MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14001-PRE/9176

PROPOSTA DE ARQUITETURA PARA IMPLEMENTAÇÃO DE SOLUÇÕES DE NAVEGAÇÃO VIA GPS EM TEMPO REAL

Rafael Augusto Galo Fernandes*

* FEG-UNESP, bolsista CNPq/MCT

Relatório Final de Projeto de Iniciação Científica (PIBIC/INPE-CNPq/MCT), orientado pelo Dr. Hélio Koiti Kuga e co-orientado pelo Dr. Rodolpho Vilhena de Moraes.

INPE
São José dos Campos
2006

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

PROPOSTA DE ARQUITETURA PARA IMPLEMENTAÇÃO DE SOLUÇÕES DE NAVEGAÇÃO VIA GPS EM TEMPO REAL

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA

(PIBIC/CNPq/INPE)

Rafael Augusto Galo Fernandes (FEG - UNESP, Bolsista PIBIC/CNPq)
E-mail: rafaelfernandes85@gmail.com

Dr. Hélio Koiti Kuga (DMC/INPE, Orientador)
E-mail: hkk@dem.inpe.br

Dr. Rodolpho Vilhena de Morais (FEG – UNESP, Co-orientador)
E-mail: rodolpho@feg.unesp.br

Julho de 2006

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

O sistema NAVSTAR-GPS (NAVigation Satellite Timing and Ranging – Global Position System) é um sistema de posicionamento e navegação que utiliza satélites em órbitas circulares. Seu desenvolvimento é controlado pelo Departamento de Defesa dos EUA e destina-se a satisfazer as exigências tanto civis quanto militares na determinação precisa de posição, velocidade e tempo em um sistema de referência comum e em qualquer lugar sobre ou acima da superfície terrestre.

Determinar a órbita de um satélite artificial significa determinar a posição e a velocidade do satélite, em relação a um referencial inercial, utilizando um conjunto de medidas de observação do satélite. Estas observações podem ser obtidas através de sistemas de rastreamento em solo ou de sensores a bordo do veículo espacial. Técnicas modernas de observação são atualmente capazes de medir a distância entre o instrumento e o satélite com a precisão de centímetros ou até melhor.

Ao longo dos anos, os requisitos das missões ficaram mais rigorosos. Com isto, os métodos de determinação de órbita tiveram que se tornar mais precisos, sem aumentar gastos computacionais e financeiros. Vários métodos foram desenvolvidos e aperfeiçoados com relação ao modelo do sistema dinâmico, às medidas e às técnicas de estimação.

Determinação de órbita usando sistema de navegação por satélites, como o sistema GPS, é uma atividade que surgiu no começo dos anos 80. Desenvolvimentos recentes têm mostrado precisão de poucos centímetros para missões de satélites com altímetro. Um ambiente de operação de pós-processamento é necessário para atingir esta precisão, com um atraso do tempo que pode atingir uma semana ou mais (Bertiger et al., 1994).

A disponibilidade a bordo de conhecimento contínuo e preciso da órbita de um satélite artificial torna prática a idéia de aumentar o grau de autonomia do sistema de controle, reduzindo a necessidade de intervenções em solo. No INPE, já existem alguns trabalhos sendo realizados para realizar controle autônomo de satélites artificiais usando GPS e outros sistemas, como os trabalhos de Orlando *et. al.* (1997) e Orlando e Kuga (1999, 2000a, 2000b).

O INPE (Instituto Nacional de Pesquisas Espaciais) vem determinando órbita de seus satélites com muito sucesso utilizando estações de rastreamento terrestres localizadas em pontos estratégicos pelo Brasil. Mas, a necessidade de desenvolver algoritmos com maior precisão e baixo custo computacional é sempre um desafio constante. A cada dia, novas técnicas vêm sendo desenvolvidas, como a utilização do sistema GPS. Porém, este tipo de tecnologia ainda não foi utilizada pelo INPE ou por qualquer instituto brasileiro.

Assim, este trabalho tem por objetivo investigar, desenvolver, implementar e comparar soluções de navegação possíveis através do uso de medidas obtidas por receptores GPS, em ambiente espacial, com geração de soluções em tempo real.

CAPÍTULO – 2

SISTEMA GPS

2.1 DESCRIÇÃO DO SISTEMA GPS

O sistema GPS (*Global Positioning System*) é um sistema de navegação por satélites desenvolvido pela Força Aérea dos Estados Unidos durante as décadas de 70 e 80 e colocado em operação na década de 90.

O princípio básico de funcionamento do GPS é determinar a posição e a velocidade tridimensionais e o tempo com alta precisão. O sistema GPS permite ao receptor determinar sua posição e tempo em qualquer lugar da superfície terrestre e a qualquer hora utilizando dados de apenas quatro satélites. O sistema pode ter um número ilimitado de usuários simultaneamente em qualquer parte do mundo.

O princípio de navegação por satélites se baseia no princípio da triangulação, a qual consiste na transmissão de sinais e dados das posições dos satélites GPS em relação a um sistema de coordenadas. O receptor mede o tempo de transmissão do sinal, o que permite calcular a distância entre o usuário e os satélites GPS, e decifra os dados. Se o relógio do receptor estiver sincronizado com os relógios dos satélites GPS, a medida das distâncias de três diferentes satélites GPS, em posição conhecida, permitirão ao usuário calcular a sua posição. Se o relógio do receptor não estiver sincronizado com os relógios dos satélites, serão necessários quatro satélites GPS, sendo o quarto para determinar o desvio do relógio, uma

quantidade desconhecida. As medidas da distância com relógio impreciso são chamadas de *pseudo-distância.*

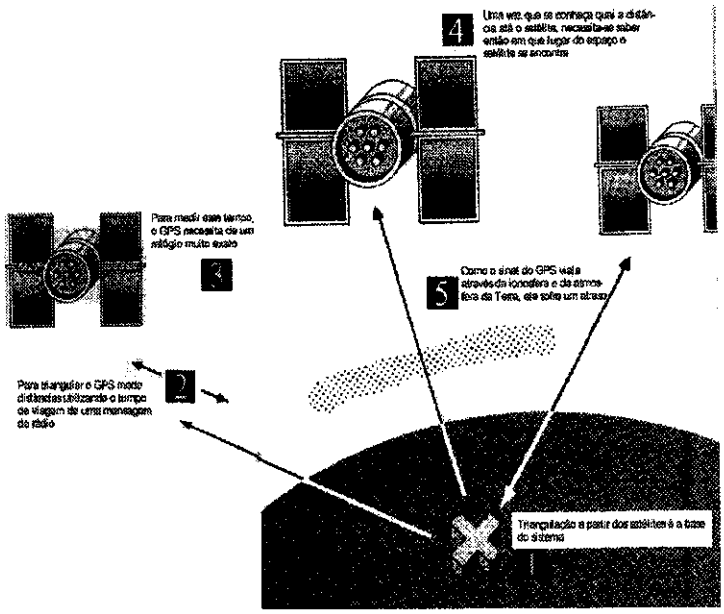


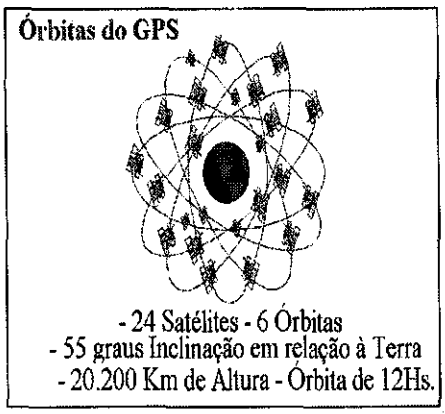
Figura 1: Princípio da triangulação (Fonte: Silva, 2001)

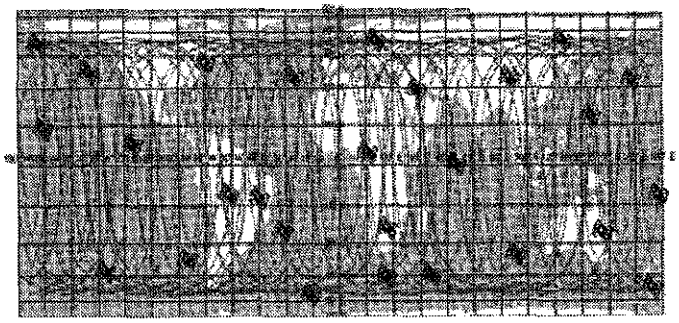
2.2 ESTRUTURA DO SISTEMA GPS

O Sistema GPS é compreendido de três segmentos: Espacial, Controle e Usuário que são descritos a seguir.

2.2.1 SEGMENTO ESPACIAL

O sistema espacial tem a função de gerar e transmitir códigos, a fase da portadora e a mensagem de navegação e consiste de uma constelação de 24 satélites GPS (21 navegando e três de reserva) em seis planos orbitais com período de 11h e 58 min. O raio da órbita é 26.560 km com excentricidade de 0,0131. Cada plano da órbita contém quatro satélites igualmente espaçados, como mostra a Figura 3, distribuídos de tal forma que forneça uma visibilidade simultânea de pelo menos quatro satélites para um usuário localizado em qualquer parte do mundo e em qualquer instante. Entretanto, o mesmo satélite se torna visível quatro minutos mais cedo a cada dia devido a diferença de quatro minutos por dia, entre a órbita do satélite e a rotação da Terra. A meta de tempo de vida dos satélites é de 7,5 anos (Leick, 1994).





Figuras 2 e 3 : Constelação do Sistema GPS (Fonte: Dana)

2.2.2 SEGMENTO DE CONTROLE

A função do segmento de controle é produzir o Tempo GPS e as efemérides dos satélites e gerenciar os veículos espaciais; isto é, atualizar periodicamente as informações que são transmitidas por todos os satélites, isto inclui as efemérides dos satélites, o *status*, os dados do relógio e o Almanaque.

O segmento de controle consiste de uma Estação Mestre de Controle (MCS), localizada na base de *Falcon Air Force* próxima a Colorado Springs no Colorado, Estados Unidos, e mais quatro estações de monitoramento no Hawaii, Kwajalein, Diego Garcia e Ascension Island. (Amato-1999)

A localização de cada estação de monitoramento oferece um monitoramento constante de cada satélite. Todas as estações de monitoramento trilham os satélites, determinam a sua performance operacional, chekam os parâmetros, e passam estas informações para a Estação de Controle Master. A estação Master pode então determinar os parâmetros de órbita de um satélite corrente, e transferir dados de correção para o mesmo satélite.

Figura 4 : Localização das estações de monitoramento (Fonte: Dana)

2.2.3 SEGMENTO USUÁRIO

O segmento usuário consiste de receptores militares e civis especialmente designados para receber, decodificar e processar os sinais dos satélites. O usuário observa e grava as transmissões de vários satélites e aplica algoritmos de solução para obter a sua posição, velocidade e tempo.

Os receptores GPS convertem os sinais recebidos em estimativas da posição, velocidade e tempo. Quatro satélites são exigidos para calcular as quatro dimensões de X, Y, Z (posição) e tempo. Existem diversos tipos de receptores para várias aplicações, dentre eles estão receivers para uso aeronáutico e marinho, placas para uso no computador, de uso portátil, para mapeamento, módulos, placas, entre outros.

2.3 SINAIS DOS SATÉLITES GPS

O sinal GPS é transmitido em duas freqüências: um sinal primário de 1575.42 MHz (chamado de L1) e uma transmissão secundária de 1227.6 MHz (chamado de L2). Estes sinais são gerados sincronamente de modo que o usuário que recebe os dois sinais pode diretamente calibrar o atraso ionosférico e aplicar correções apropriadas. Entretanto, muitos usuários civis somente usam a freqüência L1.

Os sinais são modulados com dois tipos de códigos. Existem duas modulações na freqüência mais alta (L1), mas somente uma única modulação (protegida) em L2. A freqüência L1 carrega a mensagem de navegação e os sinais do código SPS. A freqüência L2 é usada para medir o atraso ionosférico através de receptores equipados PPS.

O código C/A (*Coarse/Acquisition*) é transmitido em 1.023 MHz e é modulado na freqüência L1. É de uso civil e é sempre transmitido, mas está sujeito a degradações. O uso deste sinal é chamado de Serviço Padrão de Posicionamento (SPS). O código P (*Precise*) é, algumas vezes, chamado de código protegido e é transmitido em 10.23 MHz (10 vezes mais rápido que C/A), modulado em L1 e L2. Devido a sua modulação mais alta, o sinal é mais preciso. O sinal fornece o Serviço de Posicionamento Preciso (PPS). É de uso militar e para usuários autorizados. Esta característica é conhecida como *Antispoofing* (A-S). Quando criptografado, o código P torna-se código Y (ou P/Y).

Os operadores militares do sistema têm a capacidade de degradar intencionalmente a precisão do sinal C/A dessincronizando o relógio do satélite ou incorporando pequenos erros nas efemérides transmitidas que é chamada Disponibilidade Seletiva (SA).

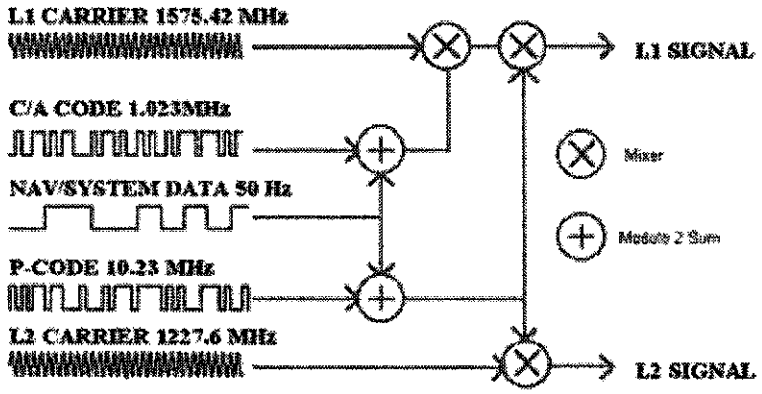


Figura 5 : Sinais do satélite GPS (Fonte: Dana)

Os receptores GPS recebem continuamente uma série de dados dos satélites GPS na forma de bits modulados em sinais. Estes dados são chamados de mensagem de navegação e são transmitidos nas freqüências L1 (1575.42 MHz) e L2 (1227.6 MHz). Estas informações são computadas e controladas pelo segmento de controle e freqüências são usadas para regular o atraso ionosférico. Os sinais L1 e L2 transmitem ao usuário as efemérides do satélite, as correções do relógio do satélite, parâmetros atmosféricos, parâmetros orbitais de todos os satélites e outros dados relevantes sobre o sistema em geral.

Os observáveis GPS são as distâncias deduzidas das diferenças medidas de tempo ou fase baseadas na comparação entre os sinais recebidos e os sinais gerados pelo receptor. Pode-se dizer que esta diferença de tempo é simplesmente o tempo que o sinal leva para propagar do

satélite à antena do receptor. Portanto, um dado observável é a diferença de tempo multiplicada pela velocidade da luz. O observável assim obtido é chamado de pseudo-distância, porque utilizam dois relógios assincronizados com o tempo, um no satélite (aquele que governa a geração do sinal do GPS) e outro no receptor (aquele que governa a geração da réplica do código), além de incluir todos os erros atmosféricos e de propagação do sinal. O receptor pode determinar este erro de sincronização junto com as suas coordenadas de posição e velocidade.

Os observáveis básicos do GPS são a pseudo-distância do código, que chamamos somente por pseudo-distância, e a pseudo-distância da fase da portadora, que chamamos somente por fase da portadora.

A equação fundamental da pseudo-distância, medida em metros, é dada por:

$$\rho_c = \rho + c[\Delta t_{GPS}(t) - \Delta t_u(t)] + \Delta_{ION} + \Delta_{TRO} + \varepsilon = c\tau\,, \qquad (1)$$

onde $\rho = \sqrt{(x_{GPS} - x)^2 + (y_{GPS} - y)^2 + (z_{GPS} - z)^2}$ é a distância geométrica entre a antena do satélite GPS e a do receptor; x, y, e z (incógnitas) são as coordenadas da posição da antena do receptor GPS; x_{GPS}, y_{GPS}, e z_{GPS} (enviadas) são as coordenadas da posição do satélite GPS; c é a velocidade da luz (conhecida); Δt_{GPS} é o erro do relógio do satélite GPS com respeito ao tempo do GPS (calculado); Δt_u é o erro de sincronização entre o relógio do satélite GPS e o receptor GPS, chamado de desvio do relógio do receptor (incógnita); t aqui é utilizado para explicitar a dependência temporal de Δt_{GPS} e Δt_u; Δ_{ION} e Δ_{TRO} são os erros devidos aos atrasos ionosféricos e troposféricos, respectivamente (estimados); ε são os ruídos de observação (estimados ou negligenciados); e τ é o tempo de propagação do sinal observado entre o satélite GPS e a antena do receptor (medido).

A medida da fase da portadora é definida como a diferença entre a fase da portadora do satélite GPS recebida pela antena do receptor e a fase do oscilador interno do receptor na época da medida. A equação fundamental da fase da portadora, medida em ciclos, é dada por:

$$\phi_c = \rho + c[\Delta t_{GPS}(t) - \Delta t_u(t)] - \Delta_{ION} + \Delta_{TRO} + \lambda N + \varepsilon\,, \qquad (2)$$

onde λ é o comprimento de onda da portadora (conhecido) e N é o número inteiro de ciclos completos da portadora (incógnita).

2.4 PRINCIPAIS FONTES DE ERRO

As principais fontes de erro do sistema GPS são as seguintes:

a) erro devido à geometria dos satélites com relação ao observador;
b) desvio dos relógios dos satélites;
c) atraso de propagação e processamento dos sinais pelos circuitos dos satélites;
d) erros devidos a trajetórias múltiplas (reflexões) dos sinais;
e) efeitos da atmosfera sobre a velocidade e a trajetória de propagação dos sinais transmitidos;
f) erros devidos à resolução, não sincronismo, e ruído, do receptor do usuário;
g) erro na determinação da posição dos satélites (erro de efemérides).

CAPÍTULO 3

MÉTODOS PARA SOLUÇÕES DE NAVEGAÇÃO ATRAVÉS DE GPS

Existem diferentes métodos programáveis para se obter soluções de navegação através do GPS. Eles podem ser classificados em:

- Métodos geométricos
- Métodos algébricos
- Métodos estatísticos.

3.1 MÉTODO GEOMÉTRICO

Os métodos geométricos são basicamente métodos simples que fornecem uma estimativa inicial grosseira para a solução de navegação, que pode ser refinada posteriormente através dos outros métodos, ou através de filtros estatísticos. O método descrito em Lopes e Kuga (1997) será implementado. Outro método geométrico alternativo seria aquele descrito em Kleusberg (1994).

Para produzir um método simples e sistemático para resolver o problema de inicialização do algoritmo, o *bias* será negligenciado. Assim, uma solução linear aproximada pode ser encontrada da seguinte maneira:

$$y_{pi}^2 \approx |r - R_i|^2 = r^T r + R_i^T R_i - 2R_i^T r, \tag{3}$$

onde r é o vetor posição do satélite usuário, R_i é o vetor posição do i-ésimo satélite GPS. Subtraindo a Eq. (3) de sua média aritmética em i, temos:

$$y_{pi}^2 - < y_{pi}^2 > \approx R_i^T R_i - < R_i^T R_i > -2[R_i - < R_i >]^T r, \tag{4}$$

onde <.> representa o operador de média aritmética.

