DEMODULADOR DE BPSK COM RECUPERAÇÃO DE PORTADORAS DEFINIDO EM "SOFTWARE" PARA OS SATÉLITES DO SISTEMA BRASILEIRO DE COLETA DE DADOS

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/CNPq/INPE)

Ramon Augusto Sousa Lins (UFRN, Bolsista PIBIC/CNPq)
E-mail: Ramon_asl@yahoo.com.br

Alexandre Guirland Nowosad (INPE/CRN, Orientador)
E-mail: agnowosad@crn.inpe.br

COLABORADOR

Manoel Jozeane Mafra de Carvalho (INPE/CRN)

Julho de 2008

RESUMO

Apresenta as atividades executadas no período de Estágio Supervisionado, iniciado em janeiro de 2008, no Instituto Nacional de Pesquisas espace Regional Nordeste (INPE/CRN), como uma extensão do trabalho de bolsa de pesquisa iniciado em agosto de 2006. Esta atividade teve por objetivo principal desenvolver um demodulador de *Binary Phase Shift Keying* (BPSK) com recuperação de portadoras para o Sistema Brasileiro de Coleta de Dados (SBCD). O SBCD é um sistema que obtém informações do meio ambiente através da coleta de dados ambientais espalhados pelo território brasileiro retransmitindo-os via satélite para as estações terrenas de Cuiabá, no estado de Mato Grosso, e Alcântara, no estado do Maranhão, onde são processados e enviados a instalação Cachoeira Paulista do Instituto Nacional de Pesquisas Espaciais, localizado no estado de São Paulo. Os dados coletados são usados em pesquisas e em apoio à assessoria à Administração Pública em geral. O SBCD tem por escopo funcionar em tempo real e com robustez. Em função desse objetivo, no seu desenvolvimento, optou-se, pelo uso da tecnologia denominada "Rádio Definido em *Software*". Já havia no INPE/CRN um demodulador em desenvolvimento na linguagem de programação visual LabVIEW™, que foi estudado e testado durante o período de estágio, fazendo-se uso dos dados recolhidos pela Plataforma de Coleta de Dados (PCDs) do INPE/CRN. Foi possível observar que algumas partes do programa funcionavam, e outras estavam incompletas e com erros de operação. Após sucessivos testes, foi possível corrigir e implementar o programa. O demodulador, desenvolvido durante o estágio, usa filtro casado para extrair os bits do sinal. Esta técnica consiste na comparação de duas formas de ondas criadas como padrão, correspondentes aos bits um e zero a serem recuperados, com o sinal recebido, detectando-se então os bits e em seguida os armazenando. Para extrair esses bits corretamente, o *jitter* (falta de sincronismo entre os vetores que estão sendo comparados) é anulado através da sincronização da forma de onda padrão correspondente ao bit um (neste caso simétrica ao bit zero) com o sinal recebido. Após esta etapa ser concluída, iniciou-se o processo de transformação do programa desenvolvido em linguagem LabVIEW™ em linguagem C/C++, tendo em vista tornar o sistema mais robusto e com maior adaptabilidade a outros sistemas, uma vez que o LabVIEW™, não permite acesso a todo seu código fonte, o que poderia resultar em problemas futuros de portabilidade e adaptação, diferentemente de um programa desenvolvido em linguagem C/C++. Esta última atividade se estendeu até o final do estágio.

Palavras-chaves:
Rádio, Demodulador de BPSK, LabVIEW™

LISTA DE FIGURAS E QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BPSK – Chaveamento binário de fase (*Binary Phase Shift Keying*).

CBERS – Satélite Sino-Brasileiro de Recursos Terrestres (*China-Brazil Earth Resources Satellite*).

CLS – Collecte Localisation Satellites

CNAE – Centro Nacional de Estudos Espaciais, França (*Centre National d'Etudes Spatiales*).

EMM – Estação Multi-Missão.

ETR – Estação Terrena de Recepção de Sinais de Satélite.

INPE-CRN – Instituto Nacional de Pesquisas Espaciais - Centro Regional do Nordeste.

MECB – Missão Espacial Completa Brasileira.

MTRs – Minitransmissores Remotos.

NOAA – Administração Nacional Oceânica e Atmosférica (*National Oceanic & Atmospheric Administration*).

PXI – Extensão PCI para Instrumentação (*PCI eXtensions for Instrumentation*).

PCD – Plataforma de Coleta de Dados.

PCI – Interconector de Componentes Periféricos (*Peripheral Component Interconnect*).

PROCODS - Processador de Coleta de Dados.

PNAE – Programa Nacional de Atividades Espaciais.

SBCD – Sistema Brasileiro de Coleta de Dados.

SCD – Satélite de Coleta de Dados.

UHF – Freqüência de Ultra Alta (*Ultra High Frequency*).

UFRN – Universidade Federal do Rio Grande do Norte.

LISTA DE APÊNDICES

SUMÁRIO

1 – INTRODUÇÃO

O estágio ocorreu na Estação Multi-Missão de Natal (EMM-Natal) do INPE/CRN. A EMM-Natal é um projeto de pesquisa e desenvolvimento tecnológico, que possui parceria com a Universidade Federal do Rio Grande do Norte (UFRN). A EMM-Natal será uma estação para múltiplas missões, configurável por *software*, fazendo uso do paradigma da tecnologia de "Rádio definido em *software*". Ela está sendo projetada para ser um centro de recepção de dados, que pretende atender a demanda do Sistema Brasileiro de Coleta de Dados, inserido no Programa Nacional de Atividades Espaciais (PNAE), que hoje não tem cobertura para uma grande parte do território brasileiro.

Dentre os projetos de pesquisas existentes na estação, havia a necessidade de implementação de um Demodulador de *Binary Phase Shift Keying*, pois o banco de homologação do INPE, que analisa os dados recuperados para posterior homologação, precisava ser atualizado. Assim, o estágio teve por objetivo, o desenvolvimento do Demodulador de BPSK para o banco de homologação de PCDs do INPE-CRN, tornando o sistema mais moderno em substituição a algumas partes de hardware que eram antes responsáveis por essa recuperação da mensagem.

O Centro Regional do INPE, com sede em Natal, foi estabelecido por volta de 1970, quando o INPE chamava-se Comissão Nacional de Pesquisas Espaciais (CNAE) e assinou um convênio com o Governo do Estado e Universidade Federal do RN, visando estabelecer um núcleo de apoio aos lançamentos de foguetes e balões operados na Barreira do Inferno.

Após algum tempo de trabalho, o INPE/CRN passou a desenvolver equipamentos destinados à conexão com satélites. Entre outros, montou uma PCD para operar com o Sistema CLS/ARGOS[1] (*Collecte Localisation Satellites*).

Em 1983, a primeira PCD brasileira era testada e homologada nos laboratórios da CNES (*Centre National d'Etudes Spatiales*) – França. Desde então a atividade de coleta de dados ambientais via satélite tem tomado dimensões nacionais, contando hoje com várias unidades operando em todo o Brasil, algumas inclusive operando com satélites brasileiros.

---

[1] ARGOS : Serviço de geo-posicionamento e telemetria de dados global. O serviço possui transponder a bordo dos satélites NOAA para coleta de dados de PCDs, e é operado conjuntamente pela NOAA e CNES.

