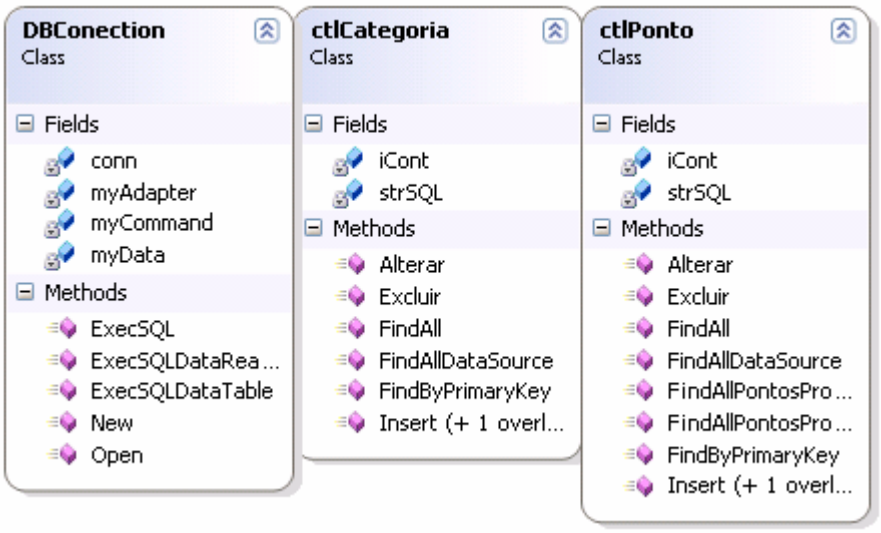


Figura 42 - Modelo de classes, Persistência.

5.3. Camada de Negócio

Nesta camada onde se encontra toda a inteligência do sistema, possui 3 classes, NegPosicao, NegPontoProximo e NegPontosProximos; também se encontram 2 módulos, ModuleGPS e ModPrincipal.

A classe NegPosicao é a classe responsável pelo controle de posicionamento do sistema, nela se encontra a rotina de atualizações de dados referentes a localização do usuário, mantendo todos os pontos mapeados e calculada a distância para cada ponto. Estão presentes também nessa classe os dados referentes ao ponto do instante anterior ao da leitura de latitude e longitude vindos do GPS, essa informação é necessária para os cálculos de Velocidade, Rota e Tempo de chegada em pontos selecionados.

A classe NegPontoProximo refere-se ao controle que se deve ter um calcular a distância do usuário com os diversos pontos cadastrados na base de dados, o objetivo é manter os pontos atualizados e fornecendo informações concisas a classe NegPosicao. A classe NegPontosProximos é uma coleção da NegPontoProximo já que vários pontos são tratados ao mesmo tempo.

O modulo ModuleGPS, contém funções responsáveis pela comunicação entre o notebook e o GPS, esse modulo é responsável por enviar a requisição dos dados de latitude e longitude e receber a resposta do GPS contendo essas informa-

ções. Nesse modulo que se encontra a padronização aplicada para o funcionamento do protocolo utilizado para a comunicação, NMEA0183[13].

O modulo ModPrincipal contem os principais métodos do sistema, métodos de controle, acesso e centralização das informações. Abaixo na Figura 43 segue o modelo de classes da camada de negócio.

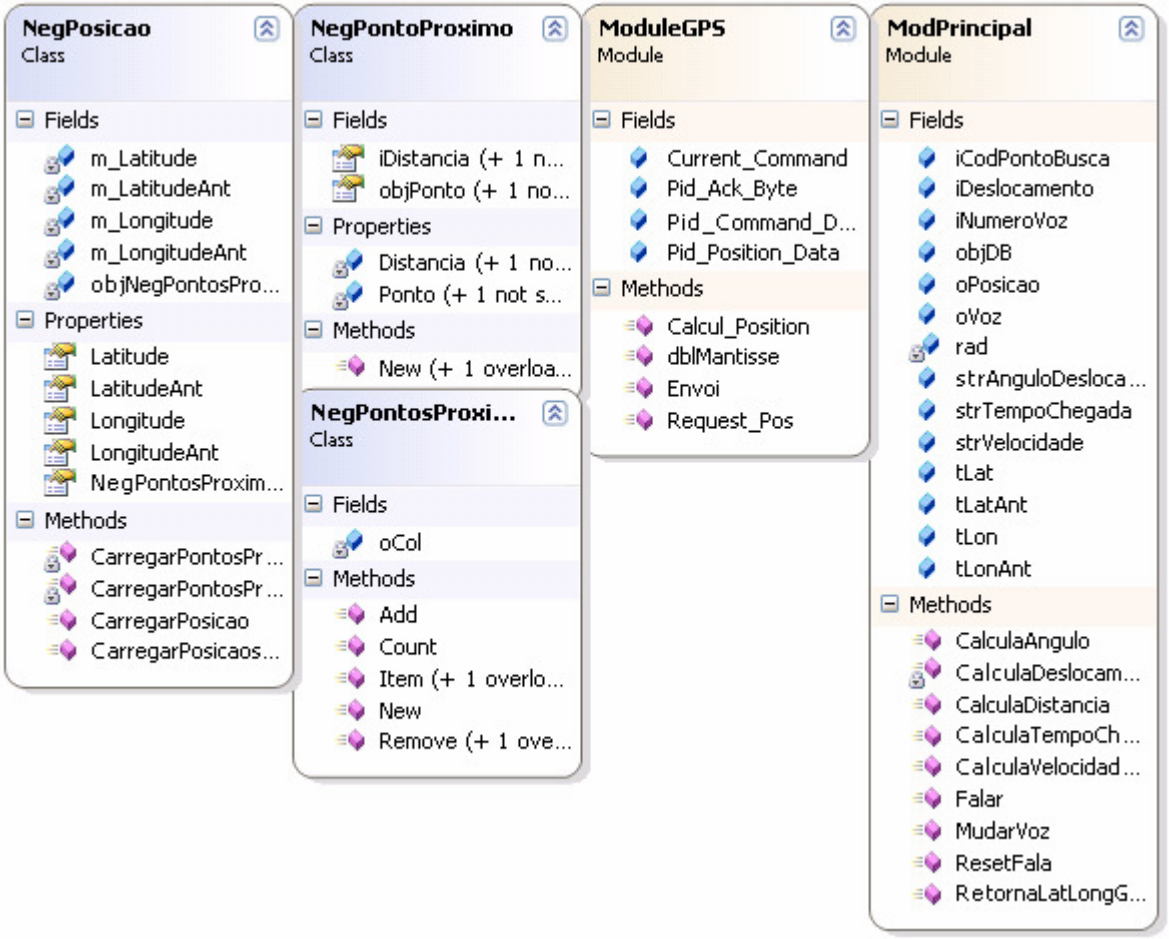


Figura 43 - Modelo de classes, Negócio.

Segue uma breve descrição dos principais métodos do modulo Modprincipal:

CalculaAngulos – Responsável por calcular o ângulo de deslocamento detalhado no item 3.4.

[13] NMEA0183 é um protocolo de comunicação desenvolvido pela Marinha Americana, trata-se de uma combinação de especificações elétricas e de dados para comunicação entre componentes eletrônicos e dispositivos receptores de GPS. (MONICO, 2000)

CalculaDeslocamento - É responsável por calcular o sentido de deslocamento horário ou anti-horário também detalhando no item 3.4 da monografia

CalculaDistancia – Capaz de calcular em metros a distância entre pontos dados as posições em latitudes e longitudes, item 3.3

CalculaTempoChegada – Capaz de calcular o tempo previsto de chegada ao ponto, após calcular a velocidade e a distância, neste método calcula o tempo previsto de chegada, item 3.5

CalculaVelocidade – Retorna a velocidade média de deslocamento do usuário em metros por segundo, item 3.4

Falar – Método responsável por invocar o processo de transcrição de texto para fala.

MudarVoz – Responsável pelo gerenciamento do timbre de voz utilizado no sistema.

ResetFala – Método de controle no processo de transcrição de texto para áudio.

RetornaLatLonGPS – Método que busca no modulo ModuleGPS dados de latitude e longitude do aparelho de GPS.

5.4. Camada de Apresentação

A camada de apresentação é responsável pelo controle dos eventos, apresentação dos dados sonoros ao usuário e por todo o gerenciamento dos pontos.

5.4.1. Navegabilidade

A navegação ao sistema foi atentamente analisada e criada de uma maneira em que o usuário tenha um menor impacto, criada de maneira tal que seja o mais simples possível, apresentada na Figura 44 - Tela Principal do SpeakPosition.

Os eventos foram trabalhados similarmente a sistemas automatizados de secretárias eletrônicas, suas teclas de acesso foram mapeadas de F1 a F8.

Os níveis de acesso formam determinados da seguinte maneira, como mostra a Tabela 7 - Mapeamento de Navegação.

Tecla/ Nível	0 - Inicial	1 - Localização	2 - Roteamento	7 - Timbre
F1	Localização	Repetição de Localização	Falar Ângulo	Maria Falar
F2	Buscar Ponto	Próximo Ponto de Localização	Falar Distância	João Falar
F3	-	Ponto Anterior de Localização	Falar Velocidade	-
F4	-	Definir Rota	Falar Tempo de Chagada	-
F5	-	Voltar ao Menu Principal	Voltar ao Menu Principal	Voltar ao Menu Principal
F6	-	-	-	-
F7	Mudar Voz	-	-	-
F8	Repetir Ajuda	Repetir Ajuda	Repetir Ajuda	Repetir Ajuda

Tabela 7 - Mapeamento de Navegação

Figura 44 - Tela Principal do SpeakPosition

5.4.2. Gerenciamento de Categorias

Foi desenvolvida uma funcionalidade de gerenciamento de categorias, inclusão, alteração e exclusão de categorias. Abaixo se encontra a tela principal de categorias na Figura 45 - Tela Principal de Categorias onde é possível visualizar e realizar ações para determinada categoria.

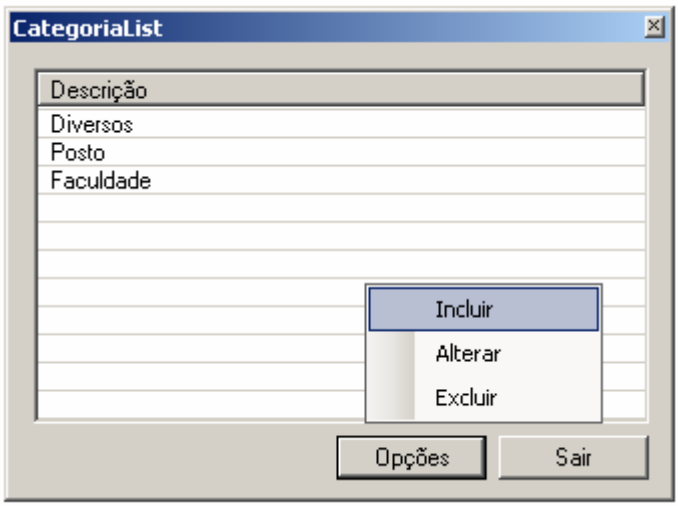


Figura 45 - Tela Principal de Categorias

Após a escolha de uma categoria e a ação a ser tomada o usuário será encaminhado à outra tela, a de detalhamento de categoria, Figura 46.

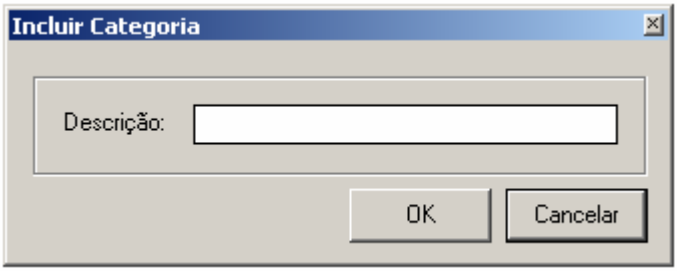


Figura 46 - Tela de detalhamento de categorias

Nesta tela mostrada na Figura 46 é possível cadastrar uma categoria nova, alterar ou excluir uma selecionada na tela principal de categorias.

5.4.3. Gerenciamento de Pontos

Similarmente a tela de gerenciamento de categoria foi desenvolvida uma tela de gerenciamento de pontos, a diferença é que esta possui diversos outros recursos a mais que a de categorias.