Definindo:

$$\tilde{R} \equiv [...: R_i - < R_i > :...]^T \tag{5}$$

e

$$z \equiv \frac{1}{2}\left[...: R_i^T R_i - < R_i^T R_i > -\left(y_{pi}^2 - < y_{pi}^2 >\right):...\right]^T \tag{6}$$

resulta em:

$$\tilde{R} r \approx z. \tag{7}$$

Uma solução aproximada deste sistema linear determinado pode ser dada por:

$$r^0 = (\tilde{R}^T \tilde{R})^{-1} \tilde{R} z. \tag{8}$$

A condição de existência para a pseudo inversa de $\tilde{R}$ pode ser especificada da seguinte maneira: um número $n \geq 4$ de satélites GPS não coplanar deve ser observado pelo usuário.

3.2 MÉTODO ALGÉBRICO

O nível de precisão da solução de navegação baseada em medidas GPS depende do tipo de medida coletada (depende do tipo e qualidade do receptor), da duração que as medidas foram coletadas e como elas foram modeladas e processadas. Se as medidas GPS são processadas

em tempo real, o problema de posicionamento mais simples consiste em resolver simultaneamente um conjunto de equações de navegação baseado em única freqüência.

No mínimo quatro medidas simultâneas são necessárias para formar a solução. A entrada para esse algoritmo são os vetores posição (3 x 1) dos satélites GPS e as medidas pseudoranges dos satélites GPS para o receptor. O método algébrico de solução de navegação estudado e proposto é computacionalmente eficiente e numericamente estável, conforme Bancroft (1985).

As equações do sistema de navegação GPS são geralmente resolvidas com uma aplicação do método de Newton:

$$x_{n+1} = x_n + H^{-1}(t - f(x_n)), \tag{9}$$

onde x é um vetor compreendendo a coordenada de posição do usuário junto com a correção do relógio, t é um vetor de medidas de quatro pseudo-distância e H é uma matriz de derivada parcial $H = f_x$.

Façamos x e $\{s_i : 1 \le i \le n\}$ indicar as coordenadas de posição do satélite e usuário no sistema de coordenadas cartesianas; e $\{t_i : 1 \le i \le n\}$ as medidas da pseudo-distância coletadas pelo usuário de cada um dos n satélites:

$$t_i = d(x, s_i) + b, \tag{10}$$

onde $d(x,y)$ é a distância de x a y e b a correção do relógio. Definimos os vetores de dados de colunas 1 x 4:

$$a_i = (s_i^T t_i)^T, \; 1 \le i \le n. \tag{11}$$

Definimos:

$$A = (a_1 \quad a_2 \quad a_3 \quad \ldots \quad a_n)^T, \tag{12}$$

$$i_0 = (1 \quad 1 \quad 1 \quad \ldots \quad 1)^T, \tag{13}$$

$$r = (r_1 \quad r_2 \quad r_3 \quad \ldots \quad r_n)^T , \tag{14}$$

onde $r_i, 1 \le i \le n$ é calculado por:

$$r_i = \langle a_i, a_i \rangle / 2 . \tag{15}$$

Calculamos a inversa generalizada:

$$B = (A^T W A)^{-1} A^T W , \tag{16}$$

onde W é a matriz positiva simétrica. Calculamos os vetores coluna 1 x 4 u e v de:

$$u = Bi_0 \tag{17}$$

e

$$v = Br \tag{18}$$

juntamente com os coeficientes $E,F,G,$ definidos por:

$$E = \langle u, u \rangle , \tag{19}$$

$$F = \langle u, v \rangle - 1 , \tag{20}$$

$$G = \langle v, v \rangle . \tag{21}$$

Resolvendo a equação quadrada:

$$E\lambda^2 + 2F\lambda + G = 0 \tag{22}$$

para o par de raízes $\lambda_{1,2}$.

$$y_{1,2} = \lambda_{1,2} u + v. \tag{23}$$

Então com a identificação

$$y^T = (x^T - b)^T \tag{24}$$

também o par x_1, b_1 ou o par x_2, b_2 resolverá o problema GPS para a posição do usuário e correção do relógio. Para diferenciar a solução real, substituímos de volta as equações definindo as pseudo-distância originais.

3.3 METODO ESTATÍSTICO

O método estatístico utiliza redundância de medidas para estatisticamente obter a melhor solução. Os sinais GPS podem ser recebidos e decodificados adequadamente pelos receptores GPS. Se os sinais forem recebidos adequadamente, um conjunto de três satélites seria suficiente para suprir as dificuldades geométricas (Lopes e Kuga, 1988, 1997).

Porém, principalmente devido a derivas no relógio, um bias é introduzido na distância computada geometricamente (pseudorange) tornando essencial o uso de quatro satélites.

Em trabalhos anteriores, Lopes e Kuga propuseram um método estatístico de mínimos quadrados para resolver o problema de determinação de órbita através de medidas GPS. O método era especialmente adequado para medidas processadas por um número redundante de satélites GPS, tornando-o distinto de métodos convencionais que invariavelmente devem analisar a matriz DOP (Dilution Of Precision).

A vantagem do método consiste no processamento de todas as medidas válidas de pseudorange assumindo precisões no mínimo igual ou melhor que as convencionais. Outro benefício do método é a falta de necessidade de analisar várias matrizes DOP 4 x 4 para selecionar a melhor configuração entre os satélites visíveis.

Para determinação de posição estática de baixa precisão, a solução de navegação pode ser obtida, por exemplo, através do algoritmo proposto por Lopes e Kuga (1988):

$$\text{Minimizar } L(r, \rho_i) = \frac{1}{2} \sum_i a_i \left| r - (R_i + \rho_i) \right|^2 \tag{25}$$

Sujeito a $\rho_i^t \rho_i = y_i^2, \quad i = 1,2,\ldots,n,$ (26)

Dados $\{(R_i, y_i, a_i), i = 1,2,\ldots,n; \quad n \geq 3\}$ (27)

onde $\boldsymbol{r}$ é o vetor posição do satélite usuário (ou alvo); R_i é o vetor posição do i-ésimo satélite GPS; ρ_i é o vetor posição relativo do satélite usuário em relação ao i-ésimo satélite GPS; y_i é o pseudo-range (pseudo-distância) medido a partir do i-ésimo satélite GPS; e a_i é um peso positivo. Assume-se que tanto y_i quanto R_i estão corrompidos por erros aleatórios não-viesados ("unbiased"), δy_i e δR_i com covariâncias dadas por:

$$E[(\delta y_i)^2] = \sigma_{yi}^2,$$
$$E[\delta R_i \; \delta^t R_i] = \sigma_{Ri}^2 I \tag{28}$$

onde $E[.]$ é o operador esperança; I é a matrix identidade; e os desvios-padrão σ_{yi} e σ_{Ri} são quantidades conhecidas. Pode-se então modificar levemente o método para levar em conta os erros sistemáticos. O método estendido é então formulado como:

Minimizar $L^*(\boldsymbol{r}, \rho_i, \Delta y) = L(\boldsymbol{r}, \rho_i) + \frac{1}{2} a^* \Delta^2 y,$ (29)

Sujeito a $\rho_i^t \rho_i = (y_{pi} + \Delta y)^2, \quad i = 1,2,\ldots,n,$ (30)

Dados $\{a^*, (R_i, y_i, a_i), i = 1,2,\ldots,n; \quad n \geq 4\}$ (31)

onde y_{pi} é a medida de pseudo-range, Δy é uma constante a ser adicionada a y_{pi} para correção do bias, e a^* é um peso positivo. Dessa forma assume-se que o pseudo-range pode ser modelado por:

$$y_{pi} = \bar{y}_i + b + \delta y_i \tag{32}$$

onde $\bar{y}_i$ é o verdadeiro range e b é o bias, com $E[b] = 0$, $E[b^2] = \sigma_b^2$, e σ_b está diretamente relacionado à precisão σ_t do relógio do receptor GPS, ou seja $\sigma_b = c\sigma_t$, onde c é a velocidade da luz. Embora b tenha sido modelado como erro aleatório de média nula, deve-se

lembrar que de fato ele representa um bias porque o que é adicionado a todas as medidas de range $\bar{y}_i$ é o mesmo valor constante: uma simples realização b. Os pesos positivos a_i e a^* são supostos obedecerem ao vínculo de normalidade:

$$a^* + \sum_i a_i = 1. \tag{33}$$

A derivação da solução pode ser obtida através do método dos multiplicadores de Lagrange. Definindo-se uma função de custo modificada:

$$l^* = L^* + \frac{1}{2}\sum_i \lambda_i a_i [\rho_i^t \rho_i - (y_{pi} + \Delta y)^2], \tag{34}$$

e impondo as condições de otimalidade:

$$\frac{\partial l^*}{\partial \boldsymbol{r}} = \sum_i a_i [\boldsymbol{r} - (R_i + \rho_i)]^t = 0, \tag{35}$$

$$\frac{\partial l^*}{\partial \rho_i} = a_i \{\lambda_i \rho_i - [\boldsymbol{r} - (R_i + \rho_i)]\}^t = 0, \tag{36}$$

$$\frac{\partial l^*}{\partial \Delta y} = a^* \Delta y - \sum_i \lambda_i a_i (y_{pi} + \Delta y) = 0, \tag{37}$$

e levando em conta o vínculo (30) segue-se que:

$$\sum_i a_i u_i [|\boldsymbol{r} - R_i| - (y_{pi} + \Delta y)] = 0 \tag{38}$$

$$\rho_i = \frac{\boldsymbol{r} - R_i}{1 + \lambda_i} = (y_{pi} + \Delta y) u_i \,, \tag{39}$$

$$a^* \Delta y = \sum_i \lambda_i a_i (y_{pi} + \Delta y) \,, \tag{40}$$

onde u_i é o versor dado por:

$$u_i = \frac{|\boldsymbol{r} - R_i|}{y_{pi} + \Delta y} - 1 \quad . \tag{41}$$

Resolvendo as equações (41) e (39) para λ_i tem-se:

$$\lambda_i = \frac{|\boldsymbol{r} - R_i|}{y_{pi} + \Delta y} - 1 \quad . \tag{42}$$

Usando o vínculo de normalidade (33) junto com a equação (42) acima, pode-se escrever:

$$\Delta y = \sum_i a_i [|\boldsymbol{r} - R_i| - y_{pi}] \tag{43}$$

e finalmente definindo-se a quantidade U como $U = \sum_i a_i u_i$ resulta:

$$f(\boldsymbol{r} - R_i, y_{pi}) \equiv \sum_i a_i (u_i - U)[|\boldsymbol{r} - R_i| - y_{pi}] = 0. \tag{44}$$

Este conjunto de equações (41), (43) e (44) fornece a solução fechada para a determinação de posição do satélite, utilizando medidas GPS viesadas, via método de mínimos quadrados. Estas equações podem ser resolvidas numericamente via o método de Newton-Raphson quando um chute inicial r^0 está disponível:

$$r^{k+1} = r^k - F^{-1} \cdot f(r^k - R_i, y_{pi}), \tag{45}$$

$$F = \left.\frac{\partial f}{\partial r}\right|_{\bar{r}-\bar{R}_i, \bar{y}_i} \tag{46}$$

Desde que os valores obedeçam a relação:

$$\left|\bar{r} - \bar{R}_i\right| - \bar{y}_i = 0, \tag{47}$$

Isto implica que:

$$F = \Lambda - UU^T, \tag{48}$$

Onde a matriz Λ é definida por:

$$\Lambda = \sum_i a_i u_i u_i^T, \tag{49}$$

Para F não singular, $u_i - s$ não pode estar no mesmo cone; por exemplo, a linha de visada dos satélites de GPS não pode cruzar a Esfera Celeste em pontos coplanares.

Para precisões realísticas no chute inicial, este método converge em poucos passos a um alto nível de precisão.

CAPÍTULO 4

INTERFACE SERIAL

4.1 – PRINCÍPIOS DA COMUNICAÇÃO SERIAL

A porta serial convencional (não a nova porta USB) é uma porta E/S muito antiga. Quase todos os PC's as possuem. A especificação comum é RS-232 (ou EIA-232). O conector para a porta serial é visto muitas vezes como um ou dois conectores de 9 pinos (em alguns casos 25 pinos) na parte posterior do PC.

O computador precisa saber em qual PORTA SERIAL está conectado o modem para poder se comunicar com ele. As portas seriais em micros PC são chamadas de COM1, COM2, COM3 e COM4.

Embora se tenha quatro portas de comunicação à disposição, existem algumas considerações a se fazer quanto à correta configuração:

Cada porta COM possui uma chamada de interrupção que ela usará para informar ao micro que está precisando de atenção. Essas interrupções são chamados de IRQ, e são numerados de 0 a 15. Para as portas seriais, existem dois IRQs:

PORTA	IRQ
COM1	IRQ4
COM2	IRQ3
COM3	IRQ4
COM4	IRQ3

Assim sendo, não é recomendado ter dois periféricos em duas portas com IRQs iguais, ou seja, ao mesmo tempo na COM1 e COM3 ou na COM2 e COM4.

Se o modem for INTERNO e se puder configurar à gosto a porta serial a usar, devem ser seguidos algumas recomendações:

- COM1: Use essa porta se seu micro não tiver uma conexão serial já como COM1 (o que não é provável). É comum de se ter um mouse instalado na COM1.
- COM2: Se seu micro estiver equipado com somente uma porta serial na COM1, e se você não já está usando a COM2 para outro equipamento, use essa porta (é a configuração mais comum e os modems normalmente vêm com a COM2 por default).
- COM3 ou COM4: Somente em casos especiais, normalmente não é o caso.

Além de poder configurar a porta serial, os modems internos também permitem configurar o IRQ a ser usado, sendo que assim você pode contornar o problema dos IRQs exposto acima.

Mas a porta serial é mais do que apenas isso. Inclui a eletrônica associada, que deve produzir sinais de acordo com a especificação EIA-232. Um pino é usado para enviar bytes de dados e outro para receber bytes de dados. Um outro pino é um terra para sinais comum. Os outros pinos "úteis" são usados principalmente para fins de sinalização com uma tensão negativa constante significando "desligado", e uma tensão positiva constante significando "ligado".

4.2 - COMO FUNCIONA -- VISÃO GERAL

4.2.1 TRANSMISSÃO

Transmissão é o envio de bytes pela porta serial para fora do computador. O que segue é uma explicação de como funcionam portas seriais mais antigas e obsoletas (com buffers de apenas 1 byte). Quando o computador quiser enviar um byte para fora, a UCP envia o byte pelo bus

dentro do computador ao endereço de E/S da porta serial. A porta serial toma o byte, e o envia, um bit de cada vez (uma corrente serial de bits) pelo pino de transmissão do conector serial.

A maior parte do trabalho na porta serial é executado pelo chip UART (Universal Asynchronous Receiver Transmitter) ou similar. O programa acionador de dispositivo, rodando na UCP, envia um byte ao endereço de E/S serial. Esse byte vai para um buffer de transmissão de 1 byte na porta serial. Quando uma porta está pronta para transmiti-lo, ela move esse byte para o registrador de deslocamentos para transmissão e envia-o pelo pino de transmissão bit a bit. Agora o buffer de transmissão está vazio e um outro byte pode ser nele colocado enquanto o byte original estiver sendo transmitido do registrador de deslocamentos para transmissão. Uma vez que um byte foi completamente transmitido, a porta toma um outro byte de seu buffer e envia-o também bit a bit. A porta continua a fazer isto enquanto houver um novo byte no buffer de transmissão.

Para que não haja quaisquer falhas na transmissão, a UCP deve manter o buffer de transmissão de 1 byte preenchido. Ela o faz usando interrupções. Assim que um byte foi movido para fora do buffer de transmissão de 1 byte, e está a ser transmitido do registrador de deslocamentos para transmissão, o buffer de transmissão está vazio e requer outro byte. A porta serial então emite uma interrupção para dizer que está pronta para o próximo byte.

A interrupção é na verdade uma tensão aplicada num fio dedicado àquela porta serial (embora em alguns casos, o mesmo fio possa ser compartilhado por mais de uma porta). O acionador de dispositivos é notificado sobre a interrupção e verifica os registradores em endereços para saber o que aconteceu. Ele descobre que o buffer de transmissão da porta serial está vazio e esperando por mais um byte. Então, se houver mais bytes para enviar, o acionador envia o próximo byte para o endereço de E/S serial. Este próximo byte deve chegar quando o byte anterior está ainda no registrador de deslocamentos de transmissão e ainda sendo transmitido bit a bit. Este novo byte vai para o buffer de transmissão mas não pode ser movido para o registrador de deslocamentos enquanto o byte anterior não tiver sido completamente transmitido. Quando o último bit tiver sido enviado, as seguintes três ações ocorrem simultaneamente:

1. Um novo byte é movido do buffer de transmissão para o registrador de deslocamentos para transmissão.

2. A transmissão deste novo byte bit a bit começa.

3. Uma outra interrupção é emitida para dizer ao acionador de dispositivos enviar um outro byte para o buffer de transmissão.

Dizemos, portanto, que a porta serial é acionada por interrupções. A cada vez a porta serial emite uma interrupção, a UCP envia-lhe um outro byte. Uma vez que um byte tiver sido enviado ao buffer de transmissão pela UCP, então a CPU fica livre para executar alguma outra atividade, até receber a próxima interrupção.

A porta serial transmite bits a uma velocidade fixa, que é escolhida pelo usuário, e às vezes chamada de taxa em bauds. A porta serial adiciona bits extras a cada byte, de modo que há muitas vezes 10 bits por byte. A uma taxa (também chamada de velocidade) de 19,200 bits por segundo (bps), ou seja 1920 bytes/s (e também 1920 interrupções/s).

4.2.2 RECEPÇÃO

Receber bytes por uma porta serial é semelhante a enviá-los, só que na direção oposta. É também acionado por interrupções. Para o tipo obsoleto de porta serial com buffers de um byte, quando um byte é recebido completamente, vai para um buffer de recepção de 1 byte.

Então a porta dá à CPU uma interrupção para dizer-lhe que apanhe um byte, de modo que a porta serial tenha espaço para armazenar o byte que está sendo recebido.

4.2.3 PINOS E FIOS

Antigos PCs usavam conectores de 25 pinos, mas somente 9 pinos eram realmente usados, por isso hoje a maioria dos conectores têm 9 pinos. Cada um dos 9 pinos são ligados a um fio. Além dos fios isolados usados para transmitir e receber dados, um outro pino (e fio) é o terra para sinais. A tensão em qualquer fio é medida em relação a esse terra. Há mais fios ainda que são para controle (sinalização) somente. Todos esses sinais poderiam ser enviados através de um único fio, porém, existe um fio separado dedicado a todo tipo de sinal.