Após estar capacitado a montar PCDs, o INPE/CRN recebeu a outorga para realizar as homologações para os transmissores do sistema ARGOS que fossem atuar no Brasil. O próximo passo foi o desenvolvimento de um sistema capaz de homologar esses dispositivos no próprio INPE. Assim, por volta de 1986, foram adquiridos os equipamentos necessários para a construção do banco de homologação, fazendo com que o INPE-CRN seja o único local no Brasil autorizado a fazer a homologação de transmissores para que estes possam trabalhar com os satélites da NOAA (*National Oceanic & Atmospheric Administration*). O sistema do INPE é capaz de fazer a homologação tanto de transmissores que trabalharão no Sistema de Coleta de Dados brasileiro, como nos do sistema ARGOS, bastando para isso pequenos ajustes de configuração, já que os testes de desempenho realizados são os mesmos.

Porém, com o passar do tempo, as PCDs evoluíram e o banco de homologação não acompanhou esse desenvolvimento, tornando-se um sistema ultrapassado e de difícil manutenção. Outro problema surgiu quando as PCDs mais modernas não disponibilizavam em sua interface determinados sinais necessários para a operação do banco de homologação. Para não parar as homologações no INPE-CRN, um receptor francês do banco de homologação de transmissores chamados COSPAS/SARSAT, que é usado em aeronaves para o rastreio em caso de pane na mesma, foi adaptado para que o banco do INPE pudesse homologar esses tipos de PCDs mais modernas. Para modernização do banco de homologação utilizou-se da tecnologia PXI (*Peripheral Component Interconnect eXtensions for Instrumentation*) para aquisição de dados e da linguagem de programação LabVIEW^{TM}. Havia em desenvolvimento um demodulador de BPSK definido na linguagem G do LabVIEW^{TM} para recuperação das mensagens enviadas. Este demodulador apresentava-se incompleto e com problemas, que foram corrigidos neste trabalho.

Nas próximas seções serão descritos: o SBCD, quanto aos seus componentes; suas especificações; o seu funcionamento; a teoria usada em sua implementação; as etapas de desenvolvimento; os resultados obtidos bem como algumas as perspectivas futuras do projeto.

2 – DESCRIÇÃO DA APLICAÇÃO E FUNDAMENTAÇÃO TEÓRICA

2.1 O SISTEMA BRASILEIRO DE COLETA DE DADOS

A partir do lançamento do primeiro satélite brasileiro o SCD1, foi iniciado a operação do Sistema Brasileiro de Coleta de Dados ambientais. O SBCD foi desenvolvido no intuito de automatizar o processo de aquisição de dados ambientais e do desenvolvimento de tecnologia nacional. Na figura 1 abaixo é mostrado o sistema visto de um modo geral.

Figura 1 - Sistema Brasileiro de Coleta de Dados ambientais.

O Sistema atual é composto pelas PCDs, MTRs (Míni Transmissores Remotos) e pelos Satélites SCD1, SCD2 e CBERS (Satélite Sino-Brasileiro de Recursos Terrestres). As PCDs e MTRs coletam, através de vários sensores acoplados, dados ambientais e realizam a geo-localização de embarcações, pessoas e animais transmitindo os dados recolhidos em forma de mensagem na faixa de freqüência UHF[2] aos satélites do sistema. Estes satélites, através dos seus *transponders*, recebem a mensagem e as retransmitem na faixa de banda-S[3] para as Estações

[2] UHF – Sigla para *ultra high frequency*, que significa freqüência ultra-alta. Indica a faixa de radiofreqüências de 300 MHz até 3 GHz. Estas freqüências são comuns para propagações de sinais de rádio.
[3] Banda-S – Faixa de freqüência de 2 até 4 GHz , englobando as freqüências de UHF.

Terrenas de Recepção de Sinais de Satélite. Essas estações demodulam as mensagens, recuperam os dados e as informações transmitidas e as envia para o Centro de Missão, em Cachoeira Paulista.

A seguir alguns exemplos de PCDs e MTRs :

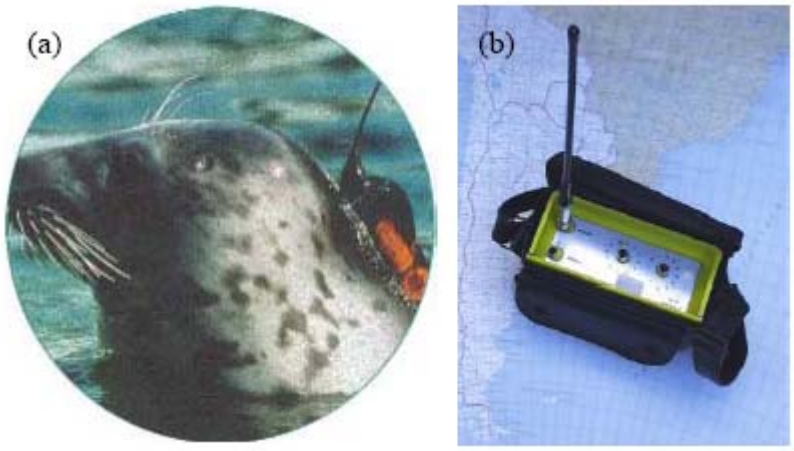
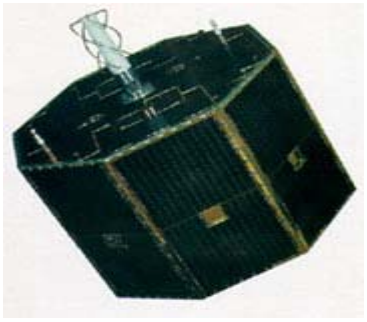


Figura 2 – Exemplos de MTRs.
(a) MTR em um elefante marinho; (b) MTR para comunicação e socorro.

Figura 3 – Exemplos de PCDs.

Os satélites SCD1, SCD2 e o CBERS que fazem parte do SBCD são apresentados na figura 4 abaixo:



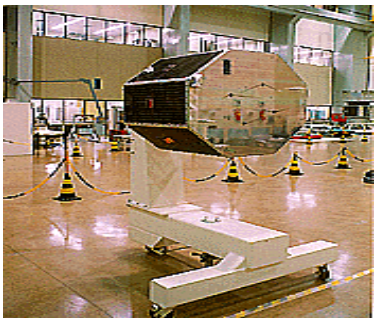


Figura 4 – Satélite SCD1 e SCD2 respectivamente.

As mensagens enviadas pelas PCDs e MTRs aos satélites obedecem a um padrão, estas são enviadas como mensagens de 360 ms a 920 ms de duração, e transmitidas a intervalos regulares de 40 s a 220 s. Os dados são codificados em *Biphase_L* em 400 bps e modulados em fase ± 60^{o} (com portadora residual) nas freqüências de 401,62 MHz (faixa MECB[4]) ou 401,65 MHz (faixa ARGOS).

No Quadro 1, apresenta-se o formato da mensagem:

TRANSMISSÃO COMPLETA T3				
Portadora pura T1	Portadora Modulada T2			
	Preâmbulo	Tamanho da mensagem	Identificação da Plataforma (ID)	Dados dos sensores
T1 = 160 ms ± 2,5 ms	24 bits (FFFE2F)	4 bits	20 bits	N x 32 bits (1≤ N ≤8)

Quadro 1 – Formato e estrutura da mensagem.