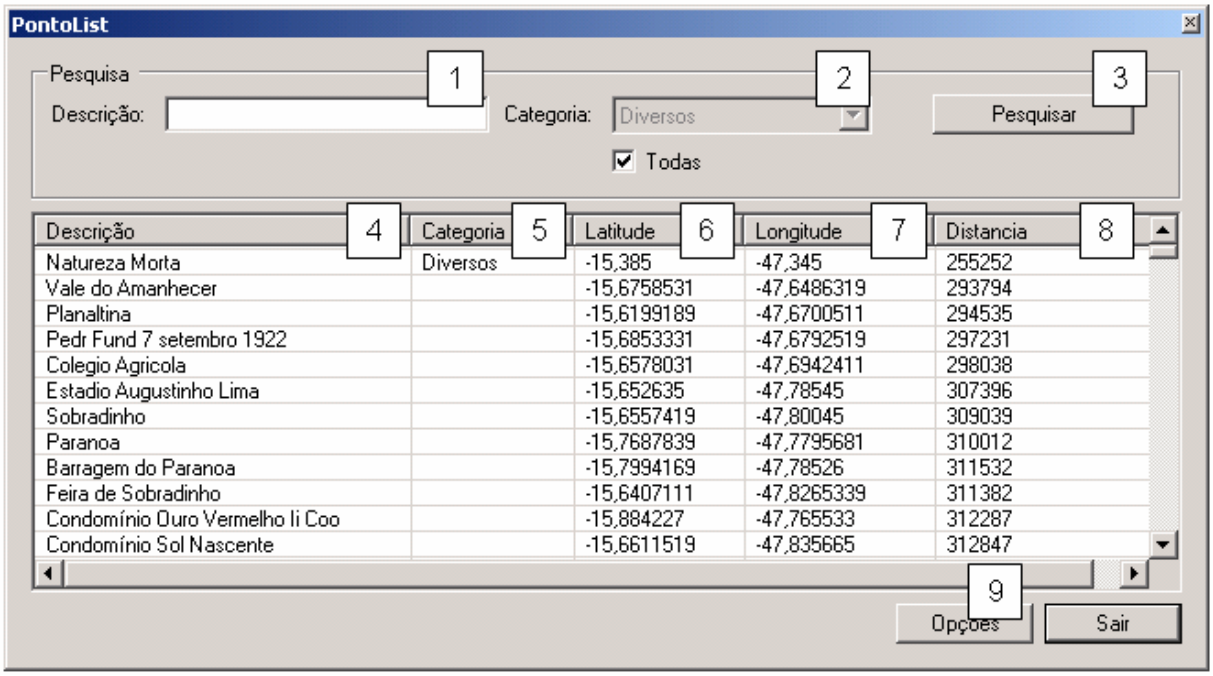


Figura 47 - Gerenciamento de Pontos

As sinalizações 1, 2 e 3 mostrada na Figura 47 refere-se a busca de pontos na base de dados, a 1 é possível encontrar na base apenas aqueles pontos que contenham as palavras digitadas no campo especificado, na 2 é possível restringir os pontos pelas categorias cadastradas, filtrando as categorias conforme o desejado e na 3 o evento para inicio de pesquisa e apresentação no "*list*" apresentado na tela.

Nas sinalizações 4, 5, 6, 7 e 8 da mesma Figura referem-se à apresentação dos pontos, a descrição do ponto - sinalização 4 -, pertencente à determinada categoria - sinalização 5 -, Latitude e Longitude do ponto - sinalizações 6 e 7 -, e finalizando a distância relativa ao usuário no momento em que a busca é efetuada.

Na sinalização 9, são as ações as quais o usuário pode efetuar com relação aos pontos, inclusão, alteração e exclusão de pontos, similar a tela de categorias

essas ações são encaminhadas para outra tela, a tela de detalhamento de pontos, mostrada na Figura 48.

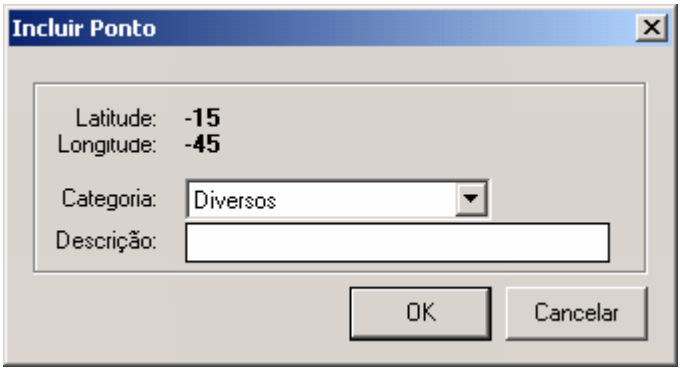


Figura 48 - Detalhamento de pontos

Na tela de detalhamento de pontos é possível cadastrar um novo ponto de interesse, notar que a latitude e longitude são informadas automaticamente, dados vindos do próprio GPS; alterar os dados de um ponto, descrição e categoria e exclusão do ponto.

5.4.4. Pesquisa

Foi desenvolvida também uma funcionalidade para o usuário pesquisar pontos de forma intuitiva com avisos sonora e achar qualquer ponto a sua escolha não mais limitada a pontos próximos, digitando as iniciais o sistema ficará encarregado de fazer a busca e fornecer resultados sonoros para o utilizador. E este poderá escolher o ponto corretamente e iniciar o recurso de roteamento.

5.5. Base de Dados

Para a implementação do sistema como dito anteriormente foi utilizado o banco de dados MySQL 4.1, por ser um banco gratuito para aplicações não comerciais, ter uma robustez e velocidade aceitáveis para o projeto e além de ser facilmente

configurável e utilizado. A tipo de tabela utilizada foi o MyISAM[14] por um excelente desempenho em leituras e escritas e não muito tão bom em concorrências de usuários, que não convém no sistema.

O modelagem dos dados foi efetuado na ferramenta DBDesigner apresentado no item 4.2.7 desta monografia e na Figura 49 segue o modelo extraído da ferramenta.

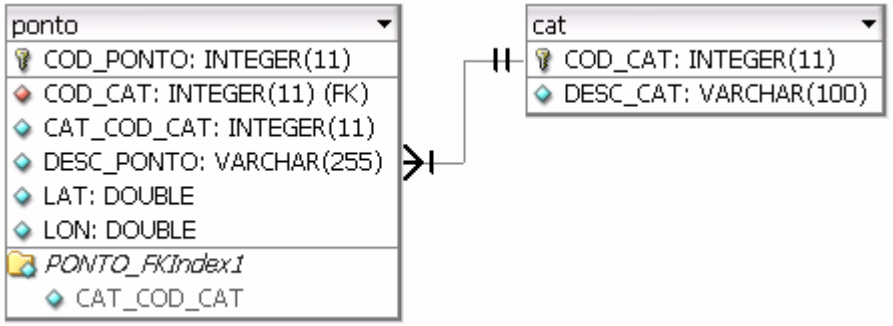


Figura 49 - Modelo de Dados

5.5.1. Caso Performático

Um fator performático para o sistema foi transferir o calculo de ordenação dos pontos por distância para o banco de dados efetuar, isso melhorou bastante o tempo de resposta. A principio estava previsto para o sistema trazer apenas os 20 mais próximos, com esse ganho foi possível sem se preocupar com performance trazer todos os pontos do banco ordenados de forma correta em relação a distância do receptor de GPS.

[14] Ver: http://www.mysqlbrasil.com.br/produtos/myisam2

6. CONSIDERAÇÕES FINAIS

Este capítulo contém as considerações finais sobre o trabalho, divididas em quatro subseções, a saber: dificuldades encontradas, resultados obtidos, conclusões e sugestões para trabalhos futuros.

6.1. Dificuldades Encontradas

Algumas dificuldades foram encontradas durante o desenvolvimento deste trabalho e serão descritas neste tópico com o intuito de explicitar a experiência do projeto, visando contribuir com trabalhos futuros.

A primeira dificuldade (e talvez a mais relevante) foi quanto à de por o protótipo funcionando em um Palm ou PocketPC. A forma de comunicação entre um PocketPC ou Palm é via InfraVermelho ou BlueTooth, com isso havia a necessidade de comprar um GPS com tal conexão, o que é muito difícil e caro, ou comprar um PocketPC com um receptor GPS Integrado.

Pode existir a mudança de plataforma de PC para PocketPC, todo o código escrito em VB.Net pode ser compilado para Windows Móbile e integrado a essa nova plataforma com pequenas alterações no código. O modulo que faz a requisição e o tratamento da resposta do GPS dos dados de latitude e longitude deverão ser adaptados a nova forma de comunicação.

Um outro ponto de interesse é que o banco de dados que se encontra em MySQL não possui versão que rode em Windows Móbile, tendo que ser migrado toda a base de dados para Access Móbile, isso é uma atividade simples de fazer mas perderá um pouco a performance do mesmo.

Outra dificuldade que se teve, diz respeito a grande quantidade de conceitos de GPS e análise de dados que precisaram ser estudadas e compreendidas para a correta utilização dos algoritmos relacionados a este tema. Também o estudo da viabilidade de utilizar o mapa de localização como se fosse esférico ou plano.

6.2. Resultados Obtidos

Foi criado uma funcionalidade que simula um ambiente real de coordenadas, sendo essencial aos testes dos algoritmos, ver Figura 50.

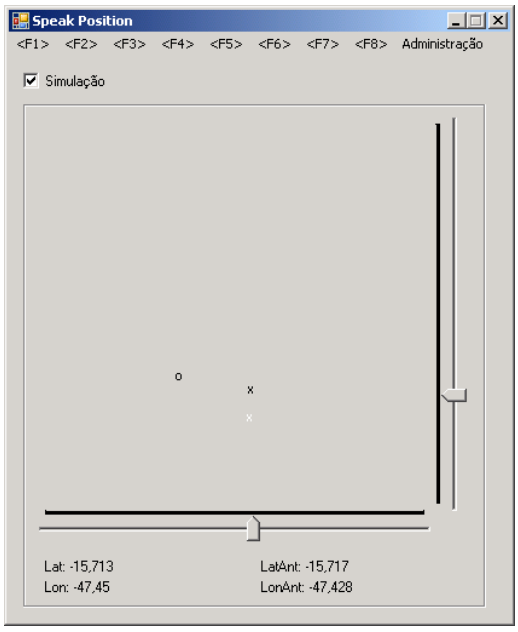


Figura 50 - Ambiente de teste

Um outro resultado obtido mais relevante com relação ao protótipo foi o tempo de resposta (performance) dos algoritmos. Por várias vezes houve a necessidade de se refatorar os códigos para que produzissem resultados aceitáveis em termos de tempo de processamento. Uma das soluções mais significativas foi a implementação de algoritmos para utilizar os mapas de forma plana quando as distâncias não eram

muito grandes, distâncias de até 50 quilômetros tinha um erro de precisão aceitável, acima disso a curvatura da terra já começa a influenciar na resposta.

O resultado obtido é satisfatório, a precisão está limitada aos dados recebidos pelo GPS por volta de 10 a 30 metros.

Foi efetuada também uma simulação no Parque da Cidade onde o autor criou um cenário. Primeiramente houve o cadastro de um ponto de interesse, logo após o autor se deslocou para longe do ponto e o sistema tinha como objetivo guia-lo até ao ponto de interesse cadastrado anteriormente. Como a precisão do GPS atualmente está em torno de 20 metros e a velocidade baixa, pois o autor está se deslocando a pé, a precisão ficou afetada e o sistema não conseguiu guiar para o ponto de interesse, mas o destino ficou próximo do que se queria. A distância do destino foi informada de forma satisfatória. Para resolver a questão requer uma precisão mais próxima da realidade com receptores mais robustos.

Deve-se considerar ainda alguns resultados secundários, como contribuições metodológicas e práticas deste trabalho, sendo:

a) Descrição detalhada de conceitos e fundamentos de processamento de áreas;
b) Desenvolvimento de algoritmos eficientes para processamento e análise de coordenadas;
c) Técnicas de enfileiramento;
d) Alto grau de encapsulamento e componentização do código-fonte do software, possibilitando reuso e utilização de objetos ou algoritmos de forma parcial por trabalhos futuros.

6.3. Conclusões

A implementação de um protótipo de posicionamento e gerenciamento de rotas para deficientes visuais, utilizando técnicas de processamento com coordenadas geográficas, receptores de GPS e saídas sonoras, conforme objetivos delineados e descritos no início deste trabalho foram sucedidos.

O método utilizado para calcular a distância entre dois pontos dados em coordenadas geográficas e o algoritmo para calculo de ângulo e gerenciamento de rota foi demonstrado de forma eficiente e prática.

A Utilização da tecnologia TTS – Text-to-Speach se mostrou bastante eficiente útil as mais diversas aplicações relativas a atividades sonoras. Não apenas para deficientes como também fornecer uma nova forma de interação com usuários de aplicações computacionais.

6.4. Sugestões de Trabalhos Futuros

Diversas linhas de pesquisa podem ser criadas a partir do trabalho aqui apresentado. As sugestões estão licitadas abaixo e serão divididas em dois grupos: “evoluções do trabalho atual” e “outras linhas de pesquisa”.

a) Evoluções do trabalho atual

- Adaptações para colocar o protótipo em um Palm, PocktPC, SmartFones;
- Utilização de mapas cartográficos e melhorar ainda mais o sistema de rotas;
- Criar um sistema de interpretação de voz para o deficiente falar o local desejado e o software reconhecer;
- Desenvolver a utilização da categoria e criar pontos de interesse ao deficiente, sugestões de apoio sem necessariamente estar previsto na rota.
- Criar mapas de níveis para tornar possível o desvio de obstáculos.

b) Outras linhas de pesquisa:

- Desenvolvimento de um sistema de leitura de jornal, revistas etc falado para utilização em veículos;
- Desenvolvimento de um sistema de navegação de uma cidade, falado, atualmente são utilizados mapas, torná-los sonoro;