9 Pinos	25Pinos	Sigla	Nome	Significado

3	2	TxD	Transmitir Dados	Transmitir byte para fora do PC
2	3	RxD	Receber Dados	Receber bytes para dentro do PC
7	4	RTS	Request To Send	Controle de fluxo RTS/CTS
8	5	CTS	Clear To Send	Controle de fluxo RTS/CTS
6	6	DSR	Data Set Ready	Pronto para comunicar
4	20	DTR	Data Terminal Ready	Pronto para comunicar
1	8	DCD	Data Carrier Detect	Modem conectado a outro
9	22	RI	Ring Indicator	Linha telefônica tocando
5	7		Terra para sinal	

LISTA DOS PINOS da PORTA SERIAL (--> direção para fora do PC)

Alguns (ou todos) esses fios de controle são chamados "linhas de controle de modem". Fios de controle de modem estão em estado ligado de +12 volts ou em estado desligado de -12 volts. Um desses fios serve para sinalizar ao computador a parar de enviar bytes para fora da porta serial. Por outro lado, um outro fio sinaliza ao dispositivo acoplado à porta serial a parar de enviar bytes para o computador. Se o dispositivo acoplado é um modem, outros fios podem dizer ao modem para desligar a linha telefônica ou dizer ao computador que uma conexão foi feita ou que a linha telefônica está chamando (alguém está tentando chamar para dentro).

Somente 3 dos 9 pinos possuem uma atribuição fixa: transmitir, receber e terra para sinal. Estes são fixados pelo hardware e não podes alterá-los. Mas as outras linhas de sinal são controladas por software e podem fazer (e significar) quase qualquer coisa. Entretanto elas podem estar em um de dois estados: positivo (+12 volts) ou negativo (-12 volts). Positivo é "ligado" e negativo é "desligado".

4.2.4 CABLAGEM ENTRE PORTAS SERIAIS

Um cabo de uma porta serial sempre se conecta a uma outra porta serial. Um modem ou outro dispositivo que se conecta a outra porta serial tem uma porta serial nele embutida. Para modems, o cabo sempre vai direto: pino 2 vai para pino 2, etc. Diz-se que o modem é um DCE (Data Communications Equipment) e que o computador é um DTE (Data Terminal Equipment). Assim, para conectar um DTE a um DCE usa-se um cabo direto.

Para conectar um DTE a outro DTE usa-se um cabo de modem nulo, e existem muitas maneiras de ligar os fios num tal cabo. Existem boas razões pelas quais funciona assim. Uma é que os sinais são unidirecionais. Se o pino 1 envia um sinal para fora, mas é incapaz de receber qualquer sinal, então obviamente não podes conectá-lo ao pino 1 de um mesmo tipo de dispositivo. Se isso for feito, ambos enviariam sinais um ao outro mas nenhum deles seriam capazes de receber qualquer sinal. Existem dois modos de lidar com esta situação. Um é ter dois diferentes tipos de equipamento onde o pino 1 do primeiro tipo envia o sinal ao pino 1 do segundo tipo (o que recebe o sinal). Isto é o que ocorre quando um PC (DTE) é conectado a um modem (DCE). Existe um segundo modo de fazê-lo sem ter dois diferentes tipos de equipamento: conectar o pino 1 (um pino que envia) a um pino que recebe no mesmo equipamento. É assim que são conectado 2 PCs um ao outro ou um PC a um terminal (DTE a DTE). O cabo usado para isto é chamado cabo de modem nulo. As designações dos pinos seriais foram originalmente definidas para conectar um terminal burro a um modem. O terminal era um DTE (Data Terminal Equipment - equipamento terminal de dados) e o modem um DCE (Data Communication Equipment - equipamento de comunicação de dados). Hoje o PC é usualmente um DTE ao invés de um terminal (mas terminais reais podem ainda ser usados desta forma). Os nomes dos pinos são os mesmos em ambos, DTE e DCE. As palavras "receber" e "transmitir" são usadas do ponto de vista do PC (DTE). O pino de transmissão do PC transmite ao pino de "transmissão" do modem (mas realmente o modem está recebendo os dados deste pino, de modo que do ponto de vista do modem este seria um pino de recepção). A porta serial foi originalmente concebida para ser usada a fim de conectar um DTE a um DCE, o que torna o cabo simples: todos os fios diretos, sem cruzamentos. Assim, quando se conecta um modem raramente se necessita preocupar-se qual pino é qual. Mas as pessoas queriam conectar um DTE a outro DTE (por exemplo, um computador a um terminal) e várias formas foram encontradas para fazer isto, fabricando-se vários tipos de cabos especiais. Neste caso, qual pino se conecta a qual outro pino se torna muito importante.

4.2.5 ENDEREÇO DE E/S E IRQ

Como o computador necessita comunicar-se com cada porta serial, o sistema operacional deve saber que cada porta serial existe e onde se encontra (seu endereço de E/S). O Sistema Operacional também precisa saber qual fio (número IRQ) a porta serial vai usar para solicitar serviço da UCP do computador. A porta serial pede serviço enviando uma interrupção neste fio. Assim, toda porta serial deve armazenar em sua memória não volátil ambos, seu endereço de E/S e seu número IRQ (pedido de interrupção). Para o PCI bus não funciona exatamente

deste jeito, desde que o PCI bus tem seu próprio sistema de interrupções. Mas como o BIOS ciente do PCI faz com que os chips mapeiem essas interrupções PCI para IRQs, parece que se comporta exatamente como descrito acima. As portas seriais são rotuladas 0x3F8, 0x3E8, etc. correspondendo a COM1, COM2 etc. Cada qual desses nomes se refere a uma certa porta serial física é determinado pelo endereço de E/S armazenado no chip de hardware da porta física. Quando um endereço de E/S é posto no bus de endereços do computador, um outro fio é energizado. Isto diz tanto à memória principal ignorar o endereço quanto a todos os dispositivos que possuem Endereços de E/S (tal como a porta serial) a escutarem no endereço para ver se ele corresponde. Se o endereço corresponder, então o dispositivo de E/S lê os dados no bus de dados.

4.2.6 INTERRUPÇÕES

Quando a porta serial pega um byte, ela sinaliza a UCP para buscar o byte enviando um sinal elétrico conhecido como interrupção a um condutor dedicado. Ela enviará também essa interrupção se houver uma demora inesperada enquanto espera o próximo byte chegar (conhecido como tempo terminal (timeout)). Cada condutor de interrupção (dentro do computador) tem um número (IRQ) e a porta serial deve saber qual condutor usar para sinalizar. Por exemplo, 0x3F8 normalmente usa o IRQ número 4, conhecido como IRQ4.

Interrupções são emitidas sempre que a porta serial precisa chamar a atenção da UCP. É importante fazer isso em uma forma oportuna, pois o buffer dentro da porta serial contém 1 bytes entrante. Se a UCP não remover tal byte prontamente, então não haverá espaço sobrando para qualquer byte entrante e ocorrerá transbordamento, e bytes se perderão. Não há Controle de Fluxo para evitar isto.

Interrupções são também emitidas quando a porta serial acabou de enviar todo o byte de seu pequeno buffer de transmissão para fora do cabo externo. Então ela tem espaço para mais 1 byte saindo. A interrupção é para notificar a UCP daquele fato de modo que ela possa por mais bytes no pequeno buffer de transmissão a ser transmitido. Também, quando uma linha de controle de modem muda de estado, uma interrupção é emitida.

4.2.7 FLUXOS DE DADOS (VELOCIDADES)

Dados (bytes representando letras, figuras, etc.) fluem para dentro e para fora da porta serial. Taxas de fluxo (tais como 56k bits/segundo) são incorretamente chamadas de "velocidades".

É importante entender que a velocidade média é freqüentemente menor que a velocidade especificada. Tempos de espera resultam em menor velocidade média. Estas esperas podem incluir longas esperas de talvez um segundo devido ao Controle de Fluxo. No outro extremo pode haver esperas muito curtas de vários micro-segundos entre bytes. Se o dispositivo na porta serial (tal como um modem) não pode aceitar toda a velocidade da porta serial, então a velocidade média deve ser reduzida.

Controle de fluxo é o poder de parar o fluxo de bytes num fio. Inclui também meios de reiniciar o fluxo sem perda de bytes. O controle de fluxo é necessário em situações que necessitam permitir mudanças bruscas na taxa de fluxo.

O sintoma da falta de controle de fluxo é a falta de pedaços de dados em arquivos enviados sem o benefício do controle de fluxo. Isto ocorre porque quando o transbordo acontece, usualmente mais do que uns poucos bytes transbordam e são perdidos.
Freqüentemente centenas ou mesmo milhares de bytes se perdem, e todos em pedaços contínuos. Pode-se utilizar controles de fluxos via software ou mesmo por hardware. Para modems, é melhor usar controle de fluxo de "hardware" que usa dois fios dedicados a "controle de modem" para enviar os sinais de tensão para "parar" e "iniciar". O controle de fluxo de software usa os fios de receber e transmitir para enviar os sinais de iniciar e parar. Para isso usa os caracteres de controle ASCII DC1 (iniciar) e DC3 (parar). São apenas inseridos na corrente regular de dados. O controle de fluxo de software não é apenas reage mais lentamente, como também não permite o envio de dados binários pelo modem pois os dados binários podem conter os caracteres de controle DC1 ou DC3 usados no controle de fluxo.

Controle de Fluxo RTS/CTS e DTR/DSR

RTS/CTS e DTR/DSR são controles de fluxo em hardware e funcionam da mesma forma. Para obter fluxo de controle RTS/CTS, necessita-se selecionar hardware de fluxo de controle. Isto ativa o hardware de fluxo de controle RTS/CTS no acionador de dispositivos. Então quando um DTE (tal como um PC) quiser parar seu fluxo de entrada, nega o RTS. Negar o "Request To Send" (-12 volts) significa "Solicito NÃO enviar para mim" (pára o envio).

Quando o PC estiver pronto para mais bytes, ele positiva o RTS (+12 volts) e que o fluxo de bytes a ele se reinicia. Sinais de fluxo de controle são sempre enviados numa direção oposta ao fluxo de bytes sendo controlado. Equipamentos DCE (modems) funcionam do mesmo modo porém envia o sinal de parada para fora do pino CTS. Assim o fluxo de controle RTS/CTS usa 2 linhas.

Para conexões DCE-DTE as linhas são diretas então o sinal é recebido num pino com mesma posição relativa que o emissor. É RTS-->RTS (PC para modem) e CTS<--CTS (modem para PC). Para DTE-DTE a conexão é tambem fácil de entender. O pino RTS envia e o pino CTS recebe. Supõe que conectamos dois PCs (PC1 e PC2) através de suas portas seriais. Teremos RTS(PC1)-->CTS(PC2) e CTS(PC1)<--RTS(PC2). Em outras palavras, RTS e CTS cruzam-se. Tal cabo (com outros sinais cruzados tambem) é chamado cabo de "modem nulo". O que é algumas vezes confuso é que existe o uso original de RTS, onde significa quase o oposto da explicação anterior acima. O significado original é: solicito te enviar. Esta solicitação deveria ser enviada de um terminal (ou computador) a um modem que, se decidisse atender ao pedido, retornaria um CTS positivo de seu pino CTS para o pino CTS do computador: estará livre para me enviar. Nota que em contraste com o moderno fluxo bidirecional RTS/CTS, isto apenas protege o fluxo em uma direção: do computador (ou terminal) ao modem. O uso original parece ser pouco usado hoje em equipamentos modernos (incluindo modems).

Os Pinos DTR e DSR

Exatamente como o RTS e CTS, estes pinos são pareados. Para conexões DTE-DTE provavelmente se cruzam. Existem duas maneiras de usar esses pinos. Uma é usá-los como substitutos para controle fluxo RTS/CTS. O pino DTR é exatamente como o pino RTS, enquanto que o pino DSR se comporta como o pino CTS. O fluxo de controle DTR é o mesmo que o fluxo de controle DTR/DSR porém é somente unidirecional e o pino DSR não é utilizado. Muitos terminais de texto e algumas impressoras usam este tipo de fluxo de controle.

O uso normal de DTR e DSR é o seguinte: Um dispositivo positivando o DTR diz que está ativo e pronto para operar. Para um modem, o significado de um sinal DTR do PC depende de como o modem está configurado. Para enviar um sinal DTR manualmente de um PC usando o

comando que ajusta a velocidade de baud a zero. Negar o DTR é algumas vezes chamado de "desligar" (hanging up) mas isto nem sempre acontece.

4.2.8 CAMINHO DO FLUXO DE DADOS, BUFFERS

Além do buffers de porta serial de 1 byte (no hardware) há ainda um outro buffer. Estes são um grande par de buffers (talvez 8k) na memória principal também conhecidos como buffers de porta serial. Quando um programa aplicativo envia bytes à porta serial, estes são primeiro guardados no buffer de porta serial de transmissão na memória principal. O par consiste num buffer de transmissão e um buffer de recepção para a direção oposta ao fluxo de bytes. O acionador de dispositivos seriais retira digamos 16 bytes deste buffer de transmissão, um byte de cada vez, e coloca-os no buffer de transmissão. Uma vez no buffer de transmissão, não há meio de evitar que sejam transmitidos. Eles são então transmitidos pela porta serial. Quando o acionador de dispositivos (sob ordens do controle de fluxo) pára o fluxo de bytes para fora do computador, o que ele realmente pára é o fluxo de bytes para fora do grande buffer de transmissão na memória principal. Mesmo depois que isto aconteceu e o fluxo serial para fora parou, um programa aplicativo pode continuar enviando bytes para o buffer de transmissão de 8k até que seja preenchido. Quando estiver preenchido, o programa aplicativo não pode enviar mais bytes (uma instrução "write" num bloco de programa em C) e o programa aplicativo temporariamente pára de rodar e espera até que algum espaço de buffer se torne disponível.

Mesmo que este programa pare, o computador não necessariamente para de computar. Ele pode passar a rodar outros processos enquanto espera por uma parada do controle de fluxo.

4.3 UARTs

UARTs (Universal Asynchronous Receiver Transmitter) são pastilhas seriais na placa-mãe. Em computadores mais antigos como muitos 486s, as pastilhas estavam na placa de controle de E/S de disco. A função dos UARTs é converter bytes da barra omnibus dos PCs para uma corrente de bits serial. O cabo que sai da porta serial tem apenas um fio para cada direção de fluxo. A porta serial envia uma corrente de bits, um de cada vez. Por outro lado, a corrente de bits que entra na porta serial pelo cabo externo é convertida para bytes (em paralelo) que o computador pode entender.

Os UARTs tratam dos dados em pedaços com o tamanho de bytes, que é convenientemente também o tamanho dos caracteres ASCII. Digamos que tens um terminal ligado a teu PC.

Quando um caractere é digitado, o terminal passa esse caractere para seu transmissor (também um UART). O transmissor envia aquele byte para a linha serial, um bit de cada vez, a uma velocidade específica. No lado do PC, o UART receptor toma todos os bits e reconstrói o byte (em paralelo) e coloca-o num buffer. Existem dois tipos básicos de UART: UARTs burros e UARTs FIFO. UARTs burros são o 8250, 16450, e os antigos 16550 e 16650.

Para entender as diferenças entre um UART burro e FIFO (Firs In First Out --entra primeiro sai primeiro), vamos antes examinar o que ocorre quando um UART enviou ou recebeu um byte. O próprio UART não pode fazer nada com os dados que passam através do mesmo, pois ele apenas os recebe e os envia. Para os UARTs burros originais, a UCP recebe uma interrupção do dispositivo serial sempre que um byte é enviado ou recebido. A UCP então move o byte recebido do buffer do UART para algum lugar da memória, ou dá ao UART um outro byte para enviar. Os UARTs 8250 e 16450 possuem apenas um buffer de 1 byte. Isso significa que a cada vez que 1 byte é enviado ou recebido, a UCP é interrompida. A baixas velocidades de transferência, isto funciona. Mas, a altas velocidades, a UCP fica tão ocupada em atender ao UART, que não sobra tempo para atender a outras tarefas. Em alguns casos, a UCP não consegue atender às interrupções em tempo, e o byte no buffer é sobrescrito, porque os dados chegam muito rápido. Isto é chamado de "atropelamento" (overrun) ou "transbordamento" (overflow).

Eis onde os UARTs FIFO são úteis. A pastilha FIFO 16550A (ou 16550) vem com buffers FIFO de 16 bytes. Isto significa que ele pode receber até 14 bytes (ou enviar 16 bytes) antes que tenha de interromper a UCP. Não apenas pode a UCP esperar por mais bytes, como também pode transferir todos os 14 (ou mais) bytes a uma só vez. Embora o limiar (nivel de disparo) possa ser fixado em 8 ao invés de 14, isto ainda é uma grande vantagem sobre os outros UARTs, que possuem buffers de 1 byte. A UCP recebe menos interrupções, e fica livre para fazer outras coisas e os dados não são perdidos.

Enquanto a maioria dos PCs possui apenas um 16550 com buffers de 16 bytes, UARTs melhores possuem buffers ainda maiores. Nota que a interrupção é emitida um pouco antes do buffer ficar cheio (a um "nível de disparo" de, digamos, 14 bytes para um buffer de 16). Isto dá lugar para alguns bytes mais a serem recebidos durante o tempo em que a interrupção está a ser atendida. O nível de disparo pode ser definido para vários valores permitidos pelo software do núcleo. Um nível de disparo de 1 funciona quase como um UART burro (exceto

que tem 15 mais bytes após emissão da interrupção. Eis uma lista de UARTs. *ND* é o *N*ível de *D*isparo.

- 8250, 16450, antigo 16550: Obsoletos com buffer de 1 byte
- 16550, 16550A, 16c552: buffer de 16 bytes, ND=1,4,8,14
- 16650: buffer de 32 bytes. Velocidade até 460.8 Kbps
- 16750: buffer de 64 bytes para enviar, 56 bytes para receber. Velocidade até 921.6 Kbps
- Hayes ESP: buffer de 1K-byte

Os obsoletos somente servem para os modems mais lentos que 14,4 kbps (velocidades DTE até 38400 bps). Para modems modernos é necessário pelo menos um 16550 (que não seja antigo). Para modems de 56k V.90, pode ser bastante mais rápido com um 16650. A principal vantagem do 16650 é o seu buffer maior, pois a velocidade extra não é necessária, a menos que a razão de compressão seja alta.

4.4 FORMAS DE ONDA DE TENSÃO

4.4.1 TENSÃO PARA UM BIT

Na porta serial EIA-232, as tensões são bipolares (positivo ou negativo em relação à terra) e devem ser em torno de 12 volts em magnitude (alguns são de 5 a 10 volts). Para os pinos de transmissão e recepção, +12 volts é um bit zero (algumas vezes chamado de "espaço") e -12 volts é um bit um (algumas vezes chamado de "marca"). Isto é conhecido como lógica invertida, pois normalmente um bit zero é ao mesmo tempo falso e negativo, enquanto que um bit um é verdadeiro e positivo. Embora os pinos de transmissão e recepção sejam em lógica invertida, outros pinos (linhas de controle de modem) são em lógica normal (tensão positiva sendo verdadeira, ou "ligado" e negativa sendo falsa, ou "desligada"). Tensão zero não tem significado lógico (usualmente significa que a unidade está desenergizada). Um intervalo de tensões é permitido. As especificações dizem que a magnitude de um sinal transmitido deve ser entre 5 e 15 volts mas não deve nunca exceder 25 volts. Qualquer tensão recebida abaixo de 3 volts é indefinido como sinal (mas alguns terminais aceitam uma tensão menor como válida). Vêem-se algumas vezes declarações errôneas de que a tensão é normalmente 5 volts (ou mesmo 3 volts), mas é usualmente 11 a 12 volts.