[4] Missão Espacial Completa Brasileira

No quadro 1, T1 representa o tempo de transmissão da portadora pura que é de 160 ms e pode variar em ± 2,5 ms.

T2 representa a portadora modulada. Em T2 o preâmbulo representa uma mensagem fixa para facilitar a detecção do sinal recebido, o tamanho da mensagem indica o número de blocos que a mensagem possui (dos 4 bits, 3 são para indicar o tamanho da mensagem e 1 bit é de paridade par). Cada plataforma possui um número de identificação (ID) de 20 bits, em que 14 bits são usados para indicar o número da plataforma e os outros 6 bits são utilizados para o código de checagem de erro. Os dados enviados podem variar de 1 a 8 blocos de 32 bits, com isso a duração mínima de transmissão de T2 é de 200 ms ± 2,5 ms (para 1 bloco de 32 bits) e de 760 ms ± 9,5 ms (para 8 blocos de 32 bits).

A transmissão completa T3 é dada pela soma de T1 e T2, assim T3 = T1 + T2. O tempo total de duração da transmissão pode variar a partir dos limites a seguir : 360 ms ±5ms ≤ T3 ≤ 920 ms ± 12 ms.

2.2 PROCESSO DE MODULAÇÃO

O propósito de um sistema de transmissão é entregar sinal de mensagem de uma fonte de informação em formato reconhecível a um usuário final, com a fonte e o usuário fisicamente separados. Para fazer isso, o transmissor modifica o sinal de mensagem para uma forma apropriada à transmissão através do canal. Esta modificação é realizada por meio de um processo conhecido como modulação, o qual envolve variar algum parâmetro de uma onda portadora de acordo com o sinal de mensagem. O receptor recria o sinal da mensagem original a partir de uma versão degradada do sinal transmitido depois da propagação através do canal.

Essa recriação é realizada utilizando-se um processo conhecido como demodulação, o qual é o inverso do processo de modulação utilizado no transmissor. No entanto, devido à inevitável presença de ruído e distorção no sinal recebido, consideramos que o receptor não é capaz de recriar exatamente o sinal de mensagem original. A degradação resultante do sinal é influenciada pelo tipo de modulação utilizado, considerando-se que alguns esquemas de modulação são menos susceptíveis a ruídos e distorções.

2.3 PROCESSO DE MODULAÇÃO DO SBCD

O *transponder* dos satélites SCD1, SCD2 e CBERS recebem os sinais emitidos pelas PCDs e MTRs, acrescido de ruídos, na faixa de freqüência de 401,635 MHz ± 30 kHz e os transladam para a faixa de 65 KHz a 125 KHz. Em seguida esta faixa modula em fase uma portadora em banda-S de 2,267 GHz, que é transmitida para estações terrenas de recepção.

O sinal, na banda-S, do satélite do SBCD é recebido nas ETRs (Estações Terrestres Remotas) do INPE, onde é demodulado, recuperando-se a faixa de freqüência de 65 KHz a 125 KHz que contém os sinais das PCDs. Esses sinais são entregues aos PROCODS (Processador de Coleta de Dados), que detectam e demodulam os sinais das PCDs, resultando nos sinais BPSK. Estes últimos são demodulados a fim de se obter as mensagens desejadas.

2.4 MODULAÇÃO ANGULAR

O processo de transmissão dos sinais relacionado ao SBCD sofre a modulação por chaveamento de fase (modulação PSK) que é um segmento da modulação em fase (PM). A modulação PM por sua vez é um segmento da modulação angular, na qual o ângulo da portadora é variado linearmente com sinal de mensagem (banda base).

Admita-se que $\theta i(t)$ indique o ângulo de uma portadora senoidal modulada, que se supõe ser uma função do sinal de mensagem. Expressa-se a onda modulada angular resultante como:

$$s(t) = A_c \cos[\theta i(t)]$$

variando o ângulo como : $\theta i(t) = 2\pi f_c t + k_p m(t)$

onde $2\pi f_c t$ representa o ângulo da portadora não-modulada, e a constante k_p, a sensibilidade à fase do modulador. Desta forma o sinal modulado em fase s(t) é descrito no domínio do tempo por:

$$s(t) = A_c \cos[2\pi f_c t + k_p m(t)].$$

2.5 MODULAÇÃO BPSK

A modulação BPSK caracteriza-se quando a informação do sinal digital é embutida nos parâmetros de fase da portadora. Nesta modulação quando ocorre a variação de um bit “0” para um bit “1” ou um bit “1” para um bit “0”, acontece uma mudança na fase da portadora em 180^o.

2.6 FILTRO CASADO

O processo de demodulação acontece de diferentes formas seguindo o caminho inverso da modulação utilizada no processo de comunicação. Escolheu-se para o processo de demodulação do sinal *Biphase_L* a utilização de filtros casados, pela facilidade de implementação. Essa facilidade existe tendo em vista que o filtro casado pode ser expresso em termos de produto escalar. A partir da Desigualdade de Schwarz, onde $|\langle v, w\rangle| \leq \|v\|.\|w\|$, é possível definir ângulo entre vetores, vetores padrões gerados e os segmentos de vetor do sinal recebido, não nulos em um espaço vetorial provido de um produto escalar.

Com isso, obtém-se que $\left|\frac{\langle v, w\rangle}{\|v\|.\|w\|}\right| \leq 1$, portanto existe um ângulo tal que $\cos\theta = \frac{\langle v, w\rangle}{\|v\|.\|w\|}$.

Realizando o produto escalar entre dois vetores $\langle v, w\rangle = \cos\theta \times \|v\|.\|w\|$ tem-se o máximo valor quando o ângulo entre esses vetores é nulo, caracterizando o filtro casado.

Para eliminar o problema de *jitter* é necessário sincronizar os vetores padrões com o sinal recebido (Proakis [6]), ou seja, os vetores precisam estar (quase) exatamente superpostos. Isso é conseguido procurando-se o máximo do produto escalar. Deslocam-se os vetores padrões até encontrar o máximo por diferenciação numérica usando diferença centrada: $f'(t) \cong f(t+h) - f(t-h)$ onde $h = 1$. O máximo será onde $f'(t) \cong 0$ ou atingir o menor valor possível.

3 – IMPLEMENTAÇÃO DO DEMODULADOR BPSK

3.1. IMPLEMENTAÇÃO EM LINGUAGEM G DO LABVIEW™

O Demodulador BPSK foi desenvolvido em linguagem G que é a linguagem de programação utilizada pelo LabVIEW™. A demodulação baseou-se na implementação de um filtro casado para extração dos bits do sinal. A princípio geram-se duas formas de ondas representadas nas figuras abaixo:

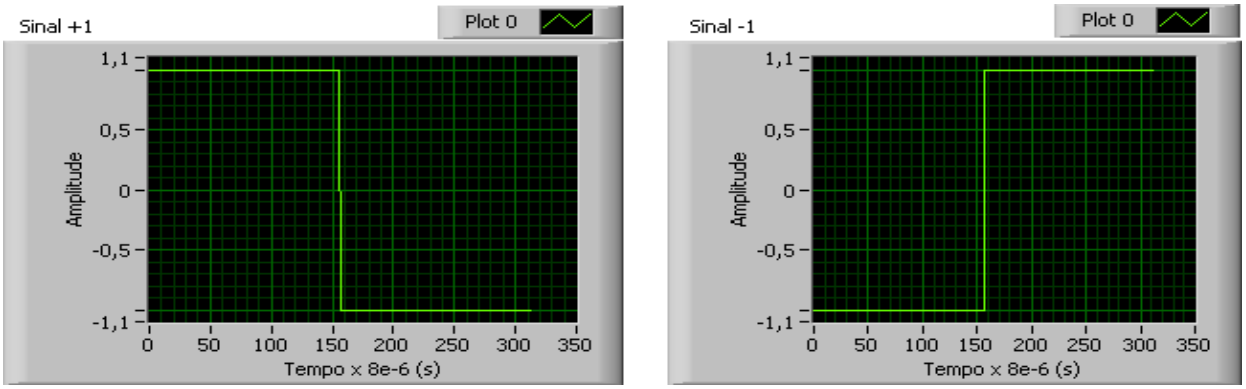


Figura 5 – Sinal +1 e Sinal -1.