Índice Remissivo

ângulo de deslocamento, 15, 41, 54, 55, 56, 57, 72
arco, 45, 46, 49, 50
Arquimedes, 47
Associação de Cegos Louis Braille, 14
circunferência, 46, 49, 50
contextualização, 13
coordenadas, 15, 23, 31, 32, 33, 34, 41, 42, 43, 44, 45, 55, 56, 63, 80, 81
coordenadas geográficas, 31
coplanares, 27, 30, 31
Co-senos, 56
deficiente visual, 13
diagrama, 61
distância entre dois pontos, 15, 41, 45, 81
elevação, 32, 33
engenharia da computação, 12
engenheiro, 12
equação, 22, 28, 29, 54, 57
Equador, 42
Eratóstenes, 47
estrutura do trabalho, 14
forças armadas, 18
funcionamento do GPS, 16
funcionamento do sistema, 25
geodésia, 37
geometria, 38, 49, 55
GPS, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 30, 33, 34, 35, 36, 37, 38, 39, 54, 59, 62, 63, 64, 71, 72, 73, 77, 78, 79, 80, 81, 85
Segmentos, 18
Greenwich, 31, 42, 44
interface, 36, 37, 63, 64
interseção, 26, 27, 28, 29, 42, 43, 50
intuito, 14
latitude, 13, 15, 32, 35, 41, 43, 44, 45, 54, 55, 57, 62, 71, 73, 77, 79
linhas imaginárias, 42
longitude, 15, 31, 32, 35, 43, 44, 45, 54, 55, 57, 62, 71, 73, 77, 79
microprocessador, 36
motivação, 14
Navstar, 17
Notebook, 65
objetivo do projeto, 13
ondas eletromagnéticas, 36
portadoras, 22, 25, 37
precisão, 13, 14, 17, 23, 24, 33, 35, 38, 39, 47, 49, 62, 65, 80
problema de locomoção, 13
protótipo, 15, 61, 62, 65, 66, 79, 80, 81
pseudo-distâncias, 22, 35
qualidade de vida, 13
receptores, 17, 18, 22, 23, 24, 25, 27, 35, 36, 37, 39, 72, 81
respostas sonoras, 13
reta, 45, 48, 50, 57
rotação, 41, 42
satélites, 16, 17, 19, 20, 21, 22, 23, 25, 26, 27, 33, 35, 36, 38, 39, 54
segmento de controle, 23
segmento de usuários, 24

segmento espacial, 19
sentido horário, 56, 57
sistema de posicionamento global, 16
sistema linear, 30, 31, 34
sociedade, 12, 14
Sputinik I, 17
superfícies esféricas, 27, 30, 34
tangente, 48, 50
tempo de chegada, 15, 41, 59
teorema, 29, 30
Terra, 19, 20, 23, 26, 31, 32, 33, 37, 39, 41, 42, 44, 45, 46, 47, 48, 49
triangulação, 33, 35
trigonometria, 47, 49, 50, 51
Trigonometria esférica, 52
Trigonometria plana, 50
unidades eletrônicas, 36
Vanguard, 17
velocidade, 13, 15, 17, 22, 26, 34, 40, 41, 58, 59, 60, 64, 67, 73, 77
velocidade da luz, 26, 34, 40
velocidade de deslocamento, 15, 41

REFERÊNCIAS BIBLIOGRÁFICAS

ÁVILA, Geraldo, VÁRIAS FACES DA MATEMÁTICA, Editora Blucher, 2007.

GUTERRES, P. Cortez. Projeto de Software sincronizado com GPS para fiscalização de estações geradoras de freqüência. Centro Universitário de Brasília – UniCEUB, Faculdade de Ciências Exatas e de Tecnologia – FAET, Curso de Engenharia da Computação 2005

LETHAM, L. GPS Made easy: using global positioning systems in the outdoors. Seattle: Published by The Mountaineers, 1996.

MONICO, J.F.G. 2000. Posicionamento pelo NAVSTAR-GPS: descrição, fundamentos e aplicações. São Paulo: Editora UNESP.

NMEA Data. On Line. (http://www.gpsinformation.org/dale/nmea.htm) - Outubro 2007

ROSA, Roberto. Cartografia Básica, Universidade Federal de Uberlândia, Laboratório de Geoprocessamento, 2004. (http://www.ig.ufu.br/lgeop/Apostilas/Cartografia.pdf)

SEEBER, G. Satellite geodesy: foundations, methods and applications. Berlin, New York: Walter de Gruyter, 1993.

SEGANTINE, P. C. L. 1999. GPS – Sistema de Posicionamento Global. Apostila didática da Universidade de São Paulo Escola de Engenharia de São Carlos, Departamento de Transportes.

TAFNER, Malcon Anderson; LOESCH, Claudio; STRINGARI, Sérgio. Comunicação de dados usando linguagem "C". Blumenau: FURB, 1996.

TORRES, Gabriel. PWM. On Line. 2007.(http://www.clubedohardware.com.br)

http/www.mar.mil.br/dhn/bhmn/download/apcap17.pdf, Acessado em 20/11/2007

ANEXO A

clsCategoria.vb

```
Public Class clsCategoria

    Private m_CodCategoria As Integer
    Private m_Descricao As String

    Public Sub New(ByVal p_CodCategoria As Integer)
        Dim octlCategoria As New ctlCategoria
        Dim oCategoria As clsCategoria
        oCategoria = octlCategoria.FindByPrimaryKey(p_CodCategoria)
        m_CodCategoria = oCategoria.CodCategoria
        m_Descricao = oCategoria.Descricao
    End Sub

    Public Sub New()

    End Sub

    Public Property CodCategoria() As Integer
        Get
            CodCategoria = m_CodCategoria
        End Get
        Set(ByVal Value As Integer)
            m_CodCategoria = Value
        End Set
    End Property

    Public Property Descricao() As String
        Get
            Descricao = m_Descricao
        End Get
        Set(ByVal Value As String)
            m_Descricao = Value
        End Set
    End Property

    Protected Overrides Sub Finalize()
        MyBase.Finalize()
    End Sub
End Class
```

clsPonto.vb

```
Public Class clsPonto
    Private m_CodPonto As Integer
    Private m_Categoria As clsCategoria
    Private m_Descricao As String
    Private m_Latitude As Double
    Private m_Longitude As Double
    Private m_Tipo As Int16
    Private m_Caminho As String

    Public Sub New(ByVal p_CodPonto As Integer)
```

```
    Dim octlPonto As New ctlPonto
    Dim oPonto As clsPonto
    oPonto = octlPonto.FindByPrimaryKey(p_CodPonto)
    m_CodPonto = oPonto.CodPonto
    m_Categoria = oPonto.Categoria
    m_Descricao = oPonto.Descricao
    m_Latitude = oPonto.Latitude
    m_Longitude = oPonto.Longitude
    m_Tipo = oPonto.Tipo
    m_Caminho = oPonto.Caminho
End Sub

Public Sub New()
    m_Categoria = New clsCategoria
End Sub

Public Property CodPonto() As Integer
    Get
        CodPonto = m_CodPonto
    End Get
    Set(ByVal Value As Integer)
        m_CodPonto = Value
    End Set
End Property

Public Property Categoria() As clsCategoria
    Get
        Categoria = m_Categoria
    End Get
    Set(ByVal Value As clsCategoria)
        m_Categoria = Value
    End Set
End Property

Public Property Descricao() As String
    Get
        Descricao = m_Descricao
    End Get
    Set(ByVal Value As String)
        m_Descricao = Value
    End Set
End Property

Public Property Latitude() As Double
    Get
        Latitude = m_Latitude
    End Get
    Set(ByVal Value As Double)
        m_Latitude = Value
    End Set
End Property

Public Property Longitude() As Double
    Get
        Longitude = m_Longitude
    End Get
    Set(ByVal Value As Double)
        m_Longitude = Value
    End Set
End Property
```

```
    Public Property Tipo() As Int16
        Get
            Tipo = m_Tipo
        End Get
        Set(ByVal Value As Int16)
            m_Tipo = Value
        End Set
    End Property

    Public Property Caminho() As String
        Get
            Caminho = m_Caminho
        End Get
        Set(ByVal Value As String)
            m_Caminho = Value
        End Set
    End Property

End Class
```

colCategoria.vb

```
Public Class colCategoria
    Private oCol As Collection

    Public Sub New()
        oCol = New Collection
    End Sub

    Public Sub Add(ByVal poCategoria As clsCategoria)
        oCol.Add(poCategoria, poCategoria.CodCategoria)
    End Sub

    Public Sub Remove(ByVal psCodCategoria As String)
        oCol.Remove(psCodCategoria)
    End Sub

    Public Sub Remove(ByVal piIndex As Integer)
        oCol.Remove(piIndex)
    End Sub

    Public Function Count() As Integer
        Count = oCol.Count()
    End Function

    Public Function Item(ByVal piIndex As Integer) As clsCategoria
        Item = oCol.Item(piIndex)
    End Function
End Class
```

colPonto.vb

```
Public Class colPonto
    Private oCol As Collection

    Public Sub New()
        oCol = New Collection
    End Sub

    Public Sub Add(ByVal poPonto As clsPonto)
        oCol.Add(poPonto, poPonto.CodPonto)
```

```
    End Sub

    Public Sub Remove(ByVal psCodPonto As String)
        oCol.Remove(psCodPonto)
    End Sub

    Public Sub Remove(ByVal piIndex As Integer)
        oCol.Remove(piIndex)
    End Sub

    Public Function Count() As Integer
        Count = oCol.Count()
    End Function

    Public Function Item(ByVal piIndex As Integer) As clsPonto
        Item = oCol.Item(piIndex)
    End Function
End Class
```

ctlCategoria.vb

```
Public Class ctlCategoria

    Private strSQL As String
    Private iCont As Integer

    Public Function Insert(ByRef objCategoria As clsCategoria) As
clsCategoria
        strSQL = " INSERT INTO CAT (COD_CAT, DESC_CAT) " & _
                 " VALUES ( " & _
                 "" & objCategoria.CodCategoria & ", " & _
                 "'" & objCategoria.Descricao & "') "
        If objDB.ExecSQL(strSQL) = True Then
            Insert = Me.FindByPrimaryKey(objCategoria.CodCategoria)
        Else
            'Erro
            Insert = Nothing
        End If

    End Function

    Public Function Insert(ByVal DescCategoria As String) As clsCategoria

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader
        Dim CodCategoria As Integer

        strSQL = "SELECT (MAX(COD_CAT)+1) AS SOMA FROM CAT"
        ObjReader = objDB.ExecSQLDataReader(strSQL)
        If ObjReader.Read() Then
            CodCategoria = ObjReader.Item("SOMA")
        Else
            CodCategoria = 1
        End If
        ObjReader.Close()

        strSQL = " INSERT INTO CAT (COD_CAT, DESC_CAT) " & _
                 " VALUES ( " & _
                 "" & CodCategoria & ", " & _
                 "'" & DescCategoria & "') "
```

```
        If objDB.ExecSQL(strSQL) = True Then
            Insert = Me.FindByPrimaryKey(CodCategoria)
        Else
            'Erro
            Insert = Nothing
        End If

    End Function

    Public Function Alterar(ByVal CodCategoria As Integer, _
                            ByVal Descricao As String) As clsCategoria

        strSQL = " UPDATE CAT SET DESC_CAT = '" & Descricao & "'" & _
                 " WHERE COD_CAT = " & CodCategoria

        If objDB.ExecSQL(strSQL) = True Then
            Alterar = Me.FindByPrimaryKey(CodCategoria)
        Else
            'Erro
            Alterar = Nothing
        End If

    End Function

    Public Sub Excluir(ByVal CodCategoria As Integer)

        strSQL = " DELETE FROM CAT " & _
                 " WHERE COD_CAT = " & CodCategoria

        If objDB.ExecSQL(strSQL) = False Then
            'ERRO
        End If

    End Sub

    Public Function FindAll(Optional ByVal sWhere As String = "", _
                            Optional ByVal sOrderBy As String = "") As
colCategoria

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader
        Dim objCategoriaS As colCategoria
        Dim objCategoria As clsCategoria

        strSQL = " SELECT * FROM CAT "

        If Len(Trim(sWhere)) > 0 Then
            strSQL = strSQL & " WHERE " & sWhere
        End If

        If Len(Trim(sOrderBy)) > 0 Then
            strSQL = strSQL & " ORDER BY " & sOrderBy
        End If

        ObjReader = objDB.ExecSQLDataReader(strSQL)
        objCategoriaS = New colCategoria

        While ObjReader.Read() = True
            objCategoria = New clsCategoria
            objCategoria.CodCategoria = ObjReader.Item("COD_CAT")
            objCategoria.Descricao = ObjReader.Item("DESC_CAT")
            objCategoriaS.Add(objCategoria)
```

```
        End While
        ObjReader.Close()

        FindAll = objCategoriaS

    End Function

    Public Function FindAllDataSource(Optional ByVal sWhere As String = "",
_
                                      Optional ByVal sOrderBy As String =
"") As System.Data.DataTable

        strSQL = " SELECT * FROM CAT "

        If Len(Trim(sWhere)) > 0 Then
            strSQL = strSQL & " WHERE " & sWhere
        End If

        If Len(Trim(sOrderBy)) > 0 Then
            strSQL = strSQL & " ORDER BY " & sOrderBy
        End If

        FindAllDataSource = objDB.ExecSQLDataTable(strSQL)

    End Function

    Public Function FindByPrimaryKey(ByVal CodCategoria As Integer) As
clsCategoria

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader

        strSQL = " SELECT * FROM CAT WHERE COD_CAT = " & CodCategoria

        ObjReader = objDB.ExecSQLDataReader(strSQL)
        If ObjReader.Read() = True Then
            FindByPrimaryKey = New clsCategoria
            FindByPrimaryKey.CodCategoria = ObjReader.Item("COD_CAT")
            FindByPrimaryKey.Descricao = ObjReader.Item("DESC_CAT")
        Else
            FindByPrimaryKey = Nothing
        End If
        ObjReader.Close()

    End Function

End Class
```