4.4.2 SEQUÊNCIA DE TENSÕES PARA UM BYTE

O pino de transmissão (TxD) é mantido a -12 V (marca) em repouso, quando nada está a ser enviado. Para iniciar um byte a tensão salta para +12 V (espaço) para o "start bit" e permanece em +12 V pela duração (período) de um bit. Em seguida vem o bit de baixa ordem do byte de dados. Se for um bit zero, nada muda e alinha mantém-se em +12 V por outro período de um bit. Se for um bit um, a tensão salta de +12 para -12 V. Depois disso vem o bit seguinte, e assim por diante. Finalmente, um bit de paridade pode ser enviado e então um "stop bit" de -12 V (marca). A linha permanece a -12 V (inerte) até o próximo "start bit". Nota que não há retorno para zero volts e portanto não existe um modo simples (exceto por um sinal de sincronização) de saber onde um bit termina e o próximo se inicia no caso em que 2 bits consecutivos são da mesma polaridade (ambos zero ou ambos um). Um segundo bit de parada seria tambem de -12 V, a mesma tensão que o primeiro bit de parada. Como não existe um sinal para marcar os limites entre os bits, o único efeito do segundo bit de parada é que a linha deve permanecer em -12 V durante duas vezes o tempo. O receptor não tem meios de detectar a diferença entre um segundo bit de parada e um tempo mais longo entre bytes. Assim, a comunicação funciona bem se um lado usa um bit de parada e o outro usa dois bits de parada, mas usando apenas um é obviamente mais rápido. Em raros casos 1 1/2 bits de parada são usados. Isto significa que a linha é mantida a -12 V por 1 1/2 períodos de tempo.

4.4.3 PARIDADE

Os caracteres são normalmente transmitidos com 7 ou 8 bits de dados. Um bit de paridade adicional pode ser (ou não) apenso a esses, resultando num comprimento de byte de 7, 8 ou 9 bits. Alguns emuladores de terminais e terminais antigos não permitem 9 bits. Alguns proibem 9 bits se 2 bits de parada forem usado, pois isto aumentaria o número de bits demais: 12 bits total. A paridade pode ser definida como ímpar, par ou inexistente (paridade de marca e espaço e alguns terminais). Com paridade ímpar, o bit de paridade é escolhido de modo que o número de bits num byte, incluindo o de paridade, seja ímpar. Se um desses bytes for corrompido por um bit ter sido alterado, o resultado será um byte ilegal de paridade par. Esse erro será detectado e ser for um byte chegando ao terminal, um símbolo de caractere errado aparece no ecran. Paridade par funciona de modo similar com todos os bytes legais (incluindo o bit de paridade) tendo um número par de bits 1. Durante a inicialização, o número de bits por caractere em geral significa que o número de bits de dados por byte (7 para o ASCII

original e 8 para vários conjuntos de caracteres ISO). Uma "marca" é um bit 1 e um "espaço" é um bit 0. Para paridade de marca, o bit de paridade é sempre um bit um. Para paridade de espaço, é sempre um bit zero. Paridade de marca ou espaço desperdiça largura de banda e deve ser evitada quando viável. "Sem paridade" significa que paridade não é adicionada. Para terminais que não permitem bytes de 9 bits, deve-se selecionar "sem paridade" quando usar o conjunto de caracteres de 8 bits pois não existem vaga para o bit de paridade.

4.4.4 FORMANDO UM BYTE (FRAMING)

Na transmissão serial de bytes via portas EIA-232, o bit de baixa ordem é sempre enviado primeiro. As portas seriais de PCs usam comunicação assíncrona onde existe um bit de partida e um bit de parada para marcar o início e fim de um byte. Isto é chamado "framing" (emolduração ou enquadramento) e o byte enquadrado é as vezes chamado de "frame" (quadro). Como resultado, um total de 9, 10, ou 11 bits por byte são enviados, sendo 10 o mais comum. O termo "8-N-1" significa 8 bits de dados, sem paridade, 1 bit de parada. Isto dá 10 bits total quando se conta o bit de partida. Um bit de parada é quase universalmente usado. A 110 bits/s (e algumas vezes a 300 bits/s) dois bits de parada eram antes usados, porém hoje o segundo bit é usado somente em situações muito incomuns.

4.4.5 ASSÍNCRONO E SINCRONIZADO

A porta serial EIA-232 conforme implementada nos PCs é assíncrona, que efetivamente significa que não existe um sinal de relógio com tiques para marcar quando cada bit é enviado. Existem somente dois estados do fio de transmissão (ou recepção): marca (-12 V) ou espaço (+12 V). Não existe um estado para 0 V. Assim, uma seqüência de bits 1 é transmitida por apenas -12 V constante sem marcadores entre os bits. Para o receptor detectar os bits individuais deve sempre ter um sinal de relógio que esteja em sincronismo com o relógio do transmissor. Tal relógio geraria um tique em sincronismo com cada bit transmitido (ou recebido). Para transmissão assíncrona, o sincronismo é realizado cercando cada byte com um bit de partida e parada (feito por hardware). O receptor escuta na linha por um bit de partida e quando detecta um dispara os tiques de seu relógio. Ele usa este tique do relógio para medir o tempo dos próximos 7, 8 ou 9 bits. Na realidade é um pouco mais complexo do que isto, pois várias amostras de um bit são freqüentemente tomadas e isto requer tiques adicionais. Então o bit de parada é lido, o relógio pára e o receptor espera pelo próximo bit de partida. Assim,

existe sincronismo durante a recepção de um byte único, mas não existe sincronismo entre um byte e o próximo.

CAPÍTULO 5

RECEPTOR ORBISAT ORB 2002 RLP

Na primeira fase do projeto, os algoritmos dos métodos propostos foram modelados e implementados em linguagem FORTRAN 90 (Compilador da Microsoft). Em tais algoritmos aparecem operações com matrizes cujas subrotinas estão disponíveis nas bibliotecas MSIMSL e MSFLIB, acessíveis somente por codificação em Fortran 90. Essas subrotinas não são compatíveis com outras versões do FORTRAN. Os resultados mostraram comparações entre os três métodos matemáticos descritos nesse trabalho, onde simulações foram realizadas com a introdução dos dados de efeméride para que a solução pudesse ser efetuada.

Com o sucesso dessa fase, desejou-se que os dados não fossem mais simulados, e sim que eles fossem recebidos por um receptor GPS. Para isso foi adquirido, um receptor nacional da marca Orbisat, modelo ORB 2002 RLP.

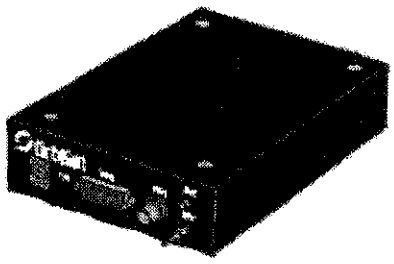


Figura 6– Receptor ORBISAT ORB 2002 RPL

O ORB 2002 RLP é uma solução para o receptor de Sistema de Posicionamento Global, cujo diagrama em blocos é mostrado na Figura 7.

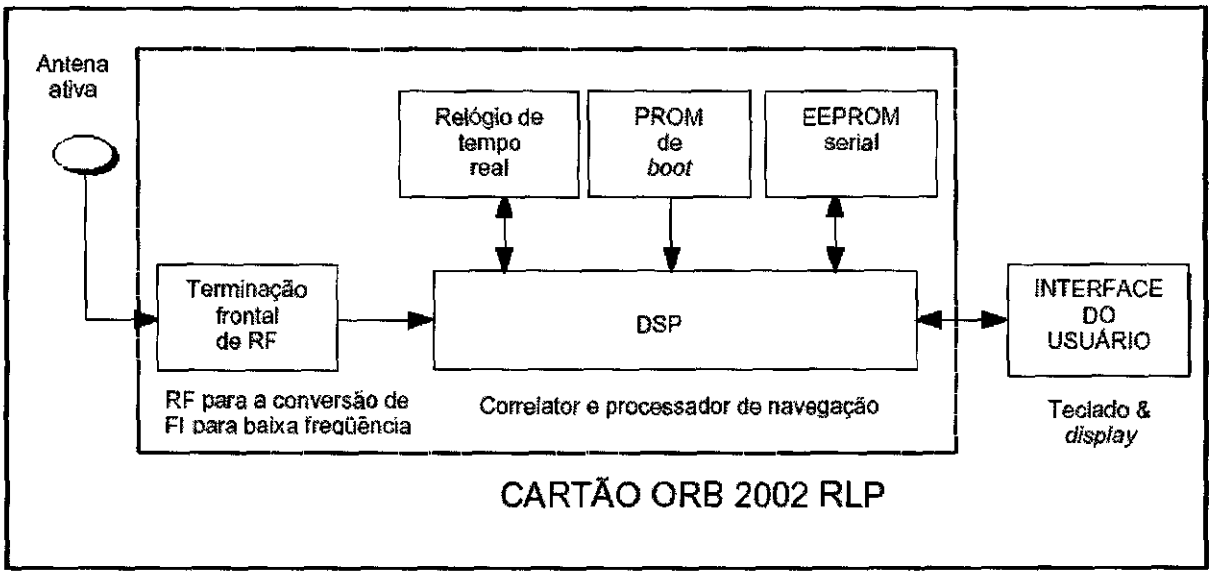


Figura 7 - Diagrama em blocos do receptor ORB 2002 RLP

O ORB 2002 RLP foi projetado para adquirir e rastrear simultaneamente sinais de 12 satélites GPS e calcular a posição, a velocidade e o tempo precisos do usuário. A Orbisat oferece um programa de interface gráfica que roda em um IBM-PC, o qual pode ser conectado ao receptor de GPS através de uma ligação RS 232. O receptor ORB 2002 RLP consiste de (vide Figura 9):

- - Terminação frontal de RF com antena
- - Correlator
- - Módulo de navegação
- - Interface do usuário

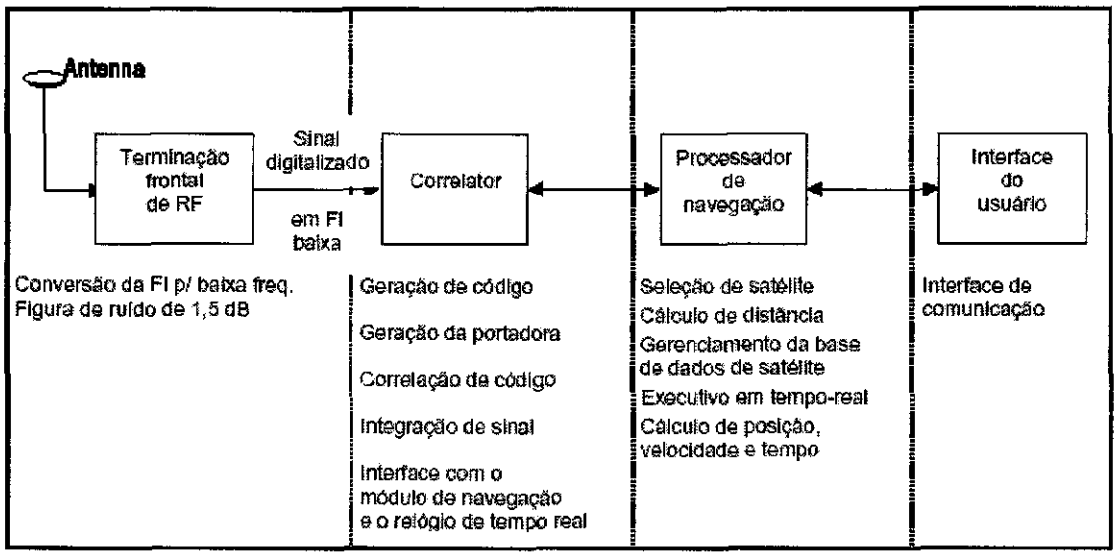


Figura 8 - As quatro seções do receptor ORB 2002 RLP.

O sinal GPS L1, recebido através de uma antena ativa, sofre uma conversão para uma FI de baixa freqüência na terminação frontal de RF – projetada em torno de um conversor para baixa freqüência VLSI. Este sinal de FI é amostrado a uma taxa pré-definida e então quantificado em um conversor A/D no interior da terminação frontal de RF. As amostras quantificadas do sinal de FI são recebidas serialmente pelo processador de sinais digitais (DSP) e armazenados em um *buffer* interno. A conversão para baixa freqüência deste sinal de FI armazenado para banda base, a correlação do sinal com o código local, as malhas de rastreamento de portadora e código, o cálculo da posição, da velocidade e do tempo do usuário, bem como a interface de comunicação com um PC hospedeiro (PC/*host*), são todos implementados no *software* operando no DSP.

Antena: A seção da antena consiste de:

- - uma antena operando em 1575,42 MHz (sinal L1)
- - um amplificador de baixo ruído (LNA) interno

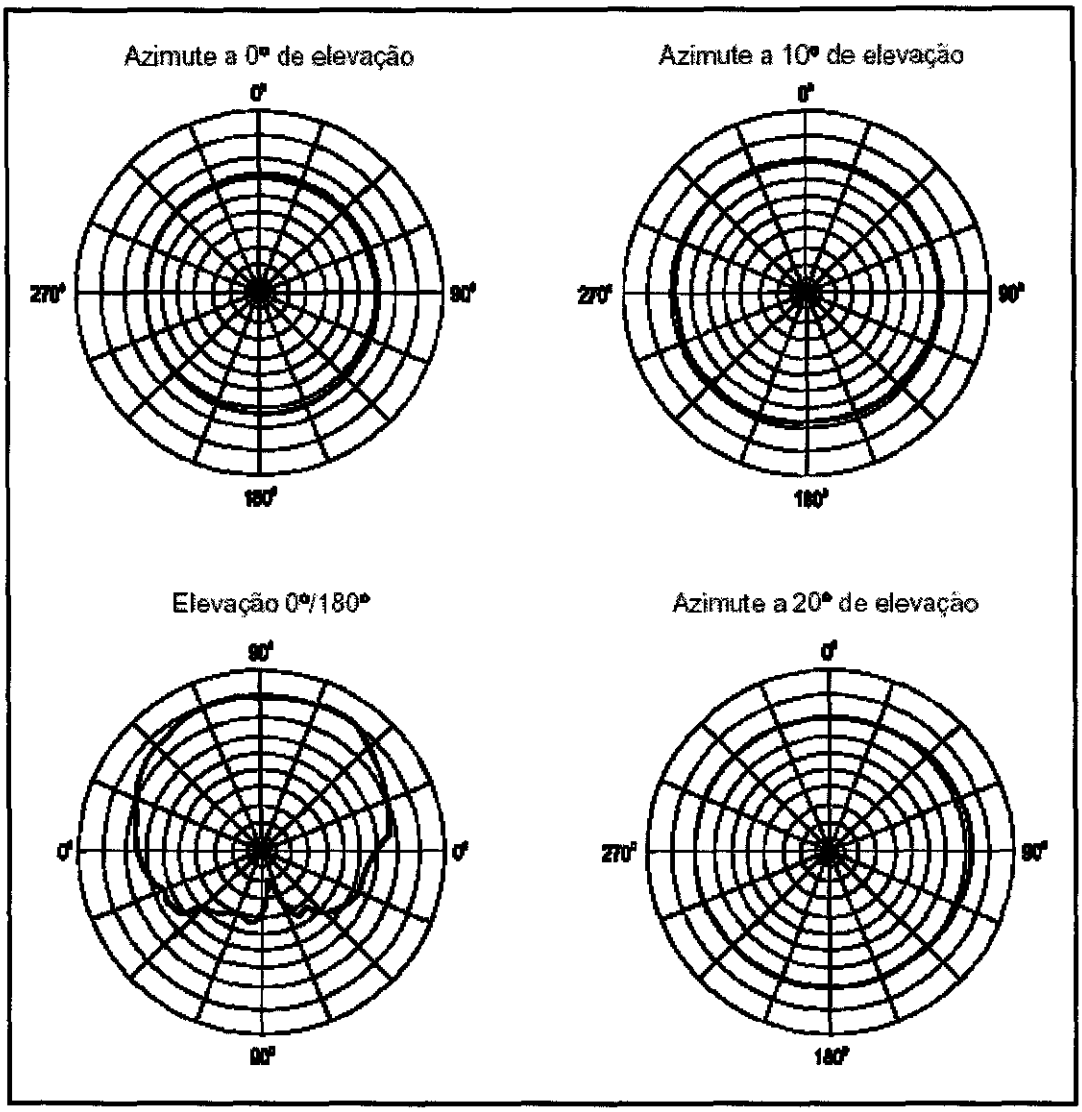


Figura 9 - Diagramas de irradiação da antena GPS.

A antena recebe o sinal L1 de todos os satélites visíveis com quase o mesmo ganho através de diagramas de irradiação hemisféricos (vide Figura 9).

O sinal de satélite na antena é bastante fraco (-130 dBm) com uma relação sinal/ruído (SNR) menor que -16 dB. O amplificador de baixo ruído interno compensa a perda de sinal no cabo de RF. As principais funções da seção da antena GPS são:

- - Recepção do sinal GPS L1 a 1575,42 MHz
- - Amplificação dos sinais de satélite recebidos

Terminação frontal de RF: A terminação frontal de RF foi projetada em torno de um conversor para baixa freqüência de RF VLSI disponível comercialmente e de um conjunto de componentes passivos de RF. Ele recebe o sinal de entrada da antena e fornece ao correlator

um sinal de FI digitalizado convertido para baixa freqüência. A terminação frontal de RF consiste dos seguintes componentes principais:

- - Amplificador de baixo ruído
- - Filtro passa-faixa
- - Conversor para baixa freqüência VLSI
- - Filtro LC
- - Filtro SAW
- - Relógio de referência

O sinal L1 de satélite GPS passa inicialmente por um amplificador de baixo ruído e, em seguida, por um filtro passa-faixa. Na seqüência, é convertido para uma FI de baixa freqüência pelo conversor VLSI.

A conversão para baixa freqüência do sinal L1 é realizada em três estágios misturadores para gerar as freqüências de FI. As freqüências do oscilador local necessárias para as operações de mistura são geradas em um sintetizador de freqüências que tem como uma de suas entradas o *clock* de referência. Cada estágio do misturador possui, na seqüência, um filtro passa-faixa e um amplificador.

A saída final da terminação frontal de RF consiste de um sinal de FI digitalizado de baixa freqüência a ser processado no próximo estágio, ou seja, no correlator. A terminação frontal de RF também fornece interfaces de sinal para o BITE (autoteste), sinal de *clock* para o correlator e nível de AGC.