A forma de onda gerada teve sua duração baseada nas especificações de projeto do sistema ARGOS. Cada bit transmitido representa uma janela de tempo de 2,5 ms, que representada no sistema digital, com dt = 8 * 10^6 (dt é o tempo de amostragem utilizado para recepção do sinal), gera uma janela com $\frac{2{,}5ms}{8*10^6 s} = 313$ amostras.

Em seguida retira-se a portadora pura presente no sinal e obtém-se apenas a mensagem em *Biphase_L* como é mostrado na figura abaixo.

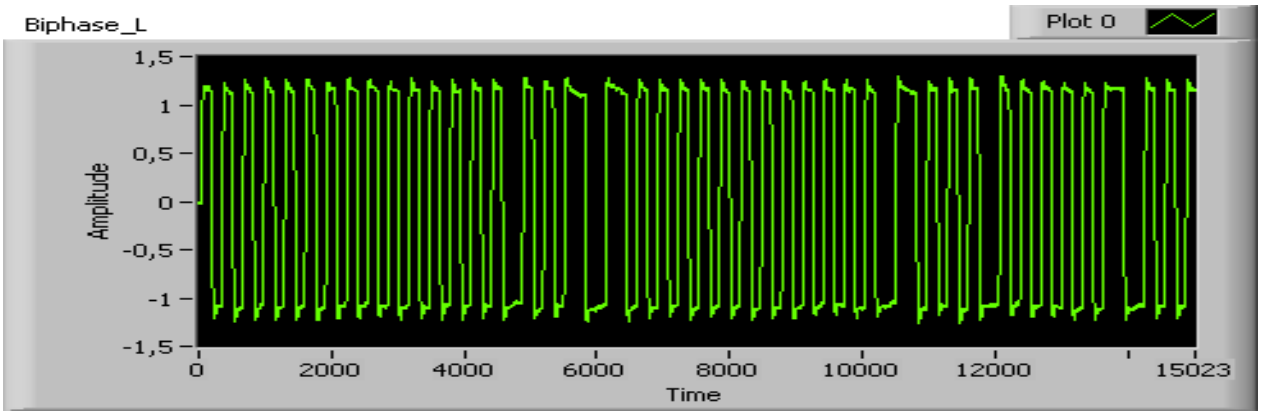


Figura 6 – Mensagem em *Biphase_L* do sinal.

Para a implementação dos filtros casados usam-se dois SubVis[5] *Dot Product* para calcular a projeção do sinal de entrada em cada um dos sinais da modulação *Biphase_L*. O circuito em seguida compara os valores designados como f +1 e f -1, que são as saídas dos filtros casados para o sinal +1 e o sinal -1 respectivamente, onde para f +1 > f -1 tem-se um bit 1 e caso contrário f -1 > f +1 tem-se um bit 0.

Porém os filtros podem estar fora de sincronismo com o sinal de entrada, fenômeno chamado *jitter*. Neste caso, para a decisão ser feita corretamente é necessário centralizar o segmento do sinal de entrada a ser identificado com as formas de onda de teste. Como os dois sinais *Biphase_L* têm igual duração, basta sincronizar um deles. Isto é feito usando-se uma versão de tempo discreto do algoritmo mencionado por Proakis [6] para sincronizar símbolos. No apêndice 1 está a representação do esquema descrito acima, com o código fonte do programa em linguagem G do Labview™.

A arquitetura do demodulador mostrada no apêndice está contida em uma outra arquitetura onde suas etapas são demonstradas através do fluxograma abaixo.

[5] SubVis são caixas de programação existentes na Linguagem G do LabView™, muitas vezes não é possível se ter acesso ao conteúdo dessas caixas (caixas pretas).