ctlPonto.vb

```
Public Class ctlPonto

        Private strSQL As String
        Private iCont As Integer

    Public Function Insert(ByRef objPonto As clsPonto) As clsPonto
        strSQL = " INSERT INTO PONTO (COD_PONTO, CAT_COD_CAT, DESC_PONTO,
LAT, LON, TIP, CAMINHO) " & _
                 " VALUES ( " & _
                 "" & objPonto.CodPonto & ", " & _
```

```
                "" & objPonto.Categoria.CodCategoria & ", " & _
                "'" & objPonto.Descricao & "', " & _
                "" & Str(objPonto.Latitude) & ", " & _
                "" & Str(objPonto.Longitude) & ", " & _
                "" & objPonto.Tipo & ", " & _
                "'" & objPonto.Caminho & "') "
        If objDB.ExecSQL(strSQL) = True Then
            Insert = Me.FindByPrimaryKey(objPonto.CodPonto)
        Else
            'Erro
            Insert = Nothing
        End If


    End Function

    Public Function Insert(ByVal CodCategoria As Integer, _
                           ByVal Descricao As String, _
                           ByVal Latitude As Double, _
                           ByVal Longitude As Double, _
                           ByVal Tipo As Integer, _
                           ByVal Caminho As String) As clsPonto

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader
        Dim CodPonto As Integer

        strSQL = "SELECT (MAX(COD_PONTO)+1) AS SOMA FROM PONTO"
        ObjReader = objDB.ExecSQLDataReader(strSQL)
        If ObjReader.Read() Then
            CodPonto = ObjReader.Item("SOMA")
        Else
            CodPonto = 1
        End If
        ObjReader.Close()

        strSQL = " INSERT INTO PONTO (COD_PONTO, CAT_COD_CAT, DESC_PONTO,
LAT, LON, TIP, CAMINHO) " & _
                " VALUES ( " & _
                "" & CodPonto & ", " & _
                "" & CodCategoria & ", " & _
                "'" & Descricao & "', " & _
                "" & Str(Latitude) & ", " & _
                "" & Str(Longitude) & ", " & _
                "" & Tipo & ", " & _
                "'" & Caminho & "') "
        If objDB.ExecSQL(strSQL) = True Then
            Insert = Me.FindByPrimaryKey(CodPonto)
        Else
            'Erro
            Insert = Nothing
        End If

    End Function

    Public Function Alterar(ByVal CodPonto As Integer, _
                            ByVal CodCategoria As Integer, _
                            ByVal Descricao As String) As clsPonto

        strSQL = " UPDATE PONTO SET CAT_COD_CAT  = " & CodCategoria & "," &
_
                " DESC_PONTO = '" & Descricao & "'" & _
```

```
                " WHERE COD_PONTO = " & CodPonto

        If objDB.ExecSQL(strSQL) = True Then
            Alterar = Me.FindByPrimaryKey(CodPonto)
        Else
            'Erro
            Alterar = Nothing
        End If

    End Function

    Public Sub Excluir(ByVal CodPonto As Integer)

        strSQL = " DELETE FROM PONTO " & _
                 " WHERE COD_PONTO = " & CodPonto

        If objDB.ExecSQL(strSQL) = False Then
            'ERRO
        End If

    End Sub

    Public Function FindAll(Optional ByVal sWhere As String = "", _
                            Optional ByVal sOrderBy As String = "") As
colPonto

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader
        Dim objPontoS As colPonto
        Dim objPonto As clsPonto

        strSQL = " SELECT * FROM PONTO LEFT JOIN CAT ON PONTO.CAT_COD_CAT =
CAT.COD_CAT "

        If Len(Trim(sWhere)) > 0 Then
            strSQL = strSQL & " WHERE " & sWhere
        End If

        If Len(Trim(sOrderBy)) > 0 Then
            strSQL = strSQL & " ORDER BY " & sOrderBy
        End If

        ObjReader = objDB.ExecSQLDataReader(strSQL)
        objPontoS = New colPonto

        While ObjReader.Read() = True
            objPonto = New clsPonto
            objPonto.CodPonto = ObjReader.Item("COD_PONTO")
            objPonto.Categoria.CodCategoria = ObjReader.Item("CAT_COD_CAT")
            objPonto.Categoria.Descricao = "" & ObjReader.Item("DESC_CAT")
            objPonto.Descricao = ObjReader.Item("DESC_PONTO")
            objPonto.Latitude = ObjReader.Item("LAT")
            objPonto.Longitude = ObjReader.Item("LON")
            objPonto.Tipo = ObjReader.Item("TIP")
            objPonto.Caminho = ObjReader.Item("CAMINHO")
            objPontoS.Add(objPonto)
        End While
        ObjReader.Close()

        FindAll = objPontoS

    End Function
```

```
    Public Function FindAllDataSource(Optional ByVal sWhere As String = "",
_
                                      Optional ByVal sOrderBy As String =
"") As System.Data.DataTable

        strSQL = " SELECT * FROM PONTO LEFT JOIN CAT ON PONTO.CAT_COD_CAT =
CAT.COD_CAT "

        If Len(Trim(sWhere)) > 0 Then
            strSQL = strSQL & " WHERE " & sWhere
        End If

        If Len(Trim(sOrderBy)) > 0 Then
            strSQL = strSQL & " ORDER BY " & sOrderBy
        End If

        FindAllDataSource = objDB.ExecSQLDataTable(strSQL)

    End Function

    Public Function FindByPrimaryKey(ByVal CodPonto As Integer) As clsPonto

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader

        strSQL = " SELECT PONTO.*, CAT.DESC_CAT FROM PONTO LEFT JOIN CAT ON
PONTO.CAT_COD_CAT = CAT.COD_CAT WHERE PONTO.COD_PONTO = " & CodPonto

        Try
            ObjReader = objDB.ExecSQLDataReader(strSQL)
            If ObjReader.Read() = True Then
                FindByPrimaryKey = New clsPonto
                FindByPrimaryKey.CodPonto = ObjReader.Item("COD_PONTO")
                FindByPrimaryKey.Categoria.CodCategoria =
ObjReader.Item("CAT_COD_CAT")
                FindByPrimaryKey.Categoria.Descricao = "" &
ObjReader.Item("DESC_CAT")
                FindByPrimaryKey.Descricao = ObjReader.Item("DESC_PONTO")
                FindByPrimaryKey.Latitude = ObjReader.Item("LAT")
                FindByPrimaryKey.Longitude = ObjReader.Item("LON")
                FindByPrimaryKey.Tipo = ObjReader.Item("TIP")
                FindByPrimaryKey.Caminho = ObjReader.Item("CAMINHO")
            Else
                FindByPrimaryKey = Nothing
            End If
            ObjReader.Close()
        Catch err As Exception
            FindByPrimaryKey = Nothing
            MsgBox(err.Message)
        End Try

    End Function

    Public Function FindAllPontosProximos(ByVal lLatitude As Double, _
                                          ByVal lLongitude As Double, _
                                          Optional ByVal bTodos As Boolean
= False) As colPonto

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader
        Dim objPontoS As colPonto
        Dim objPonto As clsPonto
```

```
        Dim iQuantidadeRegs As Integer

        strSQL = " SELECT PONTO.*, CAT.DESC_CAT, " & _
                 "((" & Str(lLatitude) & " - LAT)*(" & Str(lLatitude) & " -
LAT) + (" & Str(lLongitude) & " - LON)*(" & Str(lLongitude) & " - LON))  AS
DIST " & _
                 " FROM PONTO LEFT JOIN CAT ON PONTO.CAT_COD_CAT =
CAT.COD_CAT ORDER BY DIST "

        ObjReader = objDB.ExecSQLDataReader(strSQL)
        objPontoS = New colPonto
        iQuantidadeRegs = 20

        While ObjReader.Read() = True
            If iQuantidadeRegs >= 0 Or bTodos = True Then
                objPonto = New clsPonto
                objPonto.CodPonto = ObjReader.Item("COD_PONTO")
                objPonto.Categoria.CodCategoria =
ObjReader.Item("CAT_COD_CAT")
                objPonto.Categoria.Descricao = "" &
ObjReader.Item("DESC_CAT")
                objPonto.Descricao = ObjReader.Item("DESC_PONTO")
                objPonto.Latitude = ObjReader.Item("LAT")
                objPonto.Longitude = ObjReader.Item("LON")
                objPonto.Tipo = ObjReader.Item("TIP")
                objPonto.Caminho = ObjReader.Item("CAMINHO")
                objPontoS.Add(objPonto)
                iQuantidadeRegs = iQuantidadeRegs - 1
            End If
        End While
        ObjReader.Close()

        FindAllPontosProximos = objPontoS

    End Function

    Public Function FindAllPontosProximossWhere(ByVal lLatitude As Double,
_
                                                ByVal lLongitude As Double,
_
                                                Optional ByVal sWhere As
String = "") As colPonto

        Dim ObjReader As MySql.Data.MySqlClient.MySqlDataReader
        Dim objPontoS As colPonto
        Dim objPonto As clsPonto


        If Len(Trim(sWhere)) > 0 Then
            strSQL = " SELECT PONTO.*, CAT.DESC_CAT, " & _
                              "((" & Str(lLatitude) & " - LAT)*(" &
Str(lLatitude) & " - LAT) + (" & Str(lLongitude) & " - LON)*(" &
Str(lLongitude) & " - LON))  AS DIST " & _
                              " FROM PONTO LEFT JOIN CAT ON
PONTO.CAT_COD_CAT = CAT.COD_CAT " & _
                              " WHERE " & sWhere & _
                              " ORDER BY DIST "
        Else
            strSQL = " SELECT PONTO.*, CAT.DESC_CAT, " & _
```

```
                                "((" & Str(lLatitude) & " - LAT)*(" &
Str(lLatitude) & " - LAT) + (" & Str(lLongitude) & " - LON)*(" &
Str(lLongitude) & " - LON))  AS DIST " & _
                                " FROM PONTO LEFT JOIN CAT ON
PONTO.CAT_COD_CAT = CAT.COD_CAT ORDER BY DIST "
        End If

        ObjReader = objDB.ExecSQLDataReader(strSQL)
        objPontoS = New colPonto

        While ObjReader.Read() = True
            objPonto = New clsPonto
            objPonto.CodPonto = ObjReader.Item("COD_PONTO")
            objPonto.Categoria.CodCategoria = ObjReader.Item("CAT_COD_CAT")
            objPonto.Categoria.Descricao = "" & ObjReader.Item("DESC_CAT")
            objPonto.Descricao = ObjReader.Item("DESC_PONTO")
            objPonto.Latitude = ObjReader.Item("LAT")
            objPonto.Longitude = ObjReader.Item("LON")
            objPonto.Tipo = ObjReader.Item("TIP")
            objPonto.Caminho = ObjReader.Item("CAMINHO")
            objPontoS.Add(objPonto)
        End While
        ObjReader.Close()

        FindAllPontosProximossWhere = objPontoS

    End Function

End Class
```

DBConection.vb

```
Imports MySql.Data.MySqlClient
Public Class DBConection

    Dim conn As MySqlConnection
    Dim myCommand As MySqlCommand
    Dim myAdapter As MySqlDataAdapter
    Dim myData As DataTable

    Public Function Open() As Boolean
        conn = New MySqlConnection
        myCommand = New MySqlCommand
        myAdapter = New MySqlDataAdapter
        conn.ConnectionString = "server=localhost;user
id=root;password=;database=position"
        Try
            conn.Open()
            Open = True
        Catch myerror As MySqlException
            Open = False
            'MessageBox.Show("Erro ao conectar com o Banco de dados : " &
myerror.Message)
        End Try
    End Function

    Public Function ExecSQL(ByVal strSQL As String) As Boolean
        If conn.State = ConnectionState.Closed Then
            If Me.Open() = False Then
```

```
                ExecSQL = False
                Exit Function
            End If
        End If
        Try
            myCommand.Connection = conn
            myCommand.CommandText = strSQL
            myCommand.ExecuteNonQuery()
            ExecSQL = True
        Catch myerror As MySqlException
            ExecSQL = False
            MessageBox.Show("Erro : " & myerror.Message)
        End Try
    End Function

    Public Function ExecSQLDataReader(ByVal strSQL As String) As
MySql.Data.MySqlClient.MySqlDataReader
        If conn.State = ConnectionState.Closed Then
            If Me.Open() = False Then
                ExecSQLDataReader = Nothing
                Exit Function
            End If
        End If
        Try
            myCommand.Connection = conn
            myCommand.CommandText = strSQL
            ExecSQLDataReader = myCommand.ExecuteReader()
        Catch myerror As MySqlException
            ExecSQLDataReader = Nothing
            'MessageBox.Show("Erro ao conectar com o Banco de dados : " &
myerror.Message)
        End Try
    End Function

    Public Function ExecSQLDataTable(ByVal strSQL As String) As DataTable
        myData = New DataTable
        If conn.State = ConnectionState.Closed Then
            If Me.Open() = False Then
                ExecSQLDataTable = Nothing
                Exit Function
            End If
        End If
        Try
            myCommand.Connection = conn
            myCommand.CommandText = strSQL
            myAdapter.SelectCommand = myCommand
            myAdapter.Fill(myData)
            ExecSQLDataTable = myData
        Catch myerror As MySqlException
            ExecSQLDataTable = Nothing
            'MessageBox.Show("Erro ao conectar com o Banco de dados : " &
myerror.Message)
        End Try
    End Function

    Public Sub New()
        Open()
    End Sub
End Class
```

ModPrincipal.vb

```
Imports ACTIVEVOICEPROJECTLib
Module ModPrincipal

    Public objDB As DBConection
    Public oPosicao As NegPosicao

    Public oVoz As New DirectSS
    Public iNumeroVoz As Integer

    Public tLat As Double
    Public tLon As Double
    Public tLatAnt As Double
    Public tLonAnt As Double

    Public iCodPontoBusca As Integer

    Public iDeslocamento As Integer
    Public strAnguloDeslocamento As String
    Public strVelocidade As String
    Public strTempoChegada As String


    Private Const rad As Double = 0.01745329252


    Public Function CalculaDistancia(ByVal LatitudeA As Double, _
                                     ByVal LongitudeA As Double, _
                                     ByVal LatitudeB As Double, _
                                     ByVal LongitudeB As Double) As Integer

        Dim Alfa, S, DLambda, PhiUm, PhiDois As Double

        DLambda = (LongitudeA - LongitudeB) * rad
        PhiUm = (90 - LatitudeA) * rad
        PhiDois = (90 - LatitudeB) * rad

        Alfa = Math.Cos(PhiDois) * Math.Cos(PhiUm) + Math.Sin(PhiDois) *
Math.Sin(PhiUm) * Math.Cos(DLambda)
        If Alfa < 1 Then
            Alfa = Math.Acos(Alfa)
            S = 0 + 6378160 * Alfa
        Else
            S = 0
        End If
        CalculaDistancia = CInt(S)
    End Function.
```