As funções principais da terminação frontal de RF são:

- - Recepção do sinal GPS na banda L da antena
- - Amplificação e filtragem do sinal de RF
- - Conversão para baixa freqüência do sinal de RF para o sinal de FI
- - Digitalização do sinal de saída de FI em baixa freqüência

Correlator: O correlator recebe da terminação frontal de RF a saída de FI digitalizada em baixa freqüência e realiza medidas do sinal de satélite. As saídas das medidas são enviadas ao módulo de navegação que computa a posição, a velocidade e o tempo do usuário. O correlator consiste de:

- - *Buffers* para armazenar a FI de baixa freqüência amostrada
- - Gerador da portadora
- - Gerador de código

As principais funções do correlator são:

- - Recepção e armazenamento em *buffers* do sinal de FI de baixa freqüência digitalizado
- - Conversão para baixa freqüência do sinal de FI para banda base
- - Processamento paralelo de 12 canais
- - Geração da portadora local
- - Geração dos códigos C/A locais para 32 satélites
- - Correlação do sinal recebido com o código local
- - Interface com o módulo de navegação

A solução de *software* para o correlator fornece algoritmos maiseficientes e flexíveis.

Módulo de navegação: O módulo de navegação recebe medidas realizadas pelo correlator a partir do sinal de satélite. As medidas de pseudodistância e distância delta rastreadas pelo correlator são utilizadas pelo módulo de navegação para estimar a posição do usuário e outros parâmetros.

O módulo de navegação consiste dos seguintes módulos:

- - Gerenciamento da base de dados do satélite
- - Seleção de satélite
- - Gerenciamento de canal
- - Solução de posição, velocidade e tempo
- - Interface com o correlator e a interface do usuário
- - Execução de multitarefas em tempo real
- - Processador de dados medidos
- - Interface com o relógio de tempo real e a memória não volátil
- - Aplicação de correções de DGPS
- - Gerador de tempo preciso
- - Suporte de dados

- - Interface RS 232

O algoritmo de cálculo de posição é tal que quando a posição é determinada com 3 satélites, a VDOP é 1,0.

Interface do usuário: O *software* de interface do usuário fornecido com o receptor opera num IBM-PC conectado ao receptor através de uma ligação RS 232. Utilizando o *software*, você pode:

- Monitorar satélites que estão sendo procurados e rastreados
- Mostrar a posição, a velocidade e o tempo do usuário
- Alterar os parâmetros da porta DGPS
- Realizar o *login* de várias mensagens NMEA
- Alterar os ajustes de máscara de elevação
- Ajustar o receptor para os parâmetros de fábrica
- Iniciar o receptor a "frio"
- Configurar o receptor para o modo *power down*
- Alterar os ajustes de várias portas

Armazenagem na EEPROM: A EEPROM é usada para armazenar periodicamente parâmetros diversos e de navegação enquanto o receptor encontrar-se operacional. Na próxima vez em que o receptor for ligado, os dados armazenados na EEPROM serão carregados no receptor.

A armazenagem desses parâmetros pode levar um certo tempo, dependendo da quantidade de dados. Se a alimentação principal for desligada durante a transferência desses dados, a armazenagem pode não se concretizar na EEPROM.
Por exemplo, os dados de almanaque são armazenados na EEPROM quando o receptor obtiver dados que estão com um mês de diferença daqueles armazenados na EEPROM.

O tempo de armazenagem para isso pode levar cerca de 20 segundos. Os dados de efeméride são armazenados sempre que o receptor obtiver nova efeméride. Isso pode levar cerca de 1,5 segundo por satélite. Os parâmetros armazenados são:

- - Efemérides para 12 satélites, almanaque
- - Posição do usuário, dados de ID (identificação)

- - Periodicidade de todas as mensagens NMEA
- - Ajustes de comunicação para as portas do *host* e do DGPS
- - Modo DGPS
- - Ângulo de máscara, limite HDOP
- - Taxa de atualização de fixação
- - Configuração PTTI
- - Manter altitude, manter posição
- - Filtro de posição, filtro de velocidade, solicitação de extrapolação

O receptor não recupera os seguintes parâmetros:

- - Modo de medida crua
- - Modo *power down*

Memória *flash:* A memória *flash* mantém o código GPS e proporciona espaço para armazenagem de dados. O receptor dispõe de uma memória *flash* de 512 kbytes, na qual o usuário pode armazenar seus dados em 192 kbytes. A memória *flash* é diferente da EEPROM no modo pelo qual armazena dados. Na memória *flash*, os dados não podem ser sobrescritos antes de se apagar um determinado espaço de memória.

A organização da memória *flash* é apresentada abaixo:

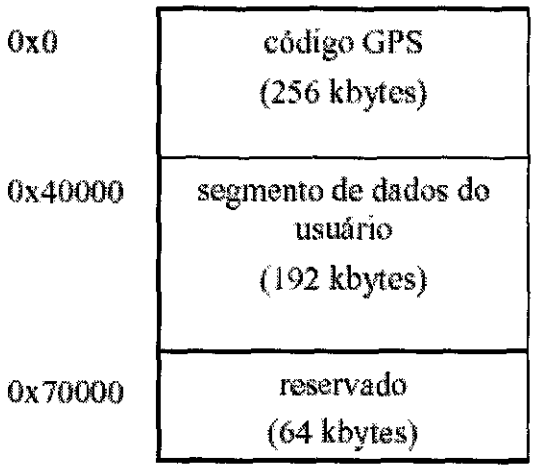


Figura 10: Organização da memória Flash

A versão atual do ORB 2002 RLP suporta somente a memória *flash* SST 39F040.

Nas mensagens *flash*, quando o usuário especifica um endereço 00000, este corresponde ao endereço 0x40000 na memória *flash*. Portanto, o endereço especificado as mensagens *flash* é

um endereço relativo. Para sobrescrever dados em um determinado local, o segmento de dados do usuário deve ser apagado totalmente utilizando a mensagem *flash* de pagar. Quando o apagamento *flash* é realizado, o segmento de dados do usuário é apagado e o receptor sofre um *reset* de *software*.

Exemplo: Para a mensagem de escrita *flash* ACFLW,000000,03,12,AB,34*<CS><CR><LF>, os dados 12,AB,34 são armazenados no local de memória 0x40000 e quando os dados são lidos pela mensagem de leitura *flash* $ACFLR,000000,03*<CS><CR><LF>, os dados serão lidos do local de memória 0x40000 e apresentados ao usuário numa mensagem $ACFLD.

CAPÍTULO 6

PROGRAMAÇÃO DETALHADA

O que primeiro se fez foi o planejamento da maneira de que o software de transmissão e recepção em tempo real seria programado. Para isso estudaram-se as arquiteturas, tanto física quanto de software, pois seriam necessários saber quais as limitações físicas e virtuais do projeto.

6.1 – ARQUITETURA FÍSICA

Conforme o esquema abaixo, a recepção de dados é realizada através de equipamentos montados e conectados a um computador. A antena acoplada junto ao receptor Orbisat é conectada pela porta serial COM1 através de um cabo RS232.

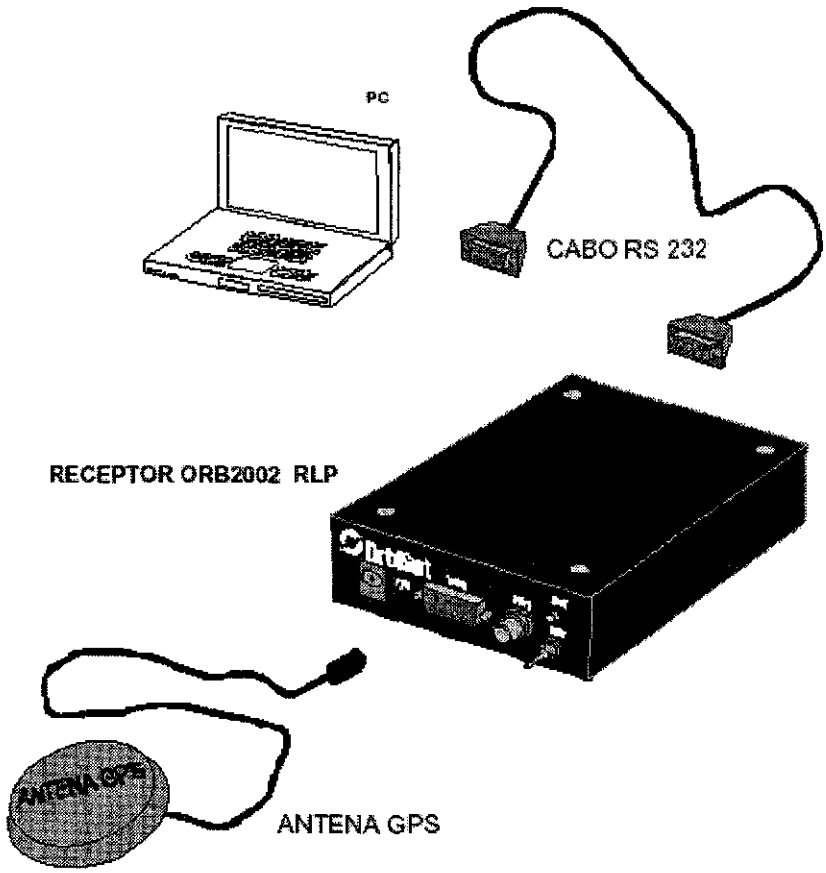


Figura 11 – Arquitetura Física

Existem diversas mensagens em diferentes formatos que o receptor ORBISAT pode enviar a partir de uma mensagem de comando enviada a este receptor pela porta serial. A diferença deste receptor em relação aos outros tipos existentes no mercado é que ele é capaz de fornecer dados brutos dos satélites, isto é, aqueles dados que não foram equacionados e que, portanto, possam ser utilizados pelo usuário para diferentes finalidades (no nosso caso, o equacionamento sob diferentes algoritmos). Na realidade o que ocorre nos receptores convencionais é que a solução é dada para o usuário, porém o aparelho funciona como uma caixa preta, sem que tenhamos acesso a solução numérica e o equacionamento dessa solução. Como temos a intenção de que no futuro do trabalho a solução gerada deva sofrer alterações a fim de melhorar a solução, necessitamos ter acesso aos métodos e algoritmos de solução de navegação.

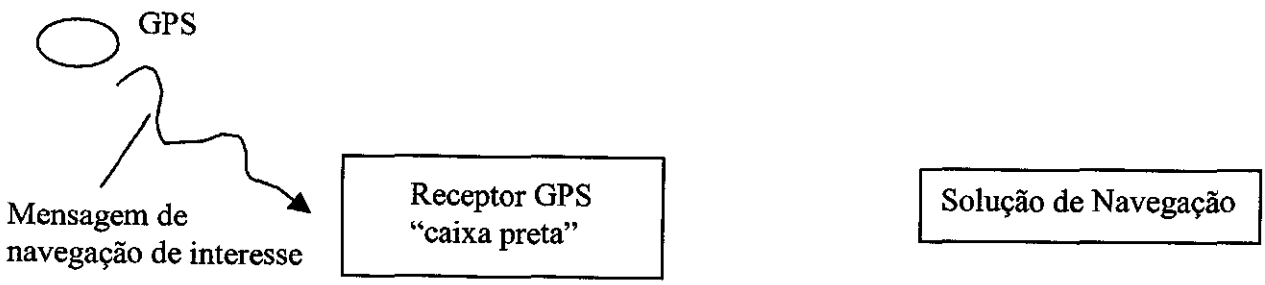


Figura 12:Esquema do receptor GPS: "caixa preta"

Para a análise estamos interessados em duas das mensagens do receptor, as quais ele envia no formato binário (customizado) através da porta serial, são elas: dados de mensagem sem tratamento (dados crus) e dados crus de efeméride.

DADOS DE MENSAGEM SEM TRATAMENTO OU CRUS

Esta mensagem é utilizada para enviar as medidas cruas e a efeméride ao *host*. O modo de dados crus pode ser ativado utilizando-se a mensagem $ACMCC com B como parâmetro. Se o modo de dados crus for habilitado, dados crus de medida serão transmitidos por *default*. Esta mensagem possui o período fixo de 1 segundo.

3F3F AF07 UUUU WWWW SSSSSSSS NNNNNNNN R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 CS LF

Descrição fornecida pelo fabricante:
3F3F ? Cabeçalho da mensagem
AF07 ? Identificação da mensagem
UUUU ? Reservado
WWWW ? 2 bytes do número da semana do GPS com MSB transmitido primeiro.
SSSSSSSS ? 4 bytes de contagem de segundos a partir do início da atual semana do GPS, com MSB transmitido primeiro. A faixa deste campo é de 0 a 604799.
NNNNNNNN ? 4 bytes do número de nanosegundos dentro de um segundo, com MSB transmitido primeiro. O tempo total é obtido adicionando-se o tempo em segundos obtido no campo anterior com o campo nanosegundos. A faixa deste campo é de 0 a 999999999.
R1....R12 ? Medidas cruas para todos os canais. Ri (i = 1 a 12) possui o seguinte formato:
VV SS PPPPPPPPPPP DDDDDDDD AA UU
VV ? 1 byte de validade da palavra transmitida com MSB primeiro. (0x80 indica "medida válida", 0x00 indica "medida inválida")
SS ? 1 byte de PRN transmitido com MSB primeiro.
PPPPPPPPPPP ? 6 bytes de pseudodistância em metros.
O número é representado utilizando-se o seguinte formato:

sinal * (1.mantissa) * 2expoente

O bit MS do primeiro byte é o bit de sinal ("0" significa um número positivo, "1" significa um número negativo). Os próximos 7 bits do primeiro byte não são utilizados. O segundo byte contém o expoente para o número. O expoente é no formato "excede 127" (expoente é acrescido de 127).

Os 4 bytes restantes contêm a mantissa de 32 bits. Assume-se um "1" implícito na frente da mantissa.

DDDDDDDD ? 4 bytes de distância delta em m/s. Os primeiros 2 bytes contêm o componente inteiro. Os próximos 2 bytes contêm a componente fracionária. Um número positivo é representado por uma parte inteira positiva e uma parte fracionária sem sinal. Um número negativo é representado como o complemento de 2 (o complemento de 2 é realizado no número de 32 bits incluindo ambos os componentes inteiro e fracionário).

AA ? representa a relação sinal/ruído da portadora do sinal do satélite
UU ? 1 byte reservado
CS ?
XORing todos os bytes começando pelo cabeçalho da mensagem (3F3F) até o byte que precede o *check sum*. Calcula *check sum*.
LF ? "mudar de linha" (0x0A)

DADOS CRUS DE EFEMÉRIDE

Esta mensagem pode ser ativada utilizando-se a mensagem $ACEPH Esta mensagem será gerada sempre que o receptor coletar nova efeméride para um satélite. O comando é efetivo somente se o modo de dados crus estiver ativado.

Ao receber esta mensagem, os dados de efeméride serão enviados aos satélites para os quais o receptor tem efeméride. Em um segundo, os dados de efeméride serão enviados para qualquer satélite. Os dados para os outros satélites serão enviados nos segundos subseqüentes, um após o outro.

Por exemplo, se há 10 satélites com efemérides, os dados de efemérides serão enviados durante 10 segundos. A solicitação de PRNs dos SV depende da disponibilidade de efeméride.

3F3F AF08 UUPP ZYXX WWWW E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12
E13 E14 E15 E16 E17 E18 E19 E20 E21 CS LF

Descrição fornecida pelo fabricante:
3F3F Cabeçalho da mensagem
AF08 ID da mensagem
UU Não utilizado (reservado)
PP PRN do SVZ Os dois bits mais significativos são reservados O próximo bit é o bit de validade. A totalidade dos dados de efeméride são válidos somente se este bit for 1. O bit menos significativo é a *flag* do intervalo de ajuste, que indica o intervalo de ajuste da curva utilizada pelo segmento de controle para determinar os parâmetros da efeméride conforme a seguir:
0 = 4 horas
1 = maior que 4 horas
Y Precisão da distância do usuário (URA)
XX Conteúdo dos dados de efeméride (IODE)
WWWW Semana de referência da efeméride
Parâmetros da efeméride (E1 a E21)
Cada parâmetro da efeméride possui 6 bytes.
O número é representado utilizando-se o seguinte formato:
Sinal * (1.mantissa) * 2expoente
O bit MS do primeiro byte é o bit de sinal ("0" significa um número positivo, "1" significa um número negativo). Os próximos 7 bits do primeiro byte não são utilizados.
O segundo byte contém o expoente para o número. O expoente é no formato "excede 127" (expoente é acrescido de 127). Os 4 bytes restantes contêm a mantissa de 32 bits. Assume-se um "1." implícito na frente da mantissa.

E1 Tempo dos dados de *clock* (segundos)
E2 Instante de referência da efeméride (segundos)
E3 Correção do atraso de grupo (Tgd, em segundos)
E4 *Offset* do *clock* (segundos)

E5 Deriva do *clock* (segundo/segundo)

E6 Taxa de deriva do *clock* (segundo/segundo2)

E7 Crs (rad)

E8 Crc (rad)

E9 Cus (rad)

E10 Cuc (rad)

E11 Cis (rad)

E12 Cic (rad)

E13 Diferença média de deslocamento (rad/s)

E14 Anomalia média no instante de referência (rad)

E15 Excentricidade

E16 Raiz quadrada do semi-eixo maior (m^1/2)

E17 Longitude do nodo ascendente *(right ascension)* no instante de referência (rad)

E18 Ângulo de inclinação no instante de referência (rad)

E19 Argumento do perigeu (rad)

E20 Taxa de longitude do nodo ascendente (rad/s)

E21 Taxa do ângulo de inclinação (rad/s)

CS *Check sum (byte wise)*

LF "Mudar de linha"

A adoção do modo de dados crus é possível quando o receptor está no modo normal com taxa de atualização de fixação de 1 segundo. Quando os dados de efeméride estiverem sendo enviados, a mensagem $GPGSV não será enviada, ou seja, haverá descontinuidade na mensagem GSV (se ela estiver ativada).

6.2 – AMBIENTE DE SOFTWARE

O programa de comunicação serial foi desenvolvido em ambiente de linguagem C++ primeiramente no Microsoft Visual Studio 6 e depois refeito no Borland C++ Builder 6.0. O Microsoft Visual Studio foi escolhido primeiramente a fim de que todo o trabalho de programação de linguagem de alto nível realizado na etapa anterior pudesse ser aproveitado. Na realidade, a vantagem de se utilizar esse compilador é o fato de que durante a compilação, todas as extensões de arquivo (tanto F90 como C++) são reconhecidas e compiladas ao mesmo tempo.

Antes de qualquer operação, o software tem comandos que estabelecem a comunicação com a porta serial, como a velocidade (9600 bauds), o número de paridade (None), o número de stop bits (2 stop bits), número de dados (8 dados) e em qual porta será utilizada para a comunicação (COM 1 ou COM 2).

6.3 – ARQUITETURA DE SOFTWARE

O driver de comunicação desenvolvido tem como idéia base à recepção de dados crus dos satélites de GPS a partir do receptor ORBISAT. Isso pode ser feito a partir de uma mensagem enviada ao receptor pelo software de comunicação com o intuito de comandá-lo. Essa mensagem tem um formato padrão, fornecido pelo fabricante do receptor, padrão que varia de fabricante para fabricante. Daí por diante somente as mensagens de interesse serão enviadas pela porta serial.

Uma vez enviadas as mensagens pelo receptor, temos que recebê-las e interpretá-las conforme a estrutura descrita pelo fabricante. Pelo estudo da estrutura das mensagens verifica-se que as mesmas podem ter um tamanho máximo; também que cada mensagem possui um cabeçalho específico e que no final da mesma existe um argumento chamado CHECK SUN (CS) que nada mais é do que um argumento onde é possível verificar se todos os dados da mensagem foram corretamente recebidos (o número de bytes enviado pelo receptor é conferido).