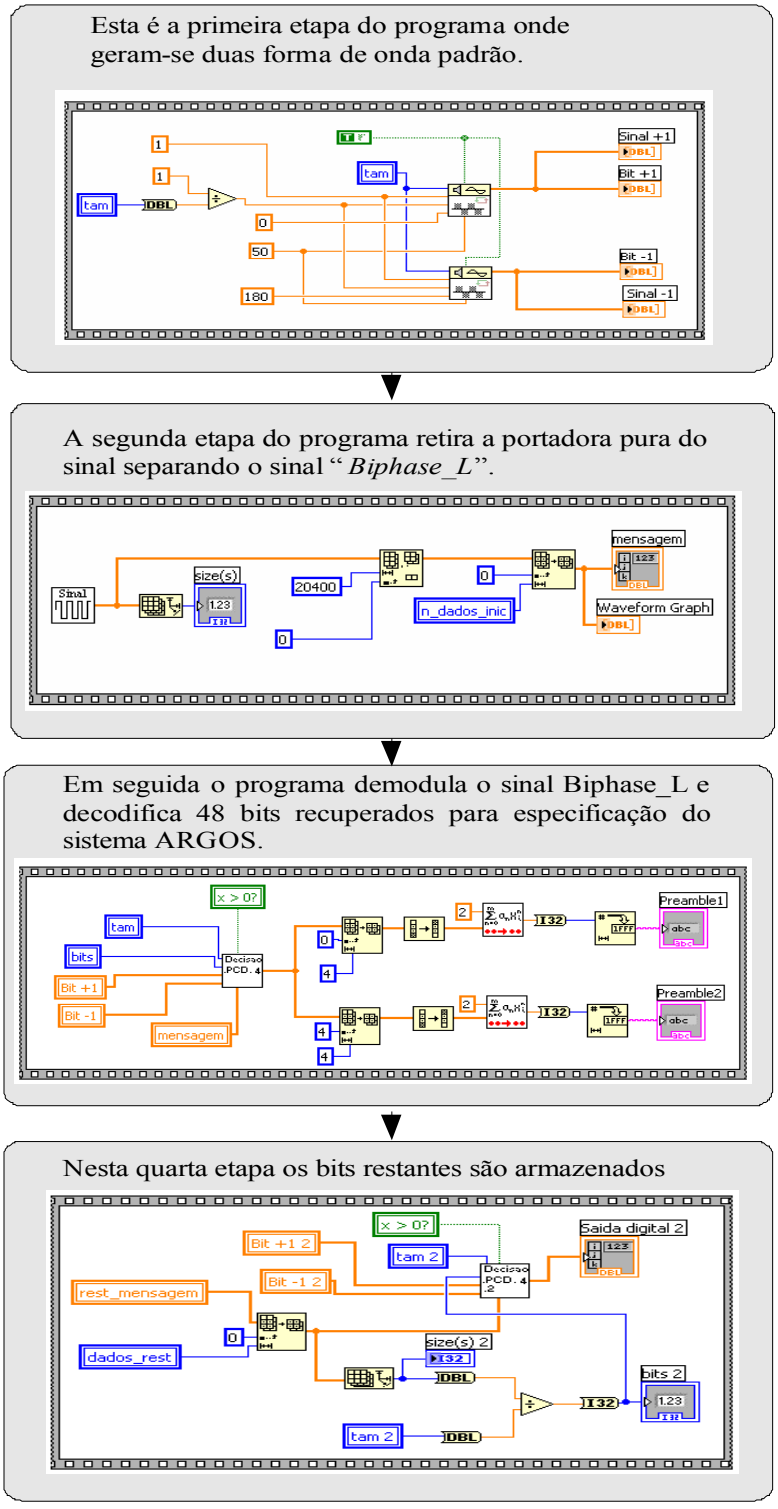


Figura 7 – Fluxograma do *software* desenvolvido que contém o Demodulador BPSK.

3.2 IMPLEMENTAÇÃO EM LINGUAGEM C/C++

Foi realizada a tradução do Demodulador em LabVIEW™ para linguagem C/C++ a fim de obter-se mais robustez. O LabVIEW™ é um aplicativo e, portanto possui alguns códigos que não possuem acesso ao código fonte, o que pode resultar em futuros problemas de portabilidade e adaptação do software, diferentemente da linguagem C/C++. Portanto, o código em C/C++ deverá ser mais fácil de manter funcionando em tempo real, devido inclusive ao fato de que o código compilado C/C++ é executado mais rapidamente que o código interpretado LabVIEW™. Um fato prático que mostra isso é que na tradução encontrou-se um erro no algoritmo de sincronismo (para corrigir o *jitter*) que não era evidente no funcionamento em LabVIEW™ devido à sua menor transparência. O princípio de funcionamento do detector é o mesmo do programa implementado na linguagem G. Baseia-se no uso de filtro casado com sincronismo para extrair os bits do sinal. No apêndice são mostrados o fluxograma desenvolvido para criação do código fonte (apêndices 2 e 3) e também o código fonte do programa desenvolvido em c/c++ (apêndice 4).

4 – RESULTADOS

4.1 DEMODULADOR BPSK EM LINGUAGEM G DO LABVIEW™

Com a implementação do demodulador BPSK foram feitas algumas simulações para testar sua funcionalidade. Este software foi testado em conjunto com o banco de homologação e atendeu satisfatoriamente a recuperação dos bits enviados das PCDs. Como o demodulador foi utilizado no bando de homologação do INPE, foram feitas algumas modificações no *software* em função das especificações do sistema Argos. Estas modificações seguem algumas características expressas no quadro 1. A seguir são demonstrados na figura 8 os resultados obtidos:

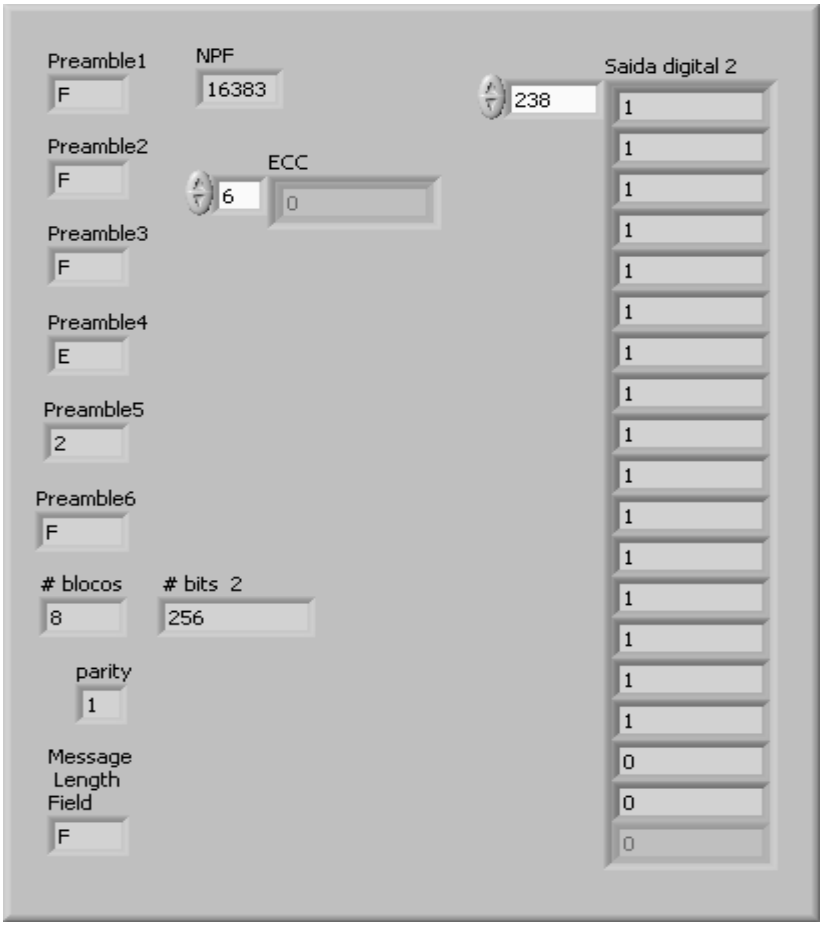


Figura 8 – Dados recuperados e utilizados no Banco de Homologação do INPE.

Na figura acima é possível observar que os dados recuperados coincidem com os valores especificados do padrão ARGOS indicado no quadro 1.

4.2 DEMODULADOR BPSK EM LINGUAGEM C/C++

Com a implementação do demodulador BPSK em linguagem c/c++ foram feitas algumas simulações para testar sua funcionalidade. O resultado foi a recuperação coerente dos dados, estes comparados com os dados anteriormente recuperados em LabVIEW™ coincidem, indicando recuperação correta dos dados. A figura abaixo demonstra o resultado obtido.

```
Valor Absoluto positivo  : -307.896
Valor Absoluto negativo  : -311.801
Valor Absoluto positivo2 : 307.896
Valor Absoluto negativo2 : 311.801
Diferenca dos valores absolutos = 3.90418



Valor Absoluto positivo  : -309.967
Valor Absoluto negativo  : -313.369
Valor Absoluto positivo2 : 309.967
Valor Absoluto negativo2 : 313.369
Diferenca dos valores absolutos = 3.402



Valor Absoluto positivo  : -311.801
Valor Absoluto negativo  : -314.684
Valor Absoluto positivo2 : 311.801
Valor Absoluto negativo2 : 314.684
Diferenca dos valores absolutos = 2.88367



Valor Absoluto positivo  : -313.369
Valor Absoluto negativo  : -315.705
Valor Absoluto positivo2 : 313.369
Valor Absoluto negativo2 : 315.705
Diferenca dos valores absolutos = 2.33612



Valor Absoluto positivo  : -314.684
Valor Absoluto negativo  : -316.445
Valor Absoluto positivo2 : 314.684
Valor Absoluto negativo2 : 316.445
Diferenca dos valores absolutos = 1.76107

Inicio de dados : 14758
Numero de bits analisado : 47
-Erro-
-Fechamento-

Valor Do BIT recuperado :  0

           - BITS RECUPERADOS -

11111111111111100010111111111111101111000000001000
```

Figura 9 – Término do processo de recuperação, com bits recuperados.