Função 1 - Calculo de Distância

```
    Public Sub RetornaLatLongGPS(ByRef Latitude As Double, _
                                 ByRef Longitude As Double)
        Latitude = tLat
        Longitude = tLon
    End Sub
```

```
    Public Sub Falar(ByVal sTexto As String, ByVal bPararSeFalando As
Boolean)
        If oVoz.Speaking = 0 Or (oVoz.Speaking > 0 And bPararSeFalando =
True) Then
            If oVoz.Speaking > 0 Then ResetFala()
            oVoz.Speak(sTexto)
        End If
    End Sub

    Public Sub ResetFala()
        oVoz.AudioReset()
        oVoz = New DirectSS
        MudarVoz()
    End Sub

    Public Sub MudarVoz()
        oVoz.CurrentMode = 1
        If (oVoz.Gender(oVoz.CurrentMode) = 1) Then
            oVoz.LipType = 0
        Else
            oVoz.LipType = 1
        End If
        Falar("", True)
        oVoz.CurrentMode = iNumeroVoz
        If (oVoz.Gender(oVoz.CurrentMode) = 1) Then
            oVoz.LipType = 0
        Else
            oVoz.LipType = 1
        End If
    End Sub


    Public Function CalculaAngulo(ByVal LatAtual As Double, _
                                   ByVal LonAtual As Double, _
                                   ByVal LatAnterior As Double, _
                                   ByVal LonAnterior As Double, _
                                   ByVal LatPonto As Double, _
                                   ByVal LonPonto As Double) As String
        Dim a, b, c As Double

        Dim Beta As Double
        Dim strSentido As String

        c = CalculaDistancia(LatAtual, LonAtual, LatAnterior, LonAnterior)
        a = CalculaDistancia(LatAtual, LonAtual, LatPonto, LonPonto)
        b = CalculaDistancia(LatPonto, LonPonto, LatAnterior, LonAnterior)

        Beta = ((a ^ 2) + (c ^ 2) - (b ^ 2)) / (2 * a * c)
        Beta = Math.Acos(Beta) / rad

        If Beta <= 15 Then
            Beta = 0
        ElseIf Beta > 15 And Beta <= 45 Then
            Beta = 30
        ElseIf Beta > 45 And Beta <= 75 Then
            Beta = 60
        ElseIf Beta > 75 And Beta <= 105 Then
            Beta = 90
        ElseIf Beta > 105 And Beta <= 135 Then
            Beta = 120
        ElseIf Beta > 135 And Beta <= 165 Then
```

```
        Beta = 150
    ElseIf Beta > 165 Then
        Beta = 180
    End If

    strSentido = CalculaDeslocamento(LatAtual, _
                 LonAtual, _
                 LatAnterior, _
                 LonAnterior, _
                 LatPonto, _
                 LonPonto)
    If Beta = 0 Then
        CalculaAngulo = "siga em frente"
    Else
        CalculaAngulo = Beta & " graus para a " & strSentido
    End If
End Function.
```

Função 2 - Angulo de Deslocamento

```
Private Function CalculaDeslocamento(ByVal LatAtual As Double, _
                                     ByVal LonAtual As Double, _
                                     ByVal LatAnterior As Double, _
                                     ByVal LonAnterior As Double, _
                                     ByVal LatPonto As Double, _
                                     ByVal LonPonto As Double) As String
    Dim m As Double
    Dim pTeste As Double
    Dim bSobe As Boolean

    'm=(y1 - y0)/(x1-x0)
    m = (LatAtual - LatAnterior) / (LonAtual - LonAnterior)
    'y = y0 + m(x-x0)
    pTeste = LatAnterior + m * (LonPonto - LonAnterior)

    If LatAtual > LatAnterior Then
        bSobe = True
    Else
        bSobe = False
    End If

    If (LatPonto > pTeste And bSobe = True And m > 0) Then
        CalculaDeslocamento = "Esquerda"
    ElseIf (LatPonto > pTeste And bSobe = False And m > 0) Then
        CalculaDeslocamento = "Direita"
    ElseIf (LatPonto < pTeste And bSobe = True And m > 0) Then
        CalculaDeslocamento = "Direita"
    ElseIf (LatPonto < pTeste And bSobe = False And m > 0) Then
        CalculaDeslocamento = "Esquerda"
    ElseIf (LatPonto > pTeste And bSobe = True And m < 0) Then
        CalculaDeslocamento = "Direita"
    ElseIf (LatPonto > pTeste And bSobe = False And m < 0) Then
        CalculaDeslocamento = "Esquerda"
    ElseIf (LatPonto < pTeste And bSobe = True And m < 0) Then
        CalculaDeslocamento = "Esquerda"
    ElseIf (LatPonto < pTeste And bSobe = False And m < 0) Then
        CalculaDeslocamento = "Direita"
    Else
        CalculaDeslocamento = ""
    End If
```

```
    End Function.
```

Função 3 - Sentido de deslocamento

```
    Public Function CalculaVelocidadeDeslocamento(ByVal LatAtual As Double,
_
                                                  ByVal LonAtual As Double,
_
                                                  ByVal LatAnterior As
Double, _
                                                  ByVal LonAnterior As
Double, _
                                                  ByVal Tempo As Integer)
As Integer
        Dim Dist As Double

        Dist = CalculaDistancia(LatAtual, LonAtual, LatAnterior,
LonAnterior)
        CalculaVelocidadeDeslocamento = Dist / Tempo

    End Function.
```

Função 4 - Velocidade de deslocamento

```
    Public Function CalculaTempoChegada(ByVal LatAtual As Double, _
                                        ByVal LonAtual As Double, _
                                        ByVal LatPonto As Double, _
                                        ByVal LonPonto As Double, _
                                        ByVal Velocidade As Integer) As
Integer
        Dim Dist As Double

        Dist = CalculaDistancia(LatAtual, LonAtual, LatPonto, LonPonto)
        If Velocidade > 0 Then
            CalculaTempoChegada = Dist / Velocidade
        Else
            CalculaTempoChegada = 0
        End If
    End Function.
```

Função 5 - Tempo de chegada

```
End Module
```

ModuleGPS.vb

```
Option Strict Off
Option Explicit On
Option Compare Binary
Module ModuleGPS

    Public Current_Command As String

    Public Const Pid_Ack_Byte As Short = 6
    Public Const Pid_Command_Data As Short = 10
    Public Const Pid_Position_Data As Short = 17
```

```
    Public Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As
Integer)

    Public Sub Request_Pos()
        Envoi((Chr(&H10S) & Chr(&HAS) & Chr(2) & Chr(2) & Chr(0) &
Chr(&HF2S) & Chr(&H10S) & Chr(3)))
    End Sub

    Public Sub Envoi(ByRef mot As String)
        Dim n As Object
        On Error Resume Next
        Do
            System.Windows.Forms.Application.DoEvents()
        Loop While (frmPrincipal.GARMIN.OutBufferCount > 0)
        frmPrincipal.GARMIN.Output = mot
        Sleep((400))
    End Sub

    Public Function Calcul_Position(ByRef packet() As Byte, ByRef j As
Short) As String
        Dim dblChaine As Double
        Dim i As Short
        Dim SolModulo, SolDivision As Double
        Dim res As String
        Dim s As Short
        Dim exposant As Integer
        Dim PI, mantisse As Double

        PI = 3.14159265358979

        dblChaine = CShort(Val(CStr(packet(j + 6))) And &HFS) * 16 ^ 12 +
(Val(CStr(packet(j + 5))) * 16 ^ 10) + (Val(CStr(packet(j + 4))) * 16 ^ 8)
+ (Val(CStr(packet(j + 3))) * 16 ^ 6) + (Val(CStr(packet(j + 2))) * 16 ^ 4)
+ (Val(CStr(packet(j + 1))) * 16 ^ 2) + Val(CStr(packet(j)))
        For i = 51 To 0 Step -1
            SolDivision = dblChaine / 2
            SolModulo = Fix(SolDivision)
            dblChaine = SolModulo
            If SolDivision = SolModulo Then
                res = 0 & res
            Else
                res = 1 & res
            End If
        Next i

        dblChaine = Val(CStr(packet(j + 7)))
        If (dblChaine And &H80S) = &H80S Then
            s = -1
        Else
            s = 1
        End If

        exposant = ((CShort(Val(CStr(packet(j + 7))) And &H7FS) * 16 +
(Val(CStr(packet(j + 6))) And &HF0S) \ 16) - 1023)

        mantisse = dblMantisse(res) * 180 / PI
        Calcul_Position = CStr(s * mantisse * 2 ^ exposant)

    End Function
```

```
    Public Function dblMantisse(ByRef str_Renamed As String) As Double
        Dim i As Short
        Dim p As Short
        Dim mantisse As Double

        p = 0
        For i = 1 To 51
            p = p + 1
            If Mid(str_Renamed, i, 1) = "1" Then mantisse = mantisse + 2 ^
-p
        Next
        mantisse = 1 + mantisse
        dblMantisse = mantisse
    End Function

End Module
```

NegPontoProximo.vb

```
Public Class NegPontoProximo
    Dim objPonto As clsPonto
    Dim iDistancia As Integer

    Public Sub New(ByVal pobjPonto As clsPonto, ByVal piDistancia As
Integer)
        objPonto = pobjPonto
        iDistancia = piDistancia
    End Sub

    Public Property Ponto() As clsPonto
        Get
            Ponto = objPonto
        End Get
        Set(ByVal Value As clsPonto)
            objPonto = Value
        End Set
    End Property

    Public Property Distancia() As Integer
        Get
            Distancia = iDistancia
        End Get
        Set(ByVal Value As Integer)
            iDistancia = Value
        End Set
    End Property

End Class
```

NegPontosProximoS.vb

```
Public Class NegPontosProximoS
        Private oCol As Collection

        Public Sub New()
            oCol = New Collection
```

```
        End Sub

    Public Sub Add(ByVal pNegPontoProximo As NegPontoProximo)
        oCol.Add(pNegPontoProximo, pNegPontoProximo.Ponto.CodPonto)
    End Sub

    Public Sub Remove(ByVal psCodPonto As String)
        oCol.Remove(psCodPonto)
    End Sub

    Public Sub Remove(ByVal piIndex As Integer)
        oCol.Remove(piIndex)
    End Sub

    Public Function Count() As Integer
        Count = oCol.Count()
    End Function

    Public Function Item(ByVal piIndex As Integer) As NegPontoProximo
        Item = oCol.Item(piIndex)
    End Function

    Public Function Item(ByVal psKey As String) As NegPontoProximo
        Item = oCol.Item(psKey)
    End Function

End Class
```

NegPosicao.vb

```
Public Class NegPosicao
    Private m_Latitude As Double
    Private m_Longitude As Double

    Private m_LatitudeAnt As Double
    Private m_LongitudeAnt As Double

    Private objNegPontosProximos As NegPontosProximoS

    Private Sub CarregarPontosProximos(Optional ByVal bTodos As Boolean =
False)
        Dim octlPonto As New ctlPonto
        Dim objPontoS As New colPonto
        Dim objNegPontoProximo As NegPontoProximo
        Dim iDistancia As Integer

        Dim iCont As Integer

        objPontoS = octlPonto.FindAllPontosProximos(Latitude, Longitude,
bTodos)

        objNegPontosProximos = New NegPontosProximoS
        For iCont = 1 To objPontoS.Count()
            iDistancia = CalculaDistancia(m_Latitude, m_Longitude,
objPontoS.Item(iCont).Latitude, objPontoS.Item(iCont).Longitude)
            objNegPontoProximo = New NegPontoProximo(objPontoS.Item(iCont),
iDistancia)
            objNegPontosProximos.Add(objNegPontoProximo)
        Next
```

```
    End Sub

    Private Sub CarregarPontosProximosWhere(Optional ByVal sWhere As String
= "")
        Dim octlPonto As New ctlPonto
        Dim objPontoS As New colPonto
        Dim objNegPontoProximo As NegPontoProximo
        Dim iDistancia As Integer

        Dim iCont As Integer

        objPontoS = octlPonto.FindAllPontosProximossWhere(Latitude,
Longitude, sWhere)

        objNegPontosProximos = New NegPontosProximoS
        For iCont = 1 To objPontoS.Count()
            iDistancia = CalculaDistancia(m_Latitude, m_Longitude,
objPontoS.Item(iCont).Latitude, objPontoS.Item(iCont).Longitude)
            objNegPontoProximo = New NegPontoProximo(objPontoS.Item(iCont),
iDistancia)
            objNegPontosProximos.Add(objNegPontoProximo)
        Next

    End Sub

    Public Sub CarregarPosicao(Optional ByVal bTodos As Boolean = False)
        m_LatitudeAnt = m_Latitude
        m_LongitudeAnt = m_Longitude
        RetornaLatLongGPS(m_Latitude, m_Longitude)
        CarregarPontosProximos(bTodos)
    End Sub

    Public Sub CarregarPosicaosWhere(Optional ByVal sWhere As String = "")
        m_LatitudeAnt = m_Latitude
        m_LongitudeAnt = m_Longitude
        RetornaLatLongGPS(m_Latitude, m_Longitude)
        CarregarPontosProximosWhere(sWhere)
    End Sub

    Public ReadOnly Property Latitude() As Double
        Get
            Latitude = m_Latitude
        End Get
    End Property

    Public ReadOnly Property Longitude() As Double
        Get
            Longitude = m_Longitude
        End Get
    End Property

    Public ReadOnly Property LatitudeAnt() As Double
        Get
            LatitudeAnt = m_LatitudeAnt
        End Get
    End Property

    Public ReadOnly Property LongitudeAnt() As Double
        Get
            LongitudeAnt = m_LongitudeAnt
        End Get
```

```
    End Property

    Public ReadOnly Property NegPontosProximos() As NegPontosProximos
        Get
            NegPontosProximos = objNegPontosProximos
        End Get
    End Property


End Class
```