MENSAGEM	TAMANHO MÁXIMO EM BYTES
Dados crus AF07	186
Efeméride AF08	138

Tamanho máximo de cada mensagem recebida

Em verdade, o que o software faz é uma espécie de filtragem que consiste em receber tudo o que chega pela porta serial e comparar com o cabeçalho da mensagem. Uma vez que o cabeçalho corresponde ao esperado, o software dá seqüência ao programa, que recebe armazenando os argumentos da mensagem. Observe que este filtro está sujeito a erros, uma vez que o cabeçalho pode, por uma coincidência bem remota, estar localizado no meio de uma mensagem não desejada pelo usuário.

Os argumentos da string são armazenados pelo software de maneira que já estejam separados em 'blocos', onde cada um desses blocos armazena uma informação de interesse do usuário. Nota-se pela análise da estrutura das mensagens recebidas que as dimensões desses blocos são diferentes e, além disso, as maneiras de conversão para números decimais de cada um desses blocos binários é feita seguindo regras diferentes fornecidas pelo fabricante.

Os dados são enviados de byte em byte pelo receptor e um problema que surge é que eles somente podem ser lidos na forma decimal, um byte por vez. Quando o dado possui uma maior faixa (mais de um byte), não conseguimos ter o seu equivalente decimal de maneira direta. Como solução, propus a conversão desses argumentos do tipo char (1byte) para oito valores do tipo int (2 bytes), simplesmente convertendo decimal em binário, com o propósito de formar um string equivalente de números binários. A partir destes números inteiros (que na realidade são binários), faz-se a conversão desejada conforme manda a especificação do fabricante.

A solução dada foi desenvolver um algoritmo para cada tipo deste aglomerado de dados. Assim, foi possível comparar quatro tipos de dados a serem interpretados, portanto desenvolveu-se quatro algoritmos diferentes.

Os dados foram classificados da seguinte maneira: dados de 2 bytes, 4 bytes, tipo mantissa e tipo fracionário. Os dados de dois e de quatro bytes têm uma conversão idêntica, porém, por simplificação e opção do programador foram separados nas suas dimensões, obtendo-se uma maior agilidade do programa. Os dados de um único byte são automaticamente convertidos em decimal, por esse fato nenhuma conversão posterior fora adotada.

Cada um dos algoritmos converte um byte em oito números inteiros (bits) além de fazer a conversão desejada em decimal seguindo a regra de 2 bytes, 4 bytes, mantissa ou fracionário, conforme detalhado na mensagem padrão do fabricante.

Com os dados dos satélites já convertidos em decimal podemos pensar em partir para a solução do problema. A partir dos dados até então obtidos uma nova rotina é rodada a qual converterá esses dados crus e dados de efeméride em coordenadas X, Y, Z e pseudodistâncias dos satélites, dados esses necessários para efetuar as soluções de navegação conforme os cálculos preparados em ambiente FORTRAN na fase anterior do projeto.

Um detalhe a ser comentado é que as rotinas em Fortran sofreram algumas alterações para serem compatíveis ao compilador da Compac, já que na versão anterior das rotinas existiam comandos ligados a algumas bibliotecas pertencentes à versão F90 do Fortran, comandos esses incompatíveis com o Visual Studio 6.0.

Uma vez com os dados estruturados, esses podem ser injetados nos algoritmos da solução de navegação, obtendo-se as coordenadas X, Y, Z e o tempo do usuário.

Uma visão mais geral do algoritmo do software pode ser visualizada no seguinte fluxograma:

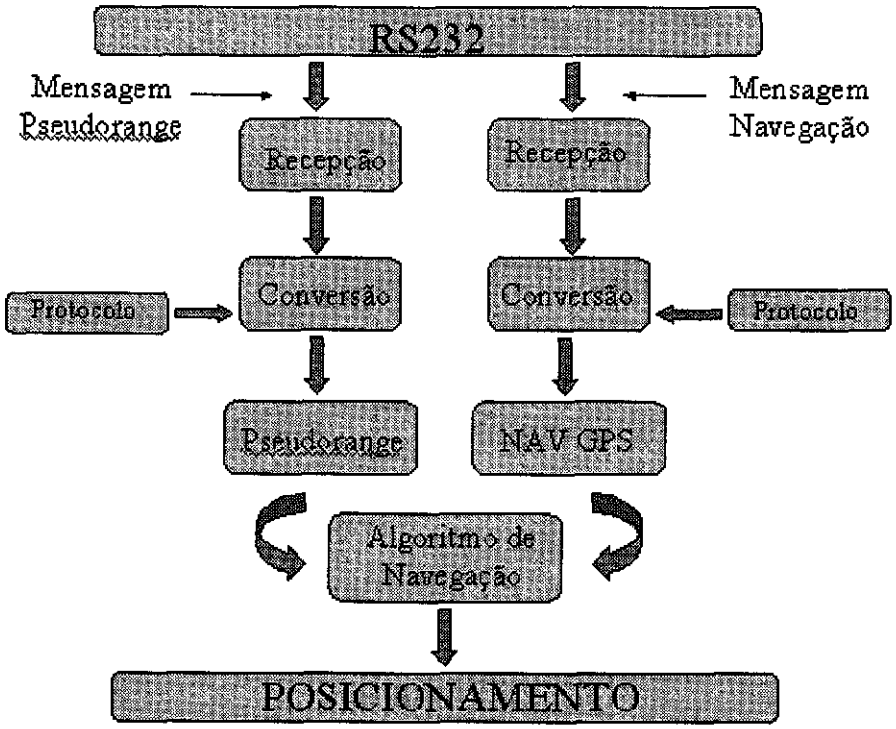


Figura 13- Fluxograma do Algoritmo

CAPÍTULO 7

STATUS ATUAL E RESULTADOS

7.1 CODIFICAÇÃO DO SOFTWARE

O software se inicia com o set up da comunicação serial, onde alguns parâmetros são ajustados de acordo com o tipo de comunicação que desejamos realizar, isto é, de acordo com as configurações pré-estipuladas pelo fabricante do equipamento e de qual porta estamos utilizando para a comunicação.

```
_outp( COM1 + 1 , 0x00 );   // Turn off interrupts - COM1
_outp( COM1 + 3 , 0x80 );    // Linha de controle do registrador - Set DLAB ON
_outp( COM1 + 0 , 0x0C );    // Divisor latch low byte
_outp( COM1 + 1 , 0x00 );    // Divisor latch high byte
_outp( COM1 + 3 , 0x03 );    // 8 Bits, No Parity, 1 Stop Bit
_outp( COM1 + 2 , 0xC7 );   // FIFO Control Register
_outp( COM1 + 4 , 0x0B );   // Turn on DTR, RTS, and OUT2
```

Figura 13: Codificação dos parâmetros da comunicação serial

Foram realizados testes para conferência destes parâmetros a fim de verificar se realmente estaríamos tendo uma comunicação serial. Para isso adaptamos um software que imprimiria na tela qualquer caractere captado pela porta serial. Após essa verificação, ilustrada na figura

abaixo, partimos para a fase de envio de comandos pela porta. Sabíamos que a partir de alguns comandos (mensagens) o receptor alteraria as mensagens que estavam sendo enviadas. A partir da verificação visual obtemos a certeza de que o set up e a comunicação serial estavam funcionando corretamente.

??¯¨ ¯ ö· »'„ï€—„&!3ö›Žl(€˜ —C$M»÷Ý%š.€Ł—g¸ö°úµùŠ.€ž —dQ–'öÌ¡7-Œ
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44
??¯¨ ¯ ö· »';...˘ D€—„#¥éöšúE)€˜ —C#`÷Ü¦+.€Ł—g°Ÿ-úµlg.€ž —dOJ° öÌ– †-É
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44
??¯¨ ¯ ö· »";... »€—„ û1öšƒ¯)€˜ —C!³V÷ÜL5.€Ł—g¼Ł $ú´ù›.€ž —dLÐöË¢í-Î
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44
??¯¨ ¯ ö· »";...0Þ€—„- 0ö™Ô²)€˜ —C x´÷Û±˜.€Ł—g½®4ú´KÅ.€ž —dJ†Ł öËÉ-|
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44
??¯¨ ¯ ö· »•;...Fô€—„ þ¨ö™]c)€˜ —C?b÷ÛU' .€Ł—g¿H*ú³Ú±.€ž —dH;'öÊ'6-˘
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44
??¯¨ ¯ ö· »–;...\¬€—„ dŁ ö˜Ú½)€˜ —C áä÷Úåþ.€Ł—gÀÅjú³Uµ.€ž —dEÊëöÊ µ,•
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44
??¯¨ ¯ ö· »—;...rÓ€—„ â~ö˜Y@)€˜ —C ªy÷ÚuÁ.€Ł—gÂd ú²Õa.€ž —dC,öÉ™[-ç
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44
??¯¨ ¯ ö· »˜;...ˆ›€—„– Kèö—Êª)€˜ —C WÁ÷Ùôs/€Ł—gÃæ¨ú²B*.€ž —dA —öÉ--g
$ACSTM,AO,0,AS,0,RC,0,EP,0,RF,0,TS,1,*44

Figura 14: Ilustração dos dados recebidos na comunicação serial.

Os dados que aparecem na tela são os caracteres capturados pela porta serial. Esses dados estão codificados num código chamado ASCII, podendo ser interpretados, como caracteres binários, de um byte a cada caractere mostrado na codificação ASCII. A tabela de caracteres ASCII contém os valores decimais e hexadecimais dos padrões de caracteres estendidos ASCII (American Standards Committee for Information Interchange). O padrão de caracteres estendido inclui o set de caracteres ASCII e 128 outros caracteres para gráficos e desenhos lineares, freqüentemente chamados de "IBM set de caracteres". Segue abaixo uma lista com a codificação ASCII e suas correspondências em decimal e hexadecimal.

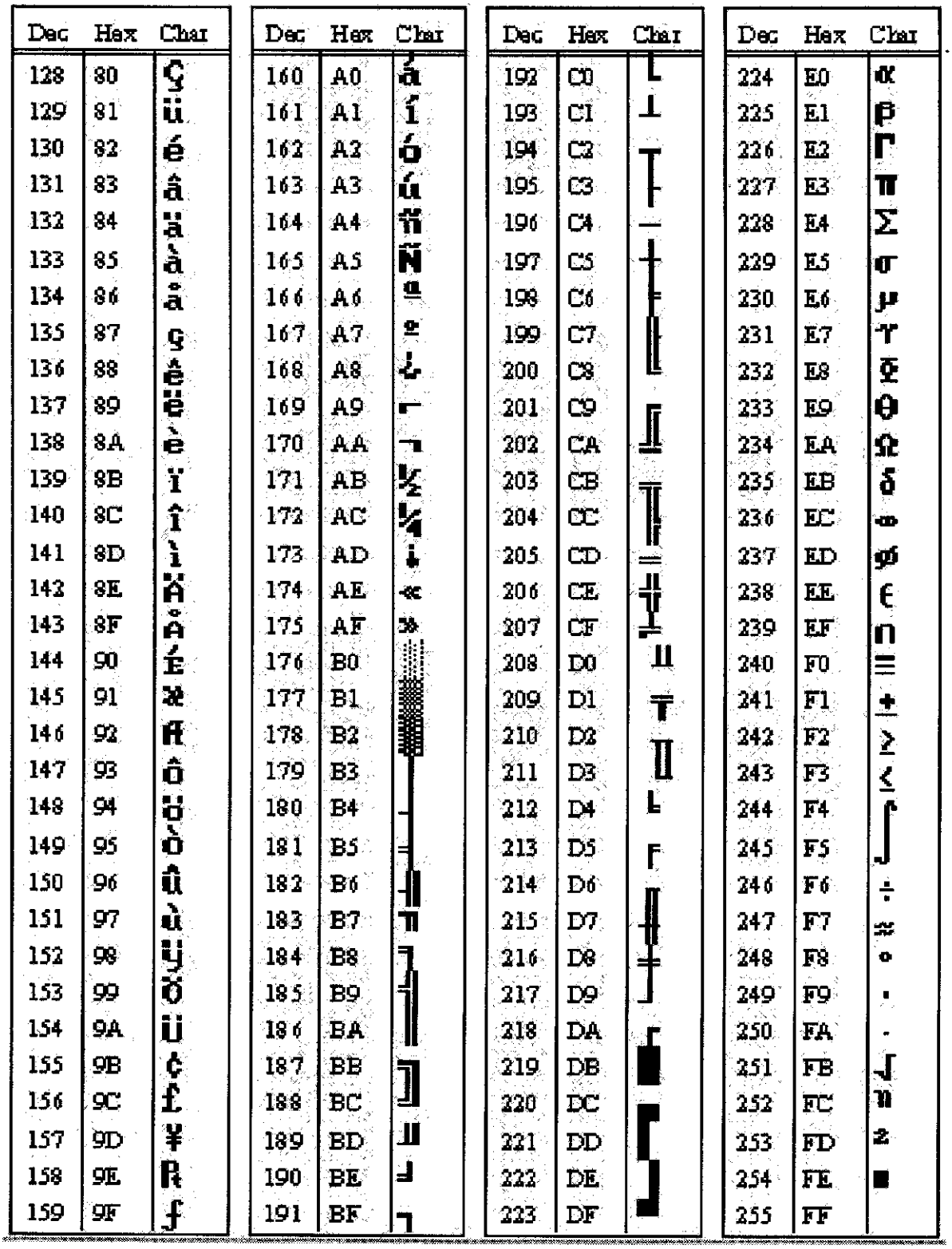
Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	STX
^C	3	03	♥	ETX
^D	4	04	♦	EOT
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HT
^J	10	0A	◙	LF
^K	11	0B	♂	VT
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	DLE
^Q	17	11	◄	DC1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^T	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	ETB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SUB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	Sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

Figura 15: lista de caracteres ASCII.



Dec	Hex	Char
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ

Dec	Hex	Char
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	▒
177	B1	▒
178	B2	▓
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

Dec	Hex	Char
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	▒
220	DC	▒
221	DD	▒
222	DE	▒
223	DF	▒

Dec	Hex	Char
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	φ
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	ⁿ
253	FD	²
254	FE	■
255	FF	

Figura 16: lista de caracteres ASCII. (cont.)

Esses strings de caracteres devem ser decifrados de acordo com o protocolo de mensagem fornecido pelo fabricante. É no protocolo que se classifica o tamanho do dado e seu formato

dentro da string. Por exemplo, existem dados de 2 bytes, 4 bytes, em forma de mantissa e também na forma fracionária. (Ver capítulo 6.1).

Um número binário de mais de um byte, ou em formato mantissa ou fracionário, de acordo com o protocolo do fabricante do receptor não poder ser convertido em decimal de forma direta. Devido a isso foram formulados os subrotinas que serão responsáveis pela conversão de cada uma das maneiras descritas pelo protocolo. Pelo pequeno número de diversidade de maneiras de conversão de dados, foram feitas quatro subrotinas independentes, simplificando o trabalho de programação. As quatro subrotinas, apesar de independentes, são muitos similares em seu algoritmo.

O algoritmo inicia-se com a transformação de um número binário com n números de bytes em 8n chars que contém os bites dispostos de forma ordenada de maneira que facilite a faze posterior do algoritmo. Como memória não é uma restrição para o programa, o número de alocações não será um problema. Esses bites dispostos um a um em cada char serão transformados novamente no número decimal correspondente.

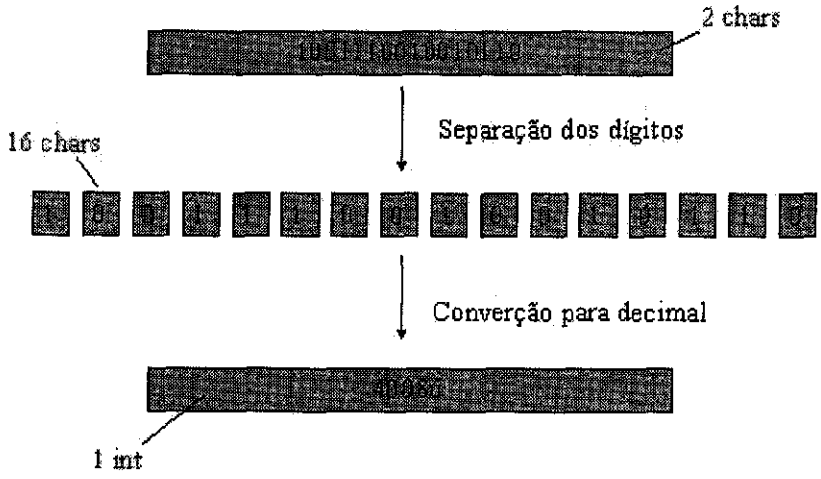


Figura 17: Algoritmo de converção de bases numéricas

Por exemplo, o número 40086 é recebido pela porta serial por 2 chars (2 bytes). No primeiro byte, é recebido o número 10011100, ou seja 9C, que representa o número decimal 156. O segundo número recebido é 10010110, ou seja 96 em hexadecimal, que representa o número decimal 150. Esses dois números foram transformados em 16 grupos, onde cada char deverá receber um valor correspondente ao valor binário do algarismo. Com esses valores alocados

em 16 espaços (no caso de 2 bytes) fica fácil introduzir um pequeno loop para obter o valor decimal correspondente da união dos 2 bytes, ou seja: 40086.