5 – CONCLUSÃO

O Instituto Nacional de Pesquisas Espaciais Centro Nordeste, oferece uma excelente estrutura para o estagiário desenvolver seu trabalho de inserção em empresas. Neste ambiente o estagiário passa por aprendizados constantes no dia-a-dia, tendo que trabalhar características como: planejamento para desenvolver o *software*, criatividade para resolver os problemas de programação que surgem, entrega de prazos de relatórios e resultados, relacionamento devido à constante necessidade de comunicar-se com o orientador e outras características mais, que são sempre exigidas no desenvolver da atividade. Os relatórios servem para documentação dos produtos desenvolvidos e conhecimento científico adquirido pela instituição. A documentação precisa incluir os fluxogramas dos programas de computador criados e a teoria usada para que os trabalhos desenvolvidos possam ser sempre usados e retomados. Esta capacidade de preservar a memória técnico-científica é fundamental para o INPE. Todos os processos descritos acima servem de experiência e enriquecimento profissional, adquiridos ao longo do estágio, os quais serão levados para vida profissional.

Em relação aos primeiros passos realizados no INPE, o estudo sobre os aspectos gerais do Sistema Brasileiro de Coleta de Dados foi importante para se ter idéia da abrangência do projeto. Em seguida iniciaram-se estudos da teoria de telecomunicações, de linguagem de programação, entre outras atividades necessárias para realização do trabalho. Após esse processo teórico deu-se início a implementação do Demodulador de BPSK em linguagem C/C++, ainda em desenvolvimento, contendo os filtros casados capazes de recuperar os bits enviados pelas PCDs.

Para o futuro espera-se que o SBCD, em particular o INPE, tenha um Demodulador de BPSK com recuperação de portadora implementado em linguagem C/C++, adaptável sem dificuldades a protocolos de comunicações e diferentes tipos de computadores onde for necessário ser instalado.

7 – REFERÊNCIAS

[1] *Centre National d'Etudes Sapatiales,* CNES. **ARGOS PLATFORM TRANSMITTER TERMINALS: General Specifications and certification.** Versão2: Julho 1988. 32p.

[2] *National Instruments Corporation*, NIC**. Introduction to LabVIEW CD-ROM.** 2004.

[3] *National Instruments Corporation*, NIC. (EBook - PDF - Engineering) **LabVIEW Function and VI Reference Manual**. Maio de 1997. 689p.

[4] *Rae, JCP*. **Detector de Sinais para os Satélites do Sistema Brasileiro de Coleta de dados usando Análise Espectral Digital.** Instituto Tecnológico de Aeronáutica, São José dos Campos. 2005. 123p.

[5] *Simon Haykin*. **Sistemas de Comunicação Analógicos e Digitais**. 4ª Edição.2004. 837p.

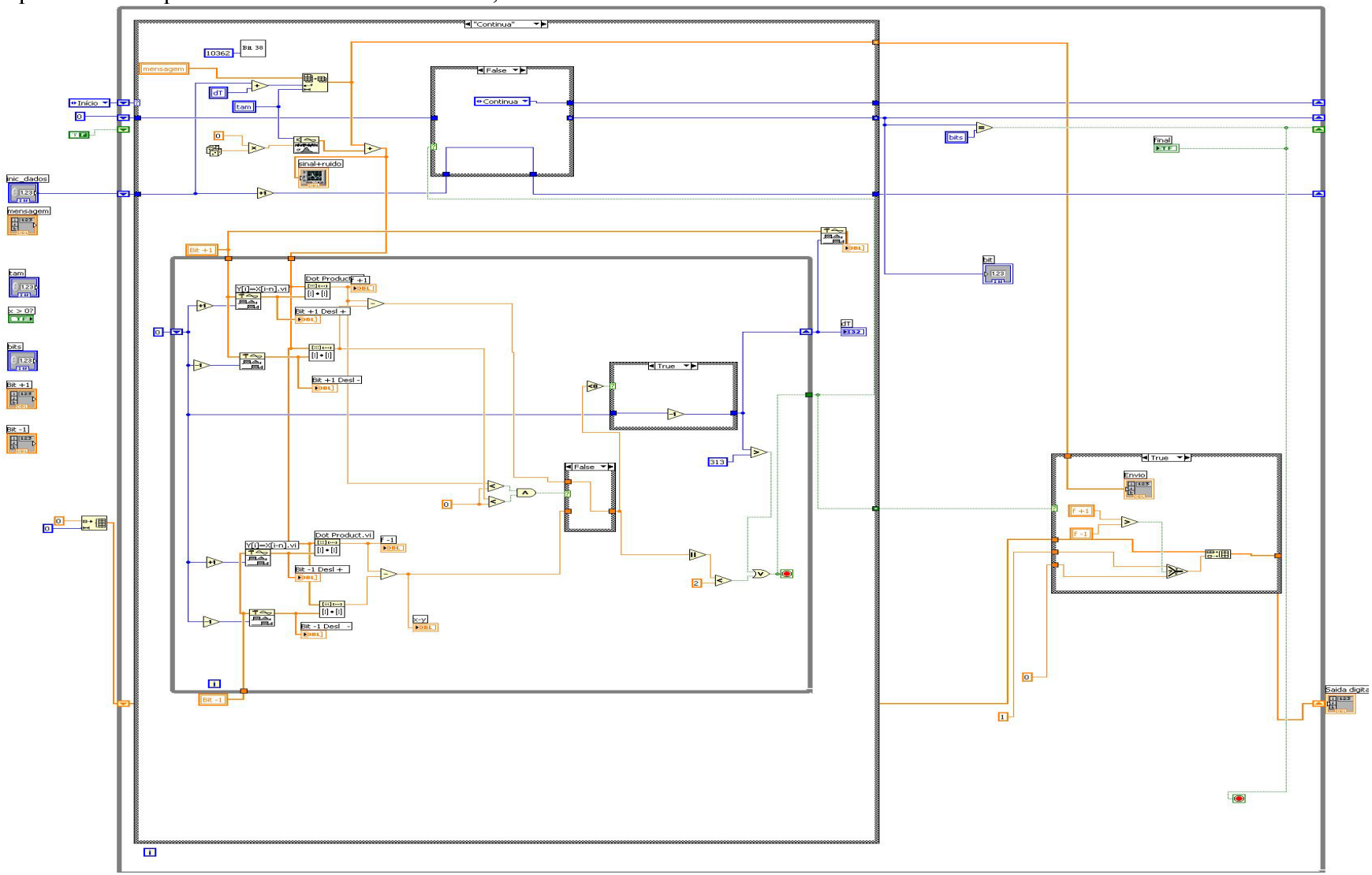
[6] *John G.Proakis.* **Digital Communications**.1983.608p.

[7] *José Luiz Boldrini*. **Álgebra Linear**. 3ª Edição.1980.411p.