frmCategoria.vb

```
Public Class frmCategoria
    Private octlCategoria As New ctlCategoria
    Private oclsCategoria As clsCategoria
    Public iEvento As Integer
    Public iCodCategoria As Integer

    Private Sub frmPonto_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        If iEvento = 1 Then
            Me.Text = "Incluir Categoria"
            txtDesc.Text = ""
            GroupBox1.Enabled = True
        ElseIf iEvento = 2 Then
            Me.Text = "Alterar Categoria"
            oclsCategoria = octlCategoria.FindByPrimaryKey(iCodCategoria)
            txtDesc.Text = oclsCategoria.Descricao
            GroupBox1.Enabled = True
        ElseIf iEvento = 3 Then
            oclsCategoria = octlCategoria.FindByPrimaryKey(iCodCategoria)
            Me.Text = "Excluir Categoria"
            txtDesc.Text = oclsCategoria.Descricao
            GroupBox1.Enabled = False
        End If
    End Sub

    Private Sub cmdOK_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdOK.Click
        If Len(Trim(txtDesc.Text)) = 0 Then
            MsgBox("Digite Corretamente a descrição")
            Exit Sub
        End If
        frmCategoriaList.bRefresh = True
        If iEvento = 1 Then
            oclsCategoria = octlCategoria.Insert(txtDesc.Text)
            If oclsCategoria.CodCategoria <= 0 Then
                MsgBox("Erro na Inclusão, Não Efetuada!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            Else
                MsgBox("Inclusão Efetuada com Sucesso!",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Atenção")
            End If
            Me.Close()
        ElseIf iEvento = 2 Then
            oclsCategoria = octlCategoria.Alterar(iCodCategoria,
txtDesc.Text)
            If oclsCategoria.CodCategoria = iCodCategoria Then
```

```
                MsgBox("Alteração Efetuada com Sucesso!",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Atenção")
            Else
                MsgBox("Erro na Alteração, Não Efetuada!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            End If
            Me.Close()
        ElseIf iEvento = 3 Then
            Call octlCategoria.Excluir(iCodCategoria)
            MsgBox("Exclusão Efetuada com Sucesso!",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Atenção")
            Me.Close()
            End If
    End Sub

    Private Sub cmdCancelar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdCancelar.Click
        frmCategoriaList.bRefresh = False
        Me.Close()
    End Sub
End Class
```

frmCategoriaList.vb

```
Public Class frmCategoriaList

    Dim octlCategoria As New ctlCategoria
    Public bRefresh As Boolean

    Private Sub PontoList_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        CarregaList()
    End Sub

    Private Sub cmdSair_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdSair.Click
        Me.Close()
    End Sub

    Private Sub cmpOpcoes_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmpOpcoes.Click
        Me.MenuPopUp.Show(lstPontos, Me.cmpOpcoes.Location.X,
Me.cmpOpcoes.Location.Y - 90)
    End Sub

    Private Sub CarregaList()
        Dim iCont As Integer
        Dim lstItem As ListViewItem
        Dim ocolCategoria As colCategoria

        ocolCategoria = octlCategoria.FindAll()
        lstPontos.Items.Clear()
        For iCont = 1 To ocolCategoria.Count
            lstItem =
lstPontos.Items.Add(ocolCategoria.Item(iCont).CodCategoria)
            lstItem.SubItems.Add(ocolCategoria.Item(iCont).Descricao)
        Next
        lstPontos.Refresh()
    End Sub
```

```
    Private Sub lstPontos_MouseDoubleClick(ByVal sender As Object, ByVal e
As System.Windows.Forms.MouseEventArgs) Handles lstPontos.MouseDoubleClick
        mnuAlterar_Click(Me, e)
    End Sub

    Private Sub lstPontos_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles lstPontos.MouseDown
        If Me.lstPontos.MouseButtons = Windows.Forms.MouseButtons.Right
Then
            Me.MenuPopUp.Show(lstPontos,
Me.lstPontos.PointToClient(Cursor.Position))
        End If
    End Sub

    Private Sub mnuIncluir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuIncluir.Click
        frmCategoria.iEvento = 1
        frmCategoria.iCodCategoria = 0
        bRefresh = False
        frmCategoria.ShowDialog(Me)
        If bRefresh = True Then CarregaList()
    End Sub

    Private Sub mnuAlterar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuAlterar.Click
        Dim iCodCat As Integer
        Try
            iCodCat = lstPontos.SelectedItems(0).Text
        Catch ex As Exception
            MsgBox("Escolha Corretamente a Categoria para Alterar!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            Exit Sub
        End Try
        frmCategoria.iEvento = 2
        frmCategoria.iCodCategoria = iCodCat
        bRefresh = False
        frmCategoria.ShowDialog(Me)
        If bRefresh = True Then CarregaList()
    End Sub

    Private Sub mnuExcluir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuExcluir.Click
        Dim iCodCat As Integer
        Try
            iCodCat = lstPontos.SelectedItems(0).Text
        Catch ex As Exception
            MsgBox("Escolha Corretamente a Categoria para Excluir!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            Exit Sub
        End Try
        frmCategoria.iEvento = 3
        frmCategoria.iCodCategoria = iCodCat
        bRefresh = False
        frmCategoria.ShowDialog(Me)
        If bRefresh = True Then CarregaList()
    End Sub
End Class
```

frmPesquisa.vb

```
Public Class frmPesquisa
    Private oNeg As New NegPosicao
    Private iCodPos As Integer
    Private bSai As Boolean

    Private Sub TextBox1_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox1.KeyPress
        If Asc(e.KeyChar) = Keys.Enter Then
            If bSai = False Then
                CarregaPontos()
            Else
                Me.Close()
            End If
        ElseIf Asc(e.KeyChar) = Keys.Escape Then
            iCodPontoBusca = 0
            Me.Close()
        End If
    End Sub

    Private Sub TextBox1_LostFocus(ByVal sender As Object, ByVal e As
System.EventArgs) Handles TextBox1.LostFocus
        Me.Focus()
    End Sub

    Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TextBox1.TextChanged
        bSai = False
        Falar(TextBox1.Text, True)
    End Sub

    Private Sub CarregaPontos()
        Dim strFilter As String
        If Len(Trim(TextBox1.Text)) = 0 Then
            iCodPos = 0
            Exit Sub
        End If
        strFilter = "DESC_PONTO LIKE '" & TextBox1.Text & "%'"
        oNeg.CarregarPosicaosWhere(strFilter)
        If oNeg.NegPontosProximos.Count > 0 Then
            iCodPos = 1
            iCodPontoBusca =
oNeg.NegPontosProximos.Item(iCodPos).Ponto.CodPonto
            Falar(oNeg.NegPontosProximos.Item(iCodPos).Ponto.Descricao,
True)
            bSai = True
        Else
            iCodPontoBusca = 0
            iCodPos = 0
        End If
    End Sub

    Private Sub frmPesquisa_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        TextBox1.Text = ""
        iCodPontoBusca = 0
        Falar("Digite as inicias do ponto e ENTER para buscar", True)
    End Sub

    Private Sub F1ToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles F1ToolStripMenuItem.Click
        If iCodPos > 0 Then
```

```
            Falar(oNeg.NegPontosProximos.Item(iCodPos).Ponto.Descricao,
True)
        End If
    End Sub

    Private Sub F2ToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles F2ToolStripMenuItem.Click
        If iCodPos > 0 Then
            iCodPos = iCodPos + 1
            If iCodPos > oNeg.NegPontosProximos.Count Then
                iCodPos = 1
            End If
            iCodPontoBusca =
oNeg.NegPontosProximos.Item(iCodPos).Ponto.CodPonto
            Falar(oNeg.NegPontosProximos.Item(iCodPos).Ponto.Descricao,
True)
            bSai = True
        End If
    End Sub

    Private Sub F3ToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles F3ToolStripMenuItem.Click
        If iCodPos > 0 Then
            iCodPos = iCodPos - 1
            If iCodPos < 1 Then
                iCodPos = oNeg.NegPontosProximos.Count
            End If
            iCodPontoBusca =
oNeg.NegPontosProximos.Item(iCodPos).Ponto.CodPonto
            Falar(oNeg.NegPontosProximos.Item(iCodPos).Ponto.Descricao,
True)
            bSai = True
        End If
    End Sub
End Class
```

frmPonto.vb

```
Public Class frmPonto
    Private octlCategoria As New ctlCategoria
    Private octlPonto As New ctlPonto
    Private oclsPonto As clsPonto
    Public iEvento As Integer
    Public iCodPonto As Integer

    Private m_Lat As Double
    Private m_Lon As Double

    Private Sub frmPonto_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        CarregaCombo()
        If iEvento = 1 Then
            Me.Text = "Incluir Ponto"
            m_Lat = tLat
            m_Lon = tLon
            txtDesc.Text = ""
            GroupBox1.Enabled = True
        ElseIf iEvento = 2 Then
            Me.Text = "Alterar Ponto"
            oclsPonto = octlPonto.FindByPrimaryKey(iCodPonto)
```

```
            m_Lat = oclsPonto.Latitude
            m_Lon = oclsPonto.Longitude
            txtDesc.Text = oclsPonto.Descricao
            GroupBox1.Enabled = True
            cmbCategoria.Text = oclsPonto.Categoria.Descricao
        ElseIf iEvento = 3 Then
            oclsPonto = octlPonto.FindByPrimaryKey(iCodPonto)
            Me.Text = "Excluir Ponto"
            m_Lat = oclsPonto.Latitude
            m_Lon = oclsPonto.Longitude
            txtDesc.Text = oclsPonto.Descricao
            GroupBox1.Enabled = False
            cmbCategoria.Text = oclsPonto.Categoria.Descricao
        End If
        lblLatitude.Text = m_Lat
        lblLongitude.Text = m_Lon
    End Sub

    Private Sub CarregaCombo()
        Dim ocolCategoria As DataTable
        'Dim iCont As Integer
        ocolCategoria = octlCategoria.FindAllDataSource()
        cmbCategoria.DataSource = ocolCategoria
        cmbCategoria.DisplayMember = "DESC_CAT"
        cmbCategoria.ValueMember = "COD_CAT"
        cmbCategoria.SelectedIndex = 0
    End Sub

    Private Sub cmdOK_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdOK.Click
        If Len(Trim(txtDesc.Text)) = 0 Then
            MsgBox("Digite Corretamente a descrição")
            Exit Sub
        End If
        frmPontoList.bRefresh = True
        If iEvento = 1 Then
            oclsPonto = octlPonto.Insert(cmbCategoria.SelectedValue,
txtDesc.Text, m_Lat, m_Lon, 1, "")
            If oclsPonto.CodPonto <= 0 Then
                MsgBox("Erro na Inclusão, Não Efetuada!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            Else
                MsgBox("Inclusão Efetuada com Sucesso!",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Atenção")
                Me.Close()
            End If
        ElseIf iEvento = 2 Then
            oclsPonto = octlPonto.Alterar(iCodPonto,
cmbCategoria.SelectedValue, txtDesc.Text)
            If oclsPonto.CodPonto = iCodPonto Then
                MsgBox("Alteração Efetuada com Sucesso!",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Atenção")
            Else
                MsgBox("Erro na Alteração, Não Efetuada!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            End If
            Me.Close()
        ElseIf iEvento = 3 Then
            Call octlPonto.Excluir(iCodPonto)
            MsgBox("Exclusão Efetuada com Sucesso!",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Atenção")
```

```
            Me.Close()
            End If
    End Sub

    Private Sub cmdCancelar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdCancelar.Click
        frmPontoList.bRefresh = False
        Me.Close()
    End Sub
End Class
```