Segue abaixo a ilustração de uma das subrotinas que fazem parte do programa.

```
double convertedois(char a[2])

{
        int i,j,rest,k;
        char z[16];
        double c;

        k=1;

        for(j=0;j<2;j++)
        {
                for(i=0;i<8;i++)  //ajusta o binário
                {
                rest=a[j]%2;
                z[k*8-1-i]=rest;
                a[j]=a[j]/2;
                }
                k=k+1;
        }

        c=0;
        for(i=0;i<16;i++) //converte para decimal
        {
                c=c+(pow(2,(15-i))*z[i]);

        }
        return(c);
}
```

Figura 18: Codificação do algoritmo de converção de bases numéricas

O software segue com a proposta de trabalho em tempo real. Para isso foi necessário preparar uma subrotina com o seguinte algoritmo:

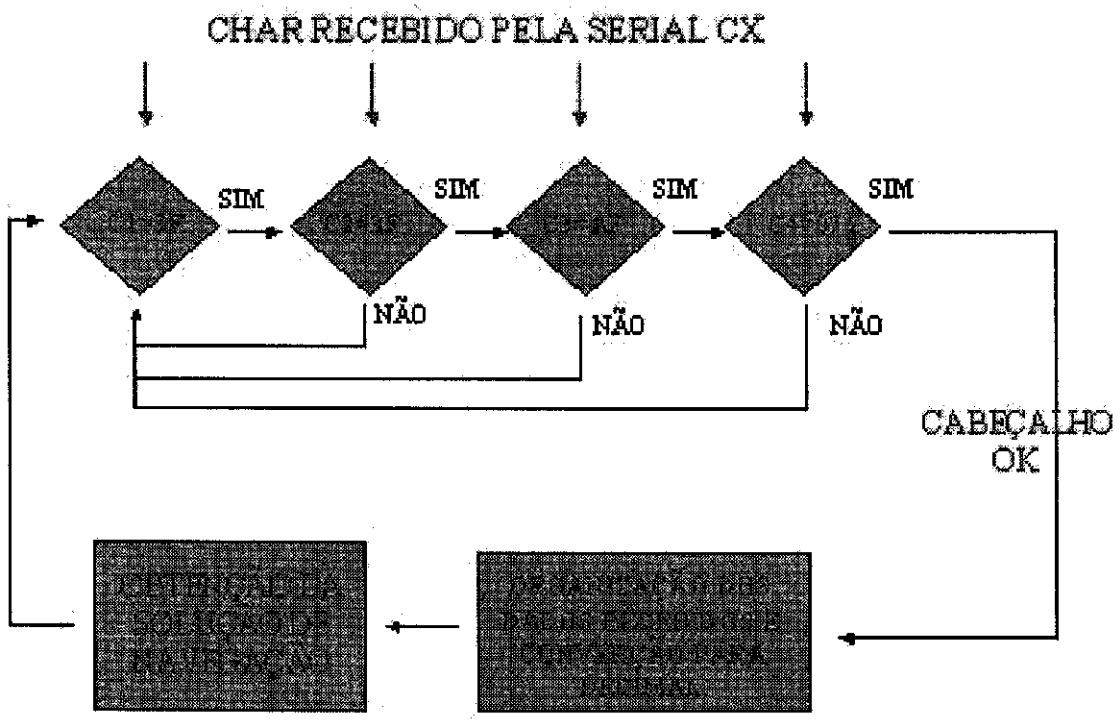


Figura 19: Algoritmo tempo REAL

A primeira mensagem de interesse é a de dados brutos, cujo cabeçalho é 3F3F AF07, e é isso que deve ser comparado com a mensagem que chega pela porta serial. A rotina que foi apresentada no algoritmo acima faz exatamente isso, foram feitos quatro loops para verificar esses quatro primeiros bytes do cabeçalho da mensagem. A cada valor confirmado procura-se o próximo valor, caso contrário reinicia-se o ciclo. Veja o algoritimo codificado abaixo:

```
d=0;
        while(d=!0x07)
                {
                        a=_inp(COM1);
                                if(a==0x3f)
                                        b=_inp(COM1);
                                                if(b==0x3f)
                                                        c=_inp(COM1);
                                                                if(c==0xaf)
                                                                        d=_inp(COM1);
                }
```

Figura 20: Codificação da conferência do cabeçalho

Com o cabeçalho conferido dá-se início a uma imensa rotina onde os dados que são recebidos são organizados e divididos em blocos de acordo com o protocolo para que os mesmos possam ser convertidos em números decimais a partir das subrotinas já mencionadas acima.

Tendo em mãos os dados convertidos em decimal, é possível utilizar os métodos de solução de navegação para obter às coordenadas do objeto em estudo. O ciclo se reinicia com a aferição do cabeçalho de uma nova mensagem a fim de obter a solução de navegação em tempo real. O ciclo pode ser parado a partir da tecla ESC.

Houveram alguns problemas nessa fase da programação que até então não haviam sido resolvidos. A rotina que verifica o cabeçalho apresentou uma falha na execução do programa. Os dados recebidos pela porta serial não eram comparados com os valores inseridos no loop e essa parte da rotina era simplesmente "pulada". Um outro problema foi que as mensagens de efemérides não eram recebidas a partir do comando da porta serial, o que foi resolvido na próxima parte do projeto após o programa ser reescrito.

Apesar dos problemas enfrentados, de forma paralela tentou-se verificar se as outras partes do software estariam funcionado corretamente, e como mostrado nesse mesmo capítulo prova-se através de simulação que não ocorrem em tempo real, que a mudança de base, o set up da comunicação serial e a organização dos dados estariam funcionando corretamente.

Devido a esses impasses fica impossível a geração de soluções de navegação em tempo real, pelo menos até que se resolva os problemas. Quanto a mensagem de efemérides existe a possibilidade de utilizar a Internet para prever as efemérides dos satélites no momento em que se fará a medição, ou aguardar alguma solução do fabricante do equipamento, que no momento estuda o caso.

Com o problema do cabeçalho, ainda se estuda novos caminhos de programação ou que se entenda qual a falha da programação. Existem ainda algumas soluções que foram apresentadas em fases anteriores do trabalho que mostram alguns resultados obtidos de maneira manual ou por simulações que não ocorreram em tempo real e sim por análise de dados coletados e armazenados em arquivos de formato *.txt.

7.2 RESULTADOS

Para a análise dos dados dos satélites escolheu-se um local adequado para a recepção dos mesmos, um local consideravelmente aberto, em um dia que não estivesse nublado e que ao menos 4 satélites estivessem disponíveis, possibilitando assim uma boa análise dos resultados.

As mensagens foram coletadas e armazenadas em um arquivo a fim de serem analisadas "offline". As mensagens nos arquivos contêm caracteres que não podem ser lidos diretamente num editor de textos, o que não impede de serem lidas através de software. Em primeiro lugar, as mensagens coletadas foram comparadas com a estrutura da mensagem do fabricante, em parâmetros como dimensões, cabeçalho, check sum, entre outros.

Ocorreram alguns problemas na interpretação dos dados com a estrutura fornecida pelo fabricante no manual do receptor, essas diferenças a princípio foram atribuídas a bugs, a erros de protocolos e a inexatidão do manual. Consultamos o fabricante e fomos esclarecidos de que existiam dados nulos (0x00) que eram ignorados pelo software, portanto não apresentados na impressão dos dados.

Após a conferência da estrutura dos dados partiu-se para a fase de conversão dos dados binários para dados em decimal. Somente os dados da mensagem de dados crus foram convertidos e armazenados na memória do software. Os dados de efeméride, em razão de serem dados enviados toda vez que há alguma alteração em algum satélite não foram recolhidos pelo receptor e sim coletados via Internet.

Esses dados foram lançados num software, o qual gerou as pseudodistâncias e coordenadas dos satélites. Enfim essas coordenadas puderam ser inseridas nos algoritmos matemáticos, gerando assim as soluções de navegação.

Na tabela seguinte estão relacionados os resultados obtidos na fase anterior do projeto onde a solução de navegação é obtida a partir de dados que convencionamos estáticos, isso pelo fato dos dados injetados no software serem de referência, ou seja, terem sido encontrados a partir de bibliografias e até mesmo na Internet.

ERRO(m)	GEOMÉTRICO	ALGÉBRICO	ESTATÍSTICO
X	-1579,869	-2,1554E-02	4,9152E-03
Y	9063,9479	-4,3333E-02	4,0050E-02
Z	-4070,309	1,3508E-03	-1,3675E-03
Δr	1137,9233	-0,02117873	0,014532567

Tabela: Resultados de Teste da fase anterior do projeto

Em seguida, mostra-se os dados gerados por um receptor GPS e lançados diretamente no software. A coleta foi realizada em Guaratinguetá com coordenadas aproximadas de 22.81198° Sul, 45.20050° Oeste, 629.4m de altitude.

Os dados gerados pelo software foram comparados com a solução dada também pelo software fornecido pelo fabricante, rodado em ambiente Windows. Esses dados estão dispostos na tabela abaixo:

PRN	X (m)	Y (m)	Z (m)	Pseudorange (m)
06	4321548.098	-16113266.768	-20447051.439	21591553,27
09	14391355.351	-14333548.847	16472675.570	23802395,23
21	5152057.244	-25738225.859	2053865.566	22054198,31
25	-10523829.014	-16836957.880	-17272537.434	24393079,87

Tabela: Dados para teste obtidos dia 23/05/2004 às 20:35:40 (GMT)

Na coluna PRN fornece-se a identificação dos satélites GPS rastreados, X,Y,Z são as coordenadas dos satélites GPS naquele instante, e o pseudo-range é a medida bruta obtida pelo receptor em tempo real.

Os resultados do processamento de posição do usuário, usando-se os métodos propostos, em conjunto com o obtido pelo receptor ORBISAT, são mostrados conforme a tabela abaixo:

	X (m)	Y (m)	Z (m)
Geométrico	4145553,29	-4173852,58	-2457716,10
Algébrico	4145558,81	-4173857,91	-2457719,23
Estatístico	4145558,81	-4173857,91	-2457719,23
Receptor	4145180,00	-4174294,00	-2457783,00

Tabela: Resultados de Teste da fase atual do projeto

Verifica-se a existência de diferenças entre os resultados computados pelo software e os resultados do receptor. Deve-se frisar que foi utilizado um dos resultados do receptor para comparação, que não está no mesmo instante dos dados.

Assume-se que as maiores fontes de discrepâncias se referem a fatores como a existência de diferenças entre os relógios do receptor e dos satélites e a erros de propagação. Esses erros

deverão ser analisados em uma etapa seguinte do trabalho e devem ser reduzidos a partir de um método iterativo, onde a partir de um chute inicial consegue-se reduzir a discrepância obtida.

CAPÍTULO 8

PROCESSAMENTO EM TEMPO REAL

8.1 NOVA ARQUITETURA PARA O SOFTWARE

Todo o projeto até agora baseou-se em dados colhidos em um instante anterior ao dos cálculos de posição, o foco agora do projeto foi alterado para fazer o programa rodar em tempo real, dando as coordenadas de segundo a segundo.

Para que isto ocorre-se o programa teve que ser todo refeito e as seguintes mudanças implementadas:

- Código reescrito usando C++ no ambiente Borland C++ Builder
- Integração de todos os algoritmos em apenas um executável
- Acesso a porta serial com programação em alto nível
- Programa dotado de multiprocessamento através do uso de "threads"
- Software funcionando no modo gráfico, compatível com as versões mais atuais do Windows

Para facilitar o desenvolvimento do software com uma interface gráfica o mesmo passou a ser reescrito no ambiente Borland C++ Builder 6.0, pois é mais fácil criar interfaces gráficas nele do que no Microsoft Visual Studio 6.0.

Todos os algoritmos para leitura de porta, interpretação dos dados, calculo de coordenadas, agora serão compilados em apenas um arquivo executável.

Para maior compatibilidade do programa, o acesso a periféricos de E/S (porta serial no caso), teve que ser feito usando programação em alto nível, ou seja, o software da um comando ao sistema operacional e esse por vez faz o acesso a porta serial. Caso isso não fosse feito, o programa não seria compatível com o Windows 2000/XP ou NT pois esses sistemas operam no modo usuário, e não dão ao usuário acesso completo a máquina.

A passagem da programação da porta serial de baixo para alto nível foi feita com a substituição de algumas instruções utilizadas anteriormente no programa.

- _inp() que coleta dados da porta serial foi substituído por ReadFile()
- _outp() que coleta dados da porta serial foi substituído por WriteFile()

Para que essas funções funcionem é necessário primeiramente abrir a porta com o comando OpenFile(). A configuração da taxa de transmissão agora são feitas com os comandos BuildCommDCB() e SetCommState(), e os timeouts da porta agora são definidos utilizando SetCommTimeouts().

Muitos dos conhecimentos sobre o funcionamento dos circuitos que controlam a comunicação serial são desnecessários, pois o sistema operacional cuida dessa parte automaticamente. Devido ao programador agora se encontrar "longe" do funcionamento interno da maquina, chamamos essa técnica de "programação de alto nível".

Foram desenvolvidas então as rotinas para leitura da porta serial e interpretação dos dados colhidos. Com a seguinte instrução a porta serial era lida e os dados enviados para a variável InBuff.

```
ReadFile(hComm, InBuff, 1000, &dwBytesRead, NULL);
```

Figura 21

Pórem havia um problema com a rotina de leitura, para que os dados não fossem lidos continuamente o programa deveria executar esse comando em um loop infinito. Mesmo o resto do programa estando nesse loop, a interface gráfica do programa ficava travada, nen mesmo o botão para parar a leitura poderia ser acionado, e os resultados não poderiam ser escritos na tela do programa.

8.2 MULTIPROCESSAMENTO

Para resolver o problema do loop infinito travando o programa, necessitaríamos de uma arquitetura de software não seqüencial, a solução foi adotar o conceito de multiprocessamento ao programa, onde várias partes separadas rodam independentes de sí, cada uma fazendo a sua parte. Assim a rotina de leitura da porta serial não travaria todo o programa. Foi utilizado então o conceito de "threads" no programa, onde cada thread seria como se fossem processos separados rodando simultaneamente.

Para usar essa ferramenta em C++ foram criadas classes TThread para o programa, que ficou então dividido em 3 dessas threads:

- Thread Principal do programa: inicializa o programa, roda a interface gráfica, inicializa a porta serial, inicia e termina as outras duas threads;

- Thread de leitura da porta: lê continuamente a porta serial em busca de dados e os envia a um buffer comum a todas threads;
- Thread de análise de dados: analisa os dados verificando sua integridade, realiza todos os cálculos de efeméride e posicionamento, e os envia para a thread principal para que sejam exibidos na tela do programa.

Foi somente com o uso desse conceito que se tornou viável a execução do programa para cálculos em tempo real.

A thread de leitura da porta lê os dados e os envia para um buffer onde ele fica armazenado para ser lido pela thread de análise dos dados

```
while(!Terminated)
  {
  ReadFile(hComm, InBuff, 1000, &dwBytesRead, NULL);
  if(dwBytesRead)
    {
    for (i=0;i < dwBytesRead; i++)
          dados[i]=InBuff[i];
    dados[i]=NULL;
    gravabuffer();
    }
  }
```

Figura 21

A função acima roda em um loop infinito até que a flag Terminated seja colocada em 1, o que ocorre quando a thread principal ordena a parada da leitura dos dados (quando se pressiona o botão Parar Leitura). A parte de análise de dados também roda em um loop infinito definido da mesma forma, o que impede que elas sejam paradas antes de terminar alguma rotina que estejam executando, o que poderia acarretar em instabilidade no programa.

```
if(ler()==0x3f)
 {
   vetor[0]=0x3f;
   if(ler()==0x3f)
   {
     vetor[1]=0x3f;
     a = 10;
     for(i=2 ; i<a ; i++)
     {
       vetor[i] = ler();
       if ( (vetor[3]==0x07)&&(i==3) )
           a = 186;
       if ( (vetor[3]==0x08)&&(i==3) )
           a = 138;
     }
```

```
}
```

Figura 22

8.3 ANALISANDO OS DADOS EM TEMPO REAL

Os dados do buffer são lidos byte a byte pela função ler() contida na thread de analise de dados, como vemos na figura 22. É então identificado o início da mensagem que contém os bytes 0x3F 3F AF e depois é lido o próximo byte que determina a natureza da mensagem, sendo ela de pseudo-distância quando 0x07 ou de efeméride quando 0x08. Após saber o tipo da mensagem, os bytes são lidos até o final de cada mensagem, assim a mensagem fica inteira contida no vetor de dados e o restante dos dados continua intacto no buffer, o que assegura ao programa que nenhuma informação será descartada.
Essa rotina é importante pois no programa anterior apenas a mensagem de pseudo-distancia era considerada, e a posição dos satélites era colocada manualmente no programa através de uma pesquisa em um site da NASA, que posta de tempos em tempos a posição dos satélites.

Tendo então uma mensagem completa no array vetor[], é feita a checagem do checksum para garantir que os dados não chegaram corrompidos do receptor GPS como indica a figura 23.
O checksum se baseia em operações XOR realizadas bit a bit desde o primeiro byte da mensagem (0xAF) até o byte que precede o checksum enviado pelo receptor.

```
__fastcall anadados::checksum()
{
    int i;
    cs = vetor[1]^vetor[0];
    for(i=2; i<fim-1 ; i++)
        cs = vetor[i]^cs;
    return cs == vetor[fim-1];
}
```

Figura 23

Caso o checksum calculado pela função seja diferente do enviado pelo receptor, os dados são descartados, o contador de mensagens erradas aumenta, e a thread volta a ler novos dados do buffer.
Caso a mensagem seja de pseudo-distância ela tem o seguinte destino:

- È retirado dela a semana GPS e o tempo do sistema GPS em segundos que é utilizado nos cálculos de posicionamento dos satélites;
- As pseudo-distâncias para até 12 satélites são retiradas para calculo da posição do usuário;

- A relação sinal/ruído da recepção de cada satélite também é retirada desta mensagem.

No caso de mensagem de efeméride são retirados os dados para calculo da órbita de um satélite, cujo o numero PRN está contido nessa mensagem. Os dados de efeméride são então salvos em um array struct chamado efemeridedata, onde o índice do array corresponde ao PRN do satélite.

Quando uma mensagem de pseudo-distância chega ao sistema, a posição de cada satélite que tenha seus dados salvos na "efemeridedata", tem sua posição calculada e mostrada na tabela de satélites que será mostrada a seguir quando será mostrada a interface gráfica.

```
//constantes
    double micro,OMEGAEDOT,A;
    micro = 398600500000000;
    OMEGAEDOT = 0.000072921151467;
    A = pow(raizdeA,2);

//calcula Mk anomalia média
    double tk,Mk,n0,n;
    n0 = sqrt(micro/pow(A,3));
    tk = t - toe;
    if (tk > 302400) tk -= 604800;
    if (tk < -302400) tk += 604800;
    n = n0 + deltan;
    Mk = M0 + n*tk;

    Mk = radtosc(Mk);

//calcula anomalia excentrica Ek numericamente
    double Ek,E,dE;
    Ek = Mk;
    int a;
    for (a = 1; a<50; a++)
    {
        E = Ek;
        Ek = Mk + e*sin(Ek);
        dE = radtosc(Ek-E);
        if (abs(dE) < (pow(10,-12)) )
            break;
    }
    Ek = radtosc(Ek);
//calcula anomalia verdadeira Vk
    double Vk;
    Vk = atan2(sqrt(1 - pow(e,2))*sin(Ek), cos(Ek) - e);

//calcula argumento de latitude
    double arglat;
    arglat = Vk + omega;
```

```
    arglat = radtosc(arglat);

//calcula correcoes harmonicas
    double deltauk,deltark,deltaik;
    deltauk = Cus*sin(2*arglat) + Cuc*cos(2*arglat);
    deltark = Crs*sin(2*arglat) + Crc*cos(2*arglat);
    deltaik = Cis*sin(2*arglat) + Cic*cos(2*arglat);

//medidas corrigidas
    double uk,rk,ik;
    uk = arglat + deltauk;
    rk = A*(1 - e*cos(Ek)) + deltark;
    ik = i0 + deltaik + IDOT*tk;

//posicoes no plano orbital
    double xklinha,yklinha;
    xklinha = rk*cos(uk);
    yklinha = rk*sin(uk);

//latitude do nodo ascendente corrigida
    double OMEGAK;
    OMEGAK = OMEGA0 + (OMEGADOT - OMEGAEDOT)*tk - OMEGAEDOT*toe;
    OMEGAK = radtosc(OMEGAK);
//posicoes x,y,z espaciais do satelite
    xk = xklinha*cos(OMEGAK) - yklinha*cos(ik)*sin(OMEGAK);
    yk = xklinha*sin(OMEGAK) + yklinha*cos(ik)*cos(OMEGAK);
    zk = yklinha*sin(ik);

    sat[prn].x = xk;
    sat[prn].y = yk;
    sat[prn].z = zk;
```

Figura 24

Vemos acima a função satpos(), que calcula a posição através de calculos já descrito em literaturas e normas, usando a teoria da mecânica celeste. Toda vez que uma nova pseudo-distância chega ao programa essa função é executada para cada um dos satélites visíveis.
Após a posição de no mínimo 4 satélites ser definida, o programa inicia o calculo da posição do usuário que é atualizada segundo a segundo na tela Usuário da interface gráfica.