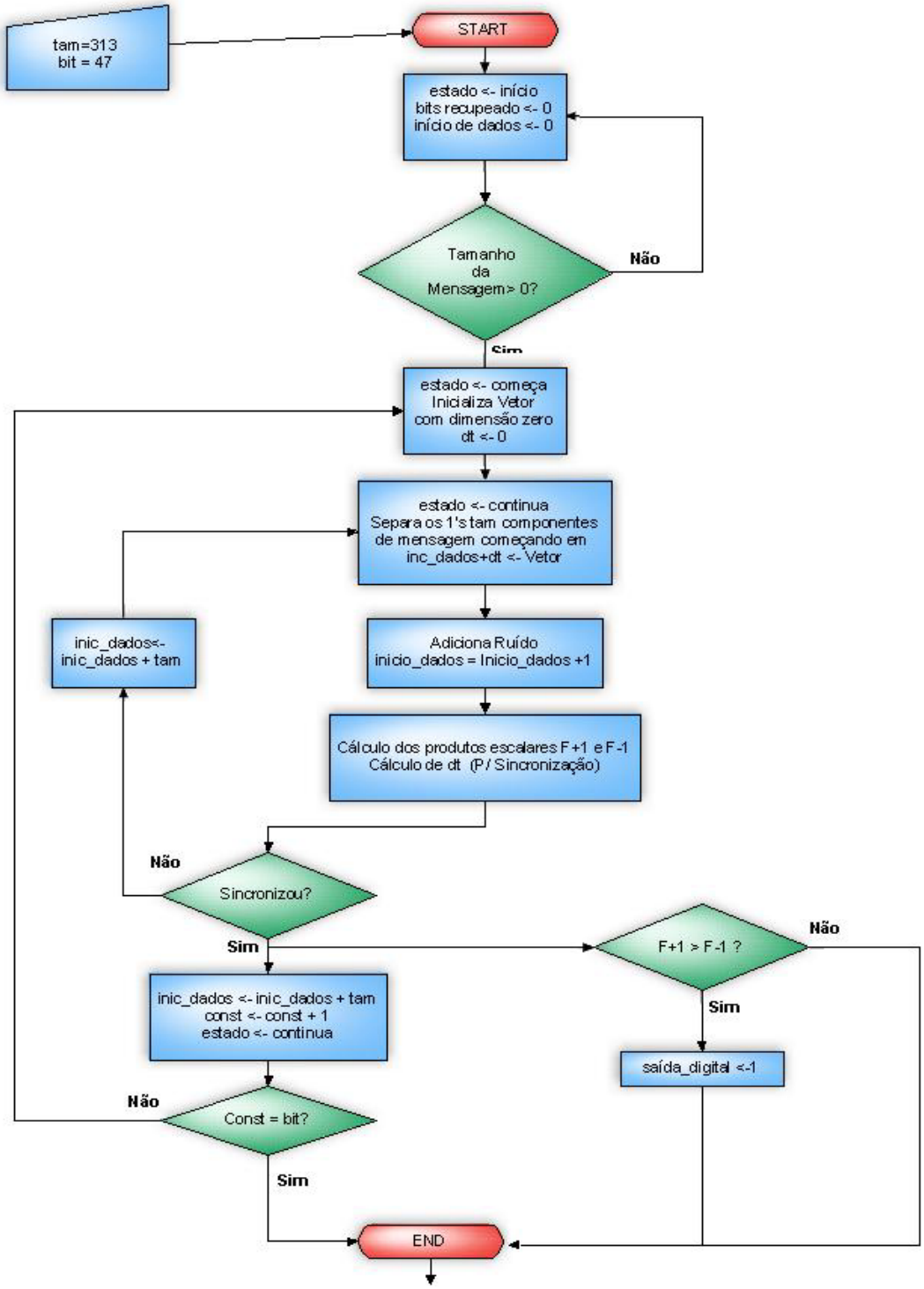
[8] *H*.M. Deitel. Como Programar em C++. 3ª Edição.2002.1098p.

APÊNDICES

Apêndice 1. Arquitetura do Demodulador BPSK, em LabVIEW™

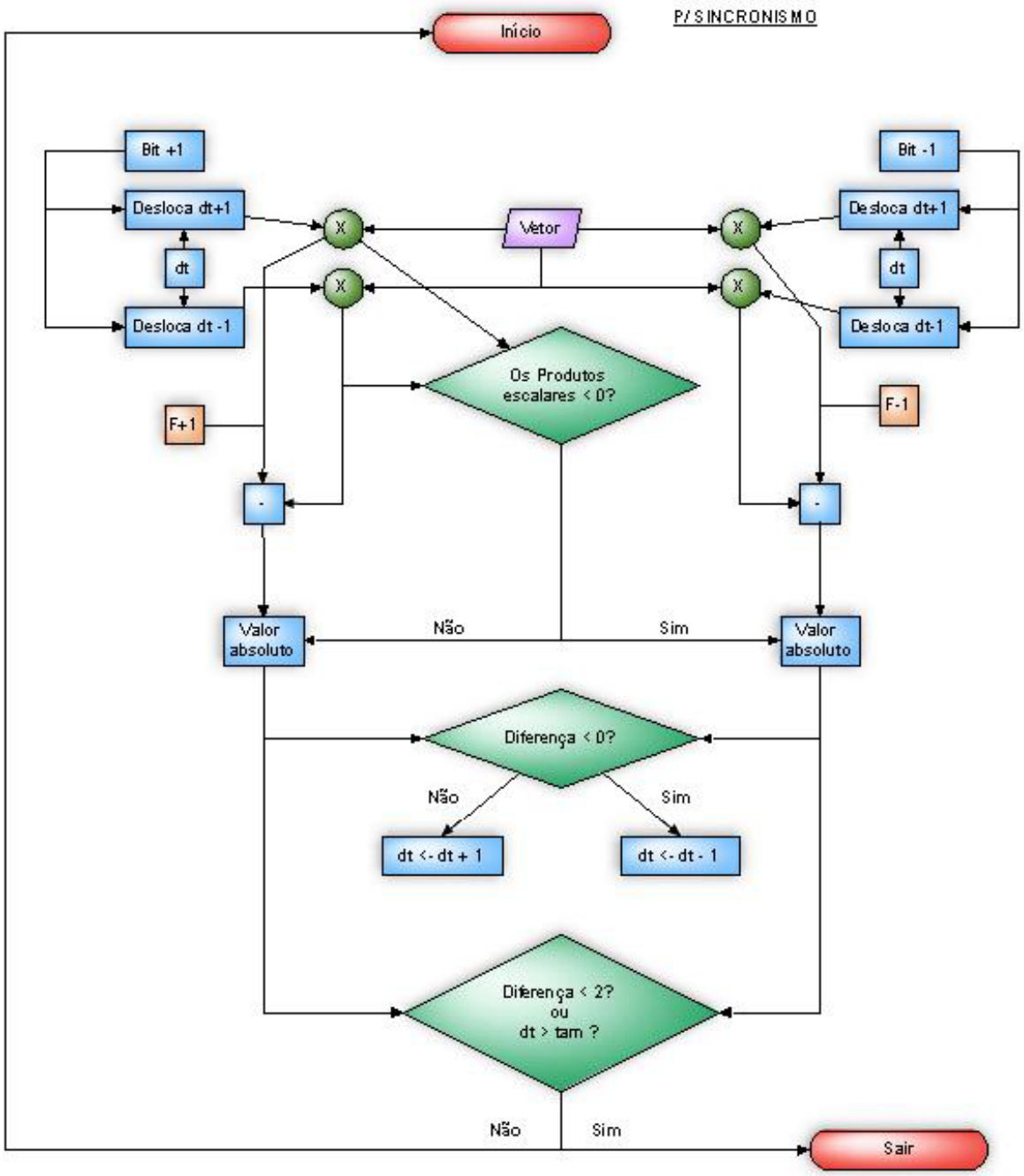


Apêndice 2. Fluxograma do detector binário desenvolvido em linguagem C/C++.