frmPontoList.vb

```
Public Class frmPontoList
    Dim octlPontos As New ctlPonto
    Public bRefresh As Boolean

    Private Sub PontoList_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        CarregaCombo()
        CarregaList()
    End Sub

    Private Sub cmdSair_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdSair.Click
        Me.Close()
    End Sub

    Private Sub cmpOpcoes_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmpOpcoes.Click
        Me.MenuPopUp.Show(lstPontos, Me.cmpOpcoes.Location.X,
Me.cmpOpcoes.Location.Y - 90)
    End Sub

    Private Sub cmdPesq_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdPesq.Click
        Dim sWhere As String
        If Len(Trim(txtDesc.Text)) = 0 And ckqTodas.Checked = False Then
            sWhere = "COD_CAT = " & cmbCategoria.SelectedValue
        ElseIf Len(Trim(txtDesc.Text)) > 0 And ckqTodas.Checked = True Then
            sWhere = "DESC_PONTO LIKE '%" & Trim(txtDesc.Text) & "%'"
        ElseIf Len(Trim(txtDesc.Text)) > 0 And ckqTodas.Checked = False
Then
            sWhere = "COD_CAT = " & cmbCategoria.SelectedValue & " AND
DESC_PONTO LIKE '%" & Trim(txtDesc.Text) & "%'"
        Else
            sWhere = ""
        End If
        CarregaList(sWhere)
    End Sub

    Private Sub CarregaList(Optional ByVal sWhere As String = "")
        Dim iCont As Integer
        Dim lstItem As ListViewItem
        oPosicao.CarregarPosicaosWhere(sWhere)
        lstPontos.Items.Clear()
        For iCont = 1 To oPosicao.NegPontosProximos.Count
            lstItem =
lstPontos.Items.Add(oPosicao.NegPontosProximos.Item(iCont).Ponto.CodPonto)
```

```
lstItem.SubItems.Add(oPosicao.NegPontosProximos.Item(iCont).Ponto.Descricao
)

lstItem.SubItems.Add(oPosicao.NegPontosProximos.Item(iCont).Ponto.Categoria
.Descricao)

lstItem.SubItems.Add(oPosicao.NegPontosProximos.Item(iCont).Ponto.Latitude)

lstItem.SubItems.Add(oPosicao.NegPontosProximos.Item(iCont).Ponto.Longitude
)

lstItem.SubItems.Add(oPosicao.NegPontosProximos.Item(iCont).Distancia)
        Next
        lstPontos.Refresh()
    End Sub

    Private Sub CarregaCombo()
        Dim octlCategoria As New ctlCategoria
        Dim ocolCategoria As DataTable
        'Dim iCont As Integer
        ocolCategoria = octlCategoria.FindAllDataSource()

        cmbCategoria.DataSource = ocolCategoria
        cmbCategoria.DisplayMember = "DESC_CAT"
        cmbCategoria.ValueMember = "COD_CAT"

        'cmbCategoria.Items.Add("TODOS")
        cmbCategoria.Text = ("")
    End Sub

    Private Sub ckqTodas_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ckqTodas.CheckedChanged
        If ckqTodas.Checked = True Then
            cmbCategoria.Enabled = False
        Else
            cmbCategoria.Enabled = True
        End If
    End Sub

    Private Sub lstPontos_MouseDoubleClick(ByVal sender As Object, ByVal e
As System.Windows.Forms.MouseEventArgs) Handles lstPontos.MouseDoubleClick
        mnuAlterar_Click(Me, e)
    End Sub

    Private Sub lstPontos_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles lstPontos.MouseDown
        If Me.lstPontos.MouseButtons  = Windows.Forms.MouseButtons.Right
Then
            Me.MenuPopUp.Show(lstPontos,
Me.lstPontos.PointToClient(Cursor.Position))
        End If
    End Sub

    Private Sub mnuIncluir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuIncluir.Click
        frmPonto.iEvento = 1
        frmPonto.iCodPonto = 0
        bRefresh = False
        frmPonto.ShowDialog(Me)
        If bRefresh = True Then cmdPesq_Click(Me, e)
```

```
    End Sub

    Private Sub mnuAlterar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuAlterar.Click
        Dim iCodPonto As Integer
        Try
            iCodPonto = lstPontos.SelectedItems(0).Text
        Catch ex As Exception
            MsgBox("Escolha Corretamente a Categoria para Alterar!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            Exit Sub
        End Try
        frmPonto.iEvento = 2
        frmPonto.iCodPonto = iCodPonto
        bRefresh = False
        frmPonto.ShowDialog(Me)
        If bRefresh = True Then cmdPesq_Click(Me, e)
    End Sub

    Private Sub mnuExcluir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuExcluir.Click
        Dim iCodPonto As Integer
        Try
            iCodPonto = lstPontos.SelectedItems(0).Text
        Catch ex As Exception
            MsgBox("Escolha Corretamente a Categoria para Excluir!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Atenção")
            Exit Sub
        End Try
        frmPonto.iEvento = 3
        frmPonto.iCodPonto = iCodPonto
        bRefresh = False
        frmPonto.ShowDialog(Me)
        If bRefresh = True Then cmdPesq_Click(Me, e)
    End Sub
End Class
```