Devido as rotinas estarem escritas inicialmente em Fortran, apenas o método algébrico foi implementado no novo programa, pois muita dificuldade foi encontrada para realizar os cálculos em C++, devido ao uso de operações matriciais nos algoritmos. A inversão de matriz, por exemplo, é uma operação difícil de ser implementada que ainda está sendo testada de várias formas, usando funções pré-prontas de terceiros e bibliotecas .dll. Foram testadas até agora funções matriciais contidas no software MatLab 6.5 e no site http://www.chelcom.ru/~anton/projects/matrix/Matrix.html .

8.4 INTERFACE GRÁFICA

A interface gráfica do programa é muito simples e prática tendo na parte superior um menu com várias informações além do botão iniciar/parar leitura da porta serial, como mostra a figura abaixo:

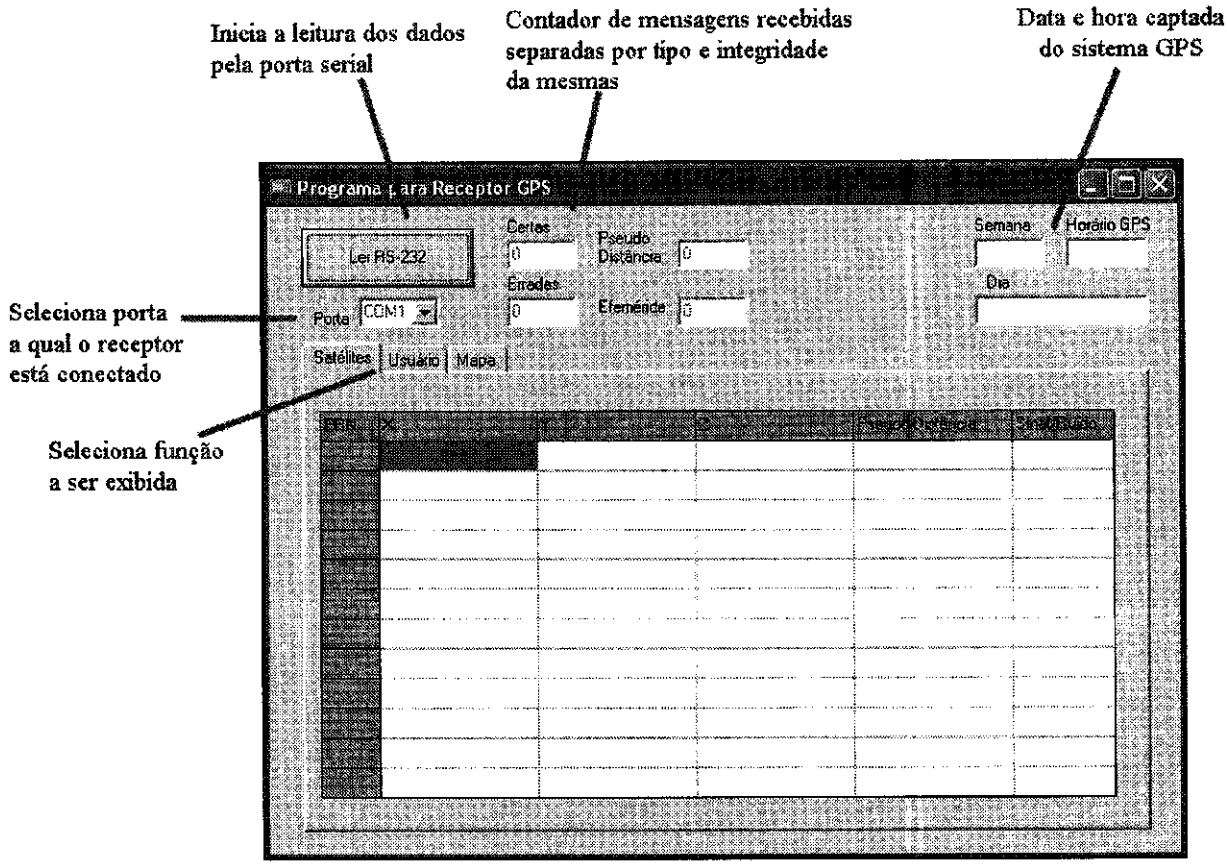


Figura 25

Após pressionado o botão Ler RS-232, o software se conecta ao receptor Orbisat, o configura para o modo de mensagens cruas, e inicia as threads de leitura e análise dos dados.
A medida que a função satpos() for executada, os resultados dela obtidos vão parar na tabela de satélites, onde são exibidos suas coordenadas x,y,z , pseudo-distância entre satélite e usuário, e relação sinal ruído da portadora.

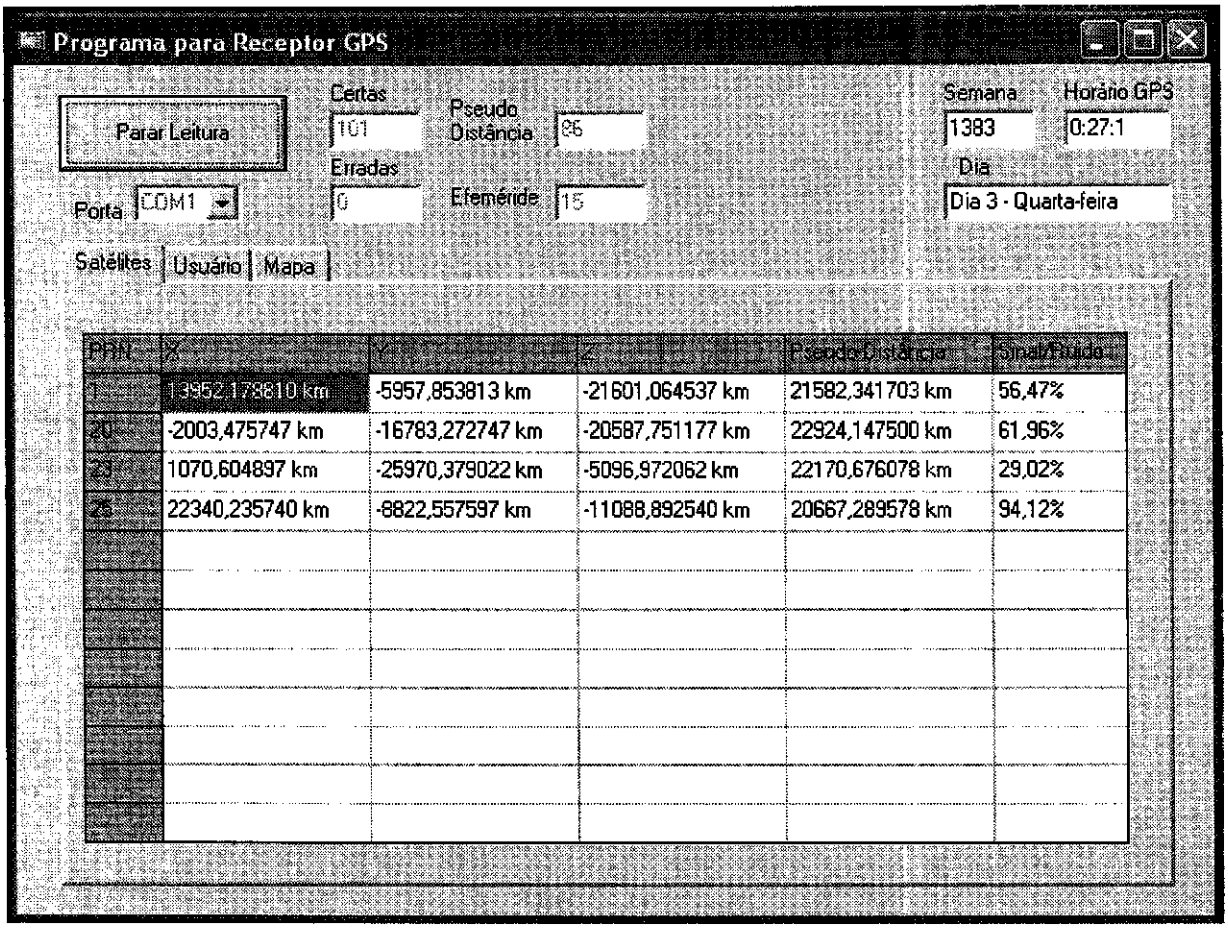
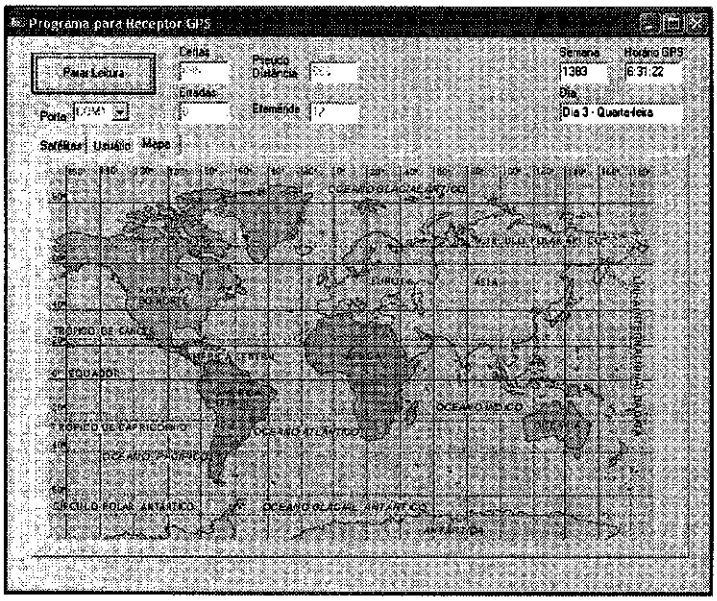
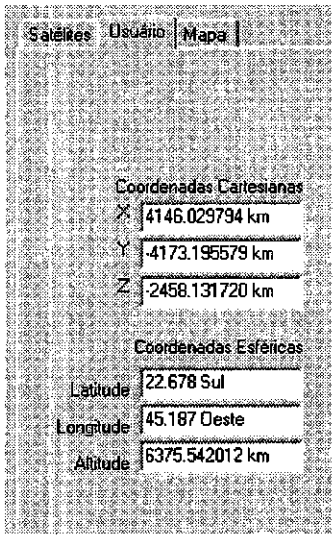


Figura 26

Com 4 satélites ou mais a posição do usuário é determinada e preenchia na guia Usuário do programa, além do mapa ter assinalado com um ponto vermelho essa posição.





Figuras 27 e 28

Nas figuras acima vemos coordenadas calculadas na cidade de Guaratinguetá-SP, todas em tempo real atualizadas de segundo em segundo.

O ponto vermelho no mapa indicando a posição, funciona de uma forma muito simples bastando apenas uma regra de 3. Quando a posição do usuário é calculada esse bitmap em forma de um ponto vermelho tem sua posição alterada em cima da outra figura do mapa mundi simplesmente alterando as propriedades Image2->Left e Image2->Top.

Para o funcionamento da interface gráfica, foi muito importante o uso da instrução Synchronize() que é chamada pela thread de análise de dados. Esta função deve ser utilizada toda vez que se desejar comunicação entre as threads secundárias e a principal (como preencher a tabela de satélites, ou o horário) o que evita falhas de proteção no sistema.

O código fonte do programa está todo comentado sendo que é possível a uma pessoa com um pouco de conhecimento em programação entende-lo facilmente. Constantemente o código está sendo reescrito e aprimorado para melhorar sua organização e aumentar a performance do programa.

CAPÍTULO 9

CONCLUSÕES E PLANOS FUTUROS

O objetivo desse trabalho foi investigar, equacionar, desenvolver, implementar e comparar soluções de navegação possíveis através do uso de medidas obtidas por receptores GPS ("Global Positioning System"), em ambiente espacial. Como resultado obteve-se um pacote de programas que comparam os três métodos (Geométrico, Algébrico e Estatístico) de computação da solução de navegação em diversos aspectos e a estrutura de um software que em tempo real retorna as soluções de navegação correlacionando os três algoritmos propostos.

As comparações já realizadas anteriormente mostraram que o método estatístico é o mais preciso dentre os três, porém o mais complexo. O método algébrico ocupa e exige mais memória. Para obtenção de uma proximação grosseira, uma alternativa é o método geométrico que é bem menos complexo e rapidamente retorna um valor bem próximo ao da solução.

Uma alternativa para melhorar a precisão dos softwares é a introdução do cálculo do parâmetro GDOP (Geometric Dilution Of Precision), que produz uma comparação entre

diferentes combinações de satélites disponíveis no instante em que se quer determinar a posição e descreve a combinação que está melhor distribuída, possibilitando uma maior precisão nos resultados.

Um software foi desenvolvido para calculo das coordenadas, por meio de leituras feitas em tempo real do receptor Orbisat. Utilizado um interface gráfica prática e agradável, o programa está muito próximo de ser terminado e se apresentou estável e sem erros da recepção dos dados.

A precisão em posição obtida está de acordo com estimativas iniciais de precisão, já que as correções para refinamento não foram ainda implementadas, sendo objeto do próximo período de Iniciação Científica.

Pôde-se verificar que o tempo de processamento de todo o software é irrisório, o que torna o software utilizável em computadores antigos que possuam porta serial livre. O programa permite modificações de acordo com a necessidade do usuário a exemplo da adição de novos mapas, o que incluiria mapas urbanos assim como existente nos GPS comerciais.

Espera-se que a partir dessas análises, em uma próxima etapa de trabalho, seja possível determinar a posição do usuário incluindo agora todas as correções do sinal (relativísticos, tempo de trânsito, erros do relógio do satélite GPS, erro do relógio do receptor, e mesmo correções atmosféricas tais como dos da troposfera e ionosfera) o que aumentaria a precisão das coordenadas obtidas.

REFERÊNCIAS BIBLIOGRÁFICAS

Amato, A.L. Cálculo de DOP´S para Satélites GPS. Trabalho de Graduação. **FEG/UNESP** Guaratinguetá 1999.

Bancroft, S. **An algebric solution of the GPS equations.** Ieee Transactions on Aerospace and Eletronics Systems, v.AES-21, n.7, 1985.

Bertiger, W.I; Bar-Server, Y.E.; Christensen, E.J.; Davis, E.S.; Guinn, J.R.; Haines, B.J.; Ibanez-Meier, R.W.; Jee, J.R.; Lichten, S.M.; Melbourne, W.G.; Muellerschoen, R.J.; Munson, T.N.; Vigue, Y.; Wu, S.C.; Yunck, T.P.; Schutz, B.E.; Abusali, P.A.M.; Rim, H.J.; Watkins, M.M.; Willis, P. GPS precise tracking of TOPEX/Poseidon: results and implications. **Journal of Geophysical Research**. v. 99, n. C12, p. 24449-24463, 1994.

Dana, P. H. **Global Positioning System Overview.** Department of Geography,

University of Texas, Austin. [online] http://www.utexas.edu/depts/gcraf/notes/gps/gps.html, Julho, 1998.

Fang, Bertrand T. Geometric Dilution of Precision in Global Positioning System Navigation. **Advances in the Astronautical Sciences**, AAS81-4005, v.4, nº1 p. 93-94, 1981.

Gomes, V. M. **Soluções de Navegação via GPS**. São José dos Campos. Relatório de Final de Projeto de Iniciação Científica Pibic - CNPq. Instituto Nacional de Pesquisas Espaciais, Janeiro 2002.

Kuga, H.K., Lopes, R.V.F., Prado, A.F.B.A., Nascimento, J. M., Chiaradia, A.P. "**Orbit determination and navigation using GPS.**" Organizado por Balthazar, J.M., Gonçalves, P.B., Brasil, R.M.F.L.R.F. Non-linear dynamics, chaos, control and their aplications to engineering sciences. Vol 3: New trends in Dynamics and Control.. 2000, v.3, p.93-108.

Leick, A. **GPS Satellite Surveying.** Department of Surveying Engineering – University of Maine, Second Edition, Wiley-interscience Publication, 1994.

Logic, B **Interfacing The Serial - RS-232 Port** 2004 [online] http://www.beyondlogic.org

Lopes, R. V. F.; Kuga, H. K. ORBEST - A GPS navigation solution algorithm without DOP analysis. **Advances in the Astronautical Sciences**, AAS97-108, v. 95, p. 153-166, 1997.

Lopes, R. V. F.; Kuga, H. K. "Optimal estimation of local orbit from GPS measurements". **Journal of Guidance, Control and Dynamics**, v.11, n.2, p. 186-188, 1988.

Lundberg, J. B.; Minter, C.F.;Yoon, S. "**Analysis of algebraic solutions to the GPS navigational equtions.**" AAS/AIAA Space Flight Mechanics Meeting, Huntsville, AL, Feb. 10-12, 1997. AAS 97-110.

Nicoletti, G.W.; Vilhena de Moraes, R. Analitical Approach for Evoluation of Orbital Pertubations due Atmospheric Drag. **Anais do COBAIN2001** (CD-ROM, 2001), pg 191-196.

Orbisat, **Manual de Operação** – ORB 2002 RLP – GPS FARO 1.0 - 2002

Orlando, V.; Kuga, H. K.; Lopes, R. V. F. Reducing the geopotential tesseral harmonic effects on autonomous longitude drift control of Sun-synchronous satellites. **Advances In The Astronautical Sciences**, v.95, p.361-374, 1997.

Orlando, V.; Kuga, H. K. Analysis of an autonomous orbit control concept using GPS. **Revista Brasileira de Ciências Mecânicas**, v.21, p. 52-59, 1999.

Orlando, V.; Kuga, H. K. Effect analysis of maximal allowable maneuver application rate for an autonomous orbit control procedure application. In: International Symposium

Space Dynamics. **Proceedings of International Symposium Space dynamics**. Biarritz: CNES, 2000a.

Orlando, V.; Kuga, H. K. Investigation on autonomous orbit control using DIODE and GPS navigation systems. Editado por Prado, A. F. B. A. **Advances in Space Dynamics**, v.1, p.338-348, 2000b.

Papyrus, C. T. **Computer Tecnologies Ltda** 2003[online] http://www.papyrus.co.il/FAQ/gps_applications_market_segments.htm

Senne, E. L. F. **Introdução à Programação de Computadores Com a Linguagem C**. FEG/ Unesp – 2000.

Silva, A. A. Determinação de Órbitas com GPS através de Mímimos Quadrados Recursivo com Rotações de Givens. Tese de Mestrado. **FEG/UNESP** Guaratinguetá 2001.

Strang, G., Borre, K. "Linear algebra, geodesy, and GPS." **Cambridge Press**, Wellesley, 1997.

Matrix Library - http://www.chelcom.ru/~anton/projects/matrix/Matrix.html

NGS Precise Orbits - http://www.ngs.noaa.gov/GPS/GPS.html

Ajuda Borland C++ Builder 6.0.

Win32 SDK Reference – Guia de referencia do Kit de Desenvolvimento de Software para windows 32bits.