frmPrincipal.vb

```
Public Class frmPrincipal
    Inherits System.Windows.Forms.Form
    Private strTexto As String
    Private Nivel1 As Integer
    Private Nivel2 As Integer
    Friend WithEvents TrackBar1 As System.Windows.Forms.TrackBar
    Friend WithEvents lblx As System.Windows.Forms.Label
    Friend WithEvents TrackBar2 As System.Windows.Forms.TrackBar
    Friend WithEvents Panel1 As System.Windows.Forms.Panel
    Friend WithEvents lblLatitude As System.Windows.Forms.Label
    Friend WithEvents lblLongitude As System.Windows.Forms.Label

    Private dbltempLatitude As Double
    Private dbltempLongitude As Double
    Friend WithEvents lblxAnt As System.Windows.Forms.Label
    Friend WithEvents lblPonto As System.Windows.Forms.Label
    Friend WithEvents mnuAdm As System.Windows.Forms.MenuItem
    Friend WithEvents mnuCategoria As System.Windows.Forms.MenuItem
    Friend WithEvents mnuPontos As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents lblLonAnt As System.Windows.Forms.Label
    Friend WithEvents lblLatAnt As System.Windows.Forms.Label


    Private PontoNavegacao As Integer

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Timer1 As System.Windows.Forms.Timer
    Friend WithEvents F1 As System.Windows.Forms.MenuItem
    Friend WithEvents F2 As System.Windows.Forms.MenuItem
    Friend WithEvents F3 As System.Windows.Forms.MenuItem
    Friend WithEvents F4 As System.Windows.Forms.MenuItem
    Friend WithEvents F5 As System.Windows.Forms.MenuItem
    Friend WithEvents F6 As System.Windows.Forms.MenuItem
    Friend WithEvents F7 As System.Windows.Forms.MenuItem
    Friend WithEvents F8 As System.Windows.Forms.MenuItem
    Public WithEvents GARMIN As AxMSCommLib.AxMSComm
    Friend WithEvents MenuItem As System.Windows.Forms.MainMenu
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.components = New System.ComponentModel.Container
        Dim resources As System.ComponentModel.ComponentResourceManager =
New System.ComponentModel.ComponentResourceManager(GetType(frmPrincipal))
        Me.Timer1 = New System.Windows.Forms.Timer(Me.components)
        Me.MenuItem = New System.Windows.Forms.MainMenu(Me.components)
        Me.F1 = New System.Windows.Forms.MenuItem
        Me.F2 = New System.Windows.Forms.MenuItem
        Me.F3 = New System.Windows.Forms.MenuItem
        Me.F4 = New System.Windows.Forms.MenuItem
        Me.F5 = New System.Windows.Forms.MenuItem
        Me.F6 = New System.Windows.Forms.MenuItem
        Me.F7 = New System.Windows.Forms.MenuItem
        Me.F8 = New System.Windows.Forms.MenuItem
        Me.mnuAdm = New System.Windows.Forms.MenuItem
```

```
        Me.mnuCategoria = New System.Windows.Forms.MenuItem
        Me.mnuPontos = New System.Windows.Forms.MenuItem
        Me.MenuItem1 = New System.Windows.Forms.MenuItem
        Me.GARMIN = New AxMSCommLib.AxMSComm
        Me.TrackBar1 = New System.Windows.Forms.TrackBar
        Me.lblx = New System.Windows.Forms.Label
        Me.TrackBar2 = New System.Windows.Forms.TrackBar
        Me.Panel1 = New System.Windows.Forms.Panel
        Me.lblxAnt = New System.Windows.Forms.Label
        Me.lblPonto = New System.Windows.Forms.Label
        Me.lblLatitude = New System.Windows.Forms.Label
        Me.lblLongitude = New System.Windows.Forms.Label
        Me.lblLonAnt = New System.Windows.Forms.Label
        Me.lblLatAnt = New System.Windows.Forms.Label
        CType(Me.GARMIN,
System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.TrackBar1,
System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.TrackBar2,
System.ComponentModel.ISupportInitialize).BeginInit()
        Me.Panel1.SuspendLayout()
        Me.SuspendLayout()
        '
        'Timer1
        '
        Me.Timer1.Interval = 10000
        '
        'MenuItem
        '
        Me.MenuItem.MenuItems.AddRange(New System.Windows.Forms.MenuItem()
{Me.F1, Me.F2, Me.F3, Me.F4, Me.F5, Me.F6, Me.F7, Me.F8, Me.mnuAdm})
        '
        'F1
        '
        Me.F1.Index = 0
        Me.F1.Shortcut = System.Windows.Forms.Shortcut.F1
        Me.F1.Text = "<F1>"
        '
        'F2
        '
        Me.F2.Index = 1
        Me.F2.Shortcut = System.Windows.Forms.Shortcut.F2
        Me.F2.Text = "<F2>"
        '
        'F3
        '
        Me.F3.Index = 2
        Me.F3.Shortcut = System.Windows.Forms.Shortcut.F3
        Me.F3.Text = "<F3>"
        '
        'F4
        '
        Me.F4.Index = 3
        Me.F4.Shortcut = System.Windows.Forms.Shortcut.F4
        Me.F4.Text = "<F4>"
        '
        'F5
        '
        Me.F5.Index = 4
        Me.F5.Shortcut = System.Windows.Forms.Shortcut.F5
        Me.F5.Text = "<F5>"
```

```
        '
        'F6
        '
        Me.F6.Index = 5
        Me.F6.Shortcut = System.Windows.Forms.Shortcut.F6
        Me.F6.Text = "<F6>"
        '
        'F7
        '
        Me.F7.Index = 6
        Me.F7.Shortcut = System.Windows.Forms.Shortcut.F7
        Me.F7.Text = "<F7>"
        '
        'F8
        '
        Me.F8.Index = 7
        Me.F8.Shortcut = System.Windows.Forms.Shortcut.F8
        Me.F8.Text = "<F8>"
        '
        'mnuAdm
        '
        Me.mnuAdm.Index = 8
        Me.mnuAdm.MenuItems.AddRange(New System.Windows.Forms.MenuItem()
{Me.mnuCategoria, Me.mnuPontos, Me.MenuItem1})
        Me.mnuAdm.Text = "Administração"
        '
        'mnuCategoria
        '
        Me.mnuCategoria.Index = 0
        Me.mnuCategoria.Text = "Categoria"
        '
        'mnuPontos
        '
        Me.mnuPontos.Index = 1
        Me.mnuPontos.Text = "Pontos"
        '
        'MenuItem1
        '
        Me.MenuItem1.Index = 2
        Me.MenuItem1.Text = "Pesquisa"
        '
        'GARMIN
        '
        Me.GARMIN.Enabled = True
        Me.GARMIN.Location = New System.Drawing.Point(415, 1)
        Me.GARMIN.Name = "GARMIN"
        Me.GARMIN.OcxState = CType(resources.GetObject("GARMIN.OcxState"),
System.Windows.Forms.AxHost.State)
        Me.GARMIN.Size = New System.Drawing.Size(38, 38)
        Me.GARMIN.TabIndex = 4
        '
        'TrackBar1
        '
        Me.TrackBar1.LargeChange = 1
        Me.TrackBar1.Location = New System.Drawing.Point(330, 1)
        Me.TrackBar1.Maximum = 1000
        Me.TrackBar1.Name = "TrackBar1"
        Me.TrackBar1.Orientation =
System.Windows.Forms.Orientation.Vertical
        Me.TrackBar1.Size = New System.Drawing.Size(42, 341)
        Me.TrackBar1.TabIndex = 5
```

```
Me.TrackBar1.TickStyle = System.Windows.Forms.TickStyle.TopLeft
Me.TrackBar1.Value = 1000
'
'lblx
'
Me.lblx.AutoSize = True
Me.lblx.Location = New System.Drawing.Point(300, 0)
Me.lblx.Name = "lblx"
Me.lblx.Size = New System.Drawing.Size(12, 13)
Me.lblx.TabIndex = 6
Me.lblx.Text = "x"
'
'TrackBar2
'
Me.TrackBar2.Location = New System.Drawing.Point(-1, 328)
Me.TrackBar2.Maximum = 1000
Me.TrackBar2.Name = "TrackBar2"
Me.TrackBar2.Size = New System.Drawing.Size(340, 42)
Me.TrackBar2.TabIndex = 7
Me.TrackBar2.TickStyle = System.Windows.Forms.TickStyle.TopLeft
Me.TrackBar2.Value = 1000
'
'Panel1
'
Me.Panel1.Controls.Add(Me.lblx)
Me.Panel1.Controls.Add(Me.lblxAnt)
Me.Panel1.Controls.Add(Me.lblPonto)
Me.Panel1.ImeMode = System.Windows.Forms.ImeMode.NoControl
Me.Panel1.Location = New System.Drawing.Point(12, 12)
Me.Panel1.Name = "Panel1"
Me.Panel1.Size = New System.Drawing.Size(314, 317)
Me.Panel1.TabIndex = 8
'
'lblxAnt
'
Me.lblxAnt.AutoSize = True
Me.lblxAnt.ForeColor = System.Drawing.SystemColors.Window
Me.lblxAnt.Location = New System.Drawing.Point(300, 0)
Me.lblxAnt.Name = "lblxAnt"
Me.lblxAnt.Size = New System.Drawing.Size(12, 13)
Me.lblxAnt.TabIndex = 7
Me.lblxAnt.Text = "x"
'
'lblPonto
'
Me.lblPonto.AutoSize = True
Me.lblPonto.Location = New System.Drawing.Point(300, 0)
Me.lblPonto.Name = "lblPonto"
Me.lblPonto.Size = New System.Drawing.Size(13, 13)
Me.lblPonto.TabIndex = 8
Me.lblPonto.Text = "o"
'
'lblLatitude
'
Me.lblLatitude.AutoSize = True
Me.lblLatitude.Location = New System.Drawing.Point(9, 373)
Me.lblLatitude.Name = "lblLatitude"
Me.lblLatitude.Size = New System.Drawing.Size(48, 13)
Me.lblLatitude.TabIndex = 9
Me.lblLatitude.Text = "Latitude:"
'
```

```
        'lblLongitude
        '
        Me.lblLongitude.AutoSize = True
        Me.lblLongitude.Location = New System.Drawing.Point(9, 390)
        Me.lblLongitude.Name = "lblLongitude"
        Me.lblLongitude.Size = New System.Drawing.Size(57, 13)
        Me.lblLongitude.TabIndex = 10
        Me.lblLongitude.Text = "Longitude:"
        '
        'lblLonAnt
        '
        Me.lblLonAnt.AutoSize = True
        Me.lblLonAnt.Location = New System.Drawing.Point(188, 390)
        Me.lblLonAnt.Name = "lblLonAnt"
        Me.lblLonAnt.Size = New System.Drawing.Size(73, 13)
        Me.lblLonAnt.TabIndex = 12
        Me.lblLonAnt.Text = "LongitudeAnt:"
        '
        'lblLatAnt
        '
        Me.lblLatAnt.AutoSize = True
        Me.lblLatAnt.Location = New System.Drawing.Point(188, 373)
        Me.lblLatAnt.Name = "lblLatAnt"
        Me.lblLatAnt.Size = New System.Drawing.Size(64, 13)
        Me.lblLatAnt.TabIndex = 11
        Me.lblLatAnt.Text = "LatitudeAnt:"
        '
        'frmPrincipal
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(465, -2)
        Me.Controls.Add(Me.lblLonAnt)
        Me.Controls.Add(Me.lblLatAnt)
        Me.Controls.Add(Me.lblLongitude)
        Me.Controls.Add(Me.lblLatitude)
        Me.Controls.Add(Me.Panel1)
        Me.Controls.Add(Me.TrackBar2)
        Me.Controls.Add(Me.TrackBar1)
        Me.Controls.Add(Me.GARMIN)
        Me.MaximizeBox = False
        Me.Menu = Me.MenuItem
        Me.Name = "frmPrincipal"
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Speak Position"
        Me.WindowState = System.Windows.Forms.FormWindowState.Maximized
        CType(Me.GARMIN,
System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.TrackBar1,
System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.TrackBar2,
System.ComponentModel.ISupportInitialize).EndInit()
        Me.Panel1.ResumeLayout(False)
        Me.Panel1.PerformLayout()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub

#End Region
```

```
    Private Sub frmPrincipal_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        objDB = New DBConection
        oPosicao = New NegPosicao
        tLat = -15
        tLon = -45
        iNumeroVoz = 2
        ResetFala()
        Nivel1 = 0
        Falar("Aperte F8 para Ajuda", True)
    End Sub

    Private Sub F1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles F1.Click
        If Nivel1 = 0 Then
            Nivel1 = 1
            Capturar()
            oPosicao.CarregarPosicao(False)
            PontoNavegacao = 1
            strTexto = "Você está a " &
oPosicao.NegPontosProximos.Item(PontoNavegacao).Distancia() & " metros do "
& oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Descricao()
            strTexto = strTexto & vbCrLf & ". Aperte F1 para Repetir, F2
para Proximo Ponto, F3 para Ponto Anterior, F4 para definir Rota, F5 voltar
ao menu principal e F8 para Ajuda"
            Falar(strTexto, True)
            PosicionarQuadro(oPosicao.Latitude, _
                             oPosicao.Longitude, lblx)

PosicionarQuadro(oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Lati
tude, _

oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Longitude, lblPonto)
        ElseIf Nivel1 = 1 Then
            strTexto = "A " &
oPosicao.NegPontosProximos.Item(PontoNavegacao).Distancia() & " metros do "
& oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Descricao()
            Falar(strTexto, True)

PosicionarQuadro(oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Lati
tude, _

oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Longitude, lblPonto)
        ElseIf Nivel1 = 2 Then
            Falar(strAnguloDeslocamento, True)
        ElseIf Nivel1 = 7 Then
            iNumeroVoz = 1
            ResetFala()
            Falar("A Voz agora é a da Maria", True)
            Nivel1 = 0
            'F8_Click(Me, e)
        End If
    End Sub

    Private Sub F2_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles F2.Click
        If Nivel1 = 0 Then
            frmPesquisa.ShowDialog(Me)
            If iCodPontoBusca > 0 Then
                Nivel1 = 1
                Capturar()
```

```
                oPosicao.CarregarPosicaosWhere("COD_PONTO = " &
iCodPontoBusca)
                PontoNavegacao = 1
                strTexto = "Você está a " &
oPosicao.NegPontosProximos.Item(PontoNavegacao).Distancia() & " metros do "
& oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Descricao()
                strTexto = strTexto & vbCrLf & ". Aperte F1 para Repetir,
F4 para definir Rota, F5 voltar ao menu principal e F8 para Ajuda"
                Falar(strTexto, True)
            Else
                Falar("busca cancelada", True)
            End If
        ElseIf Nivel1 = 1 Then
            PontoNavegacao = PontoNavegacao + 1
            If PontoNavegacao > oPosicao.NegPontosProximos.Count Then
                PontoNavegacao = 1
            End If
            strTexto = "A " &
oPosicao.NegPontosProximos.Item(PontoNavegacao).Distancia() & " metros do "
& oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Descricao()
            Falar(strTexto, True)

PosicionarQuadro(oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Lati
tude, _

oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Longitude, lblPonto)
        ElseIf Nivel1 = 2 Then
            Falar(iDeslocamento & " metros do " &
oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Descricao(), True)
        ElseIf Nivel1 = 7 Then
            iNumeroVoz = 2
            ResetFala()
            Falar("A Voz agora é a do João", True)
            Nivel1 = 0
            'F8_Click(Me, e)
        End If
    End Sub

    Private Sub F3_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles F3.Click
        If Nivel1 = 0 Then

        ElseIf Nivel1 = 1 Then
            PontoNavegacao = PontoNavegacao - 1
            If PontoNavegacao = 0 Then
                PontoNavegacao = oPosicao.NegPontosProximos.Count
            End If
            strTexto = "A " &
oPosicao.NegPontosProximos.Item(PontoNavegacao).Distancia() & " metros do "
& oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Descricao()
            Falar(strTexto, True)
        ElseIf Nivel1 = 2 Then
            Falar(strVelocidade, True)
        ElseIf Nivel1 = 7 Then

        End If
    End Sub

    Private Sub F4_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles F4.Click
        If Nivel1 = 0 Then
```

```
        ElseIf Nivel1 = 1 Then
            Nivel1 = 2
            Timer1.Enabled = True
        ElseIf Nivel1 = 2 Then
            Falar(strTempoChegada, True)
        End If
    End Sub

    Private Sub F5_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles F5.Click
        If Nivel1 = 1 Or Nivel1 = 2 Then
            Nivel1 = 0
            F8_Click(Me, e)
        End If
    End Sub

    Private Sub F7_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles F7.Click
        If Nivel1 = 0 Then
            Nivel1 = 7
            Falar("Aperte F1 para a Maria falar, F2 para o João falar, F5
voltar ao menu principal e F8 para Repetir", True)
        End If
    End Sub

    Private Sub F8_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles F8.Click
        Select Case Nivel1
            Case 0
                Falar("Aperte F1 para Localização, F2 para Buscar Ponto, F7
para mudar Voz e F8 para Repetir", True)
            Case 1
                Falar("Aperte F1 para Falar Ponto, F2 para Proximo Ponto,
F3 para Ponto Anterior, F4 para definir Rota, F5 voltar ao menu principal e
F8 para Repetir", True)
            Case 2
                Falar("Aperte F1 para Falar Ângulo, F2 para Distância, F3
para Velocidade, F4 para Tempo de Chegada, F5 voltar ao menu principal e F8
para Repetir", True)
            Case 7
                Falar("F1 para a Maria falar, F2 para o João falar, F5
voltar ao menu principal e F8 para Repetir", True)
        End Select
    End Sub

    Private Sub TrackBar1_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TrackBar1.Scroll
        Dim x As Double
        Dim y As Double
        x = -15 - (-(TrackBar1.Value - 1000)) * 0.001
        y = -47 - (-(TrackBar2.Value - 1000)) * 0.001
        lblLatitude.Text = "Lat: " & CStr(x)
        PosicionarQuadro(x, y, lblx)
    End Sub

    Private Sub TrackBar2_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TrackBar2.Scroll
        Dim x As Double
        Dim y As Double
        x = -15 - (-(TrackBar1.Value - 1000)) * 0.001
```

```
        y = -47 - (-(TrackBar2.Value - 1000)) * 0.001
        lblLongitude.Text = "Lon: " & CStr(y)
        PosicionarQuadro(x, y, lblx)
    End Sub

    Private Sub PosicionarQuadro(ByVal Lat As Double, ByVal Lon As Double,
ByVal olabel As Label)
        Dim a As System.Drawing.Point
        a = New System.Drawing.Point
        a.Y = (-(Lat - 1) - 16) * 300
        a.X = (Lon + 48) * 300
        olabel.Location = a
    End Sub

    Private Sub Capturar()
        Desconectar()
        Conectar()
        Request_Pos()

        'tLatAnt = tLat
        'tLonAnt = tLon
        'lblLatAnt.Text = "LatAnt: " & tLatAnt
        'lblLonAnt.Text = "LonAnt: " & tLonAnt

        'tLat = -15 - (-(TrackBar1.Value - 1000)) * 0.001
        'tLon = -47 - (-(TrackBar2.Value - 1000)) * 0.001

        'PosicionarQuadro(tLat, tLon, lblx)
        'PosicionarQuadro(tLatAnt, tLonAnt, lblxAnt)
    End Sub

    Private Sub Conectar()
        Dim NewPort As Object
        Dim OldPort As Short
        Dim ReOpen As Boolean

        On Error Resume Next

        OldPort = Me.GARMIN.CommPort
        'Escolha de Serial
        NewPort = 6

        If NewPort <> OldPort Then
            If Me.GARMIN.PortOpen Then
                Me.GARMIN.PortOpen = False
                ReOpen = True
            End If
            Me.GARMIN.CommPort = NewPort
            If Err.Number = 0 Then
                If ReOpen Then
                    Me.GARMIN.PortOpen = True
                End If
            End If
            If Err.Number Then
                MsgBox(ErrorToString(), 48)
                Me.GARMIN.CommPort = OldPort
                Exit Sub
            End If
        End If

        On Error Resume Next
```

```
        If Not GARMIN.PortOpen Then
            GARMIN.PortOpen = True
            If Err.Number Then Exit Sub
        End If
    End Sub

    Private Sub Desconectar()
        If GARMIN.PortOpen Then Me.GARMIN.PortOpen = False
    End Sub

    Private Sub GARMIN_OnComm(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles GARMIN.OnComm
        Dim Buffer() As Byte
        Dim i As Short
        Dim toto As String
        Dim ret As Integer
        Dim nZoneCorrection As Integer
        Dim Clock As Date
        Dim packet() As Byte

        On Error Resume Next
        Err.Clear()
        If Me.GARMIN.CommEvent = MSCommLib.OnCommConstants.comEvReceive
Then
            Buffer = GARMIN.Input
            packet = Buffer
            If UBound(packet) = 0 Or UBound(packet) = -1 Then Exit Sub
            If packet(1) = Pid_Ack_Byte Then
                If packet(3) = Pid_Command_Data Then
                    If packet(9) = Pid_Position_Data Then
                        tLat = Calcul_Position(packet, 12)
                        tLon = Calcul_Position(packet, 20)
                        oPosicao.CarregarPosicao(False)
                        PosicionarQuadro(oPosicao.Latitude, _
                                         oPosicao.Longitude, lblx)
                    End If
                End If
            End If
        End If
    End Sub

    Private Sub mnuCategoria_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles mnuCategoria.Click
        frmCategoriaList.ShowDialog(Me)
    End Sub

    Private Sub mnuPontos_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuPontos.Click
        frmPontoList.ShowDialog(Me)
    End Sub

    Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem1.Click
        frmPesquisa.ShowDialog(Me)
    End Sub

    Private Sub MontaFalas()
        Dim iTempo As Integer
        Dim iSegundos As Integer
        Dim iMinutos As Integer
```

```
        Dim iHoras As Integer

        strAnguloDeslocamento = CalculaAngulo(tLat, _
              tLon, _
              tLatAnt, _
              tLonAnt, _

oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Latitude, _

oPosicao.NegPontosProximos.Item(PontoNavegacao).Ponto.Longitude)
        iDeslocamento = CalculaDistancia(tLat, _
                                         tLon, _
                                         tLatAnt, _
                                         tLonAnt)
        strVelocidade = Int(iDeslocamento / 10) & " metros por segundo"

        iTempo = CalculaTempoChegada(tLat, _
                           tLon, _
                           tLatAnt, _
                           tLonAnt, _
                           Int(iDeslocamento / 10))

        If iTempo <= 60 Then
            strTempoChegada = iTempo & " segundo"
        ElseIf iTempo > 60 And iTempo <= 3600 Then
            iMinutos = iTempo \ 60
            iSegundos = iTempo Mod 60
            strTempoChegada = iMinutos & " minutos e " & iSegundos & "
segundos"
        ElseIf iTempo > 3600 Then
            iHoras = iTempo \ 3600
            iMinutos = iTempo Mod 3600
            strTempoChegada = iHoras & " horas e " & iMinutos & " minutos"
        End If

    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        Capturar()
        MontaFalas()
        VerificarSeFala()
    End Sub

    Private Sub VerificarSeFala()
        If iDeslocamento > 10 Then
            Falar(strAnguloDeslocamento, True)
        End If
    End Sub
End Class
```