No fluxograma demonstrado acima a caixa que representa o cálculo dos produtos escalares F+1 e F-1 (sincronismo) são representados por outro fluxograma, demonstrado a seguir no apêndice 3.

Apêndice 3. Fluxograma do sincronismo.



Apêndice 4. Código Fonte do *software* desenvolvido em linguagem C/C++.

```
#include <cstdlib>
#include <iostream>
#include <math.h>
#include "fase1cont3.h" // Utilização da função onda_quadradapos e
onda_quadradaneg;
#include "msgteste3.h" // Utilização da mensagem utilizada em labview;
using namespace std;

int main()
{

    cout << "                        ------------------ " << endl;
    cout << "                        |DETECTOR BINARIO| " << endl;
    cout << "                        ------------------ " << endl;
    system("color 0f");
    int i=313,n,x,y=0,inic_dados=0,a=1,bit=0,tam=313;
    double valor_abspositivo,ypos[15024],yneg[15024],ydesp[15024],
      ydesp2[15024],ydesn[15024],valor_abspositivo2,ydesn2[15024],
      valor_absp[15024],valor_absp2[15024],valor_absn[15024],
      valor_absnegativo,valor_absn2[15024],dif=0,dif2=0,dif3=0,s[15024],
      valor_absnegativo2;
    int inicio=1,comeca=2,continua=3,erro=4,fechamento=5,continu;
    int isubset,k=1,dt,z=0,bita[48],ii=0;

do
{
    switch(inicio)//Malha de inicialização do decodificador BPSK
    {
    case 1:         cout << "-inicio- \n\n";
                    inic_dados;
                    cout << "Dados iniciais : " << inic_dados <<endl;
                    if (k>0)
                    cout << "bit :" << bit <<"\n\n";
    case 2:
                    cout <<"-Comeca- \n\n";
                    cout <<"Dados iniciais : " << inic_dados <<endl;
                    dt=0;
                    cout << "bit :" << bit << "\n\n";
    case 3:
                    continu: ;
                    cout <<"\n" << "-Continua- \n\n";
                    y=0;
                    dt=0;
                    n=0;
    for (i=inic_dados+dt; i<313+inic_dados; i++)
    {
    s[i]=xmsg[i]; // array subset do labview para separar a msg em vetores de
313 amostras;
    }

    for (i=inic_dados+dt; i<313+inic_dados; i++)
    {
    cout << s[i]; // array subset do labview para separar a msg em vetores de
```

```
313 amostras;
    }

    for (i=inic_dados,ii=0; i<313+inic_dados;i++,ii++)
    {
        ypos[i]= a * onda_quadradapos(ii);//Gera o vetor onda quadrada
positiva;
        yneg[i]= a * onda_quadradaneg(ii);//Gera o vetor onda quadrada
negativa;
        /*xmsg[i]=ypos[i];Faz com que a msg seja igual o vetor gerado para
comparação;*/
    }
    do
    {
    //Desloca o vetor de para frente ou para tras dependendo do n;
     for (i=inic_dados; i<313+inic_dados;i++)
    {
        x = n+1;
        if ((0<=i-x)&&(i-x<313+inic_dados))
        ydesp[i]=ypos[i-x];
        else
        ydesp[i]=0;
    }
     for (i=inic_dados; i<313+inic_dados;i++)
    {
        x = n-1 ;
        if ((0<=i-x)&&(i-x<313+inic_dados))
        ydesn[i]=ypos[i-x];
        else
        ydesn[i]=0;
    }
    for (i=inic_dados; i<313+inic_dados;i++)
    {
        x = n+1;
        if ((0<=i-x)&&(i-x<313+inic_dados))
        ydesp2[i]=yneg[i-x];
        else
        ydesp2[i]=0;
    }
      for (i=inic_dados; i<313+inic_dados;i++)
    {
        x = n-1 ;
        if ((0<=i-x)&&(i-x<313+inic_dados))
        ydesn2[i]=yneg[i-x];
        else
        ydesn2[i]=0;
    }
    for (i=inic_dados; i<313+inic_dados;i++)
    {
    // Gera o produto escalar e o valor absoluto do deslocamento positivo da
onda quadrada positiva;
        valor_absp[-1]= 0;
        valor_absp[i] = valor_absp[i-1]+(ydesp[i]*s[i]);
        valor_abspositivo = valor_absp[i];
        //cout <<  valor_abspositivo << "\n";
    }
```

```
    for (i=inic_dados; i<313+inic_dados;i++)
    {
    // Gera o produto escalar e o valor absoluto do deslocamento negativo da
onda quadrada positiva;
        valor_absn[-1]= 0;
        valor_absn[i] = valor_absn[i-1]+(ydesn[i]*s[i]);
        valor_absnegativo = valor_absn[i];
    }
    for (i=inic_dados; i<313+inic_dados;i++)
    {
    // Gera o produto escalar e o valor absoluto do deslocamento positivo da
onda quadrada negativa;
        valor_absp2[-1]= 0;
        valor_absp2[i] = valor_absp2[i-1]+(ydesp2[i]*s[i]);
        valor_abspositivo2 = valor_absp2[i];
    }
     for (i=inic_dados; i<313+inic_dados;i++)
    {
    // Gera o produto escalar e o valor absoluto do deslocamento positivo da
onda quadrada negativa;
        valor_absn2[-1]= 0;
        valor_absn2[i] = valor_absn2[i-1]+(ydesn2[i]*s[i]);
        valor_absnegativo2 = valor_absn2[i];
    }
        cout << "\n\n\n\n\n\n";
    //Valor absoluto do deslocamento positivo;
        cout << "Valor Absoluto positivo  : ";
        cout << valor_abspositivo << endl;
    //Valor absoluto do deslocamento negativo;
        cout << "Valor Absoluto negativo  : ";
        cout << valor_absnegativo << endl;
    //Valor absoluto do deslocamento positivo da onda quadrada negativa;
        cout << "Valor Absoluto positivo2 : ";
        cout << valor_abspositivo2 << endl;
    //Valor absoluto do deslocamento negativo da onda quadrada negativa;
        cout << "Valor Absoluto negativo2 : ";
        cout << valor_absnegativo2 << endl;
    //Diferença dos valores absolutos;
        dif = valor_abspositivo - valor_absnegativo;
        dif2 = valor_abspositivo2 - valor_absnegativo2;
        if ((valor_abspositivo > 0) && (valor_absnegativo) > 0)
           {
           cout << "Diferenca dos valores absolutos = " << dif << endl;
           dif3=dif;
           }//Verdadeiro é 1 e falso é 0;
        else
        {
           cout << "Diferenca dos valores absolutos = " << dif << endl;
           dif3=dif2;
        }//Verdadeiro é 1 e falso é 0;
        if (dif3 < 0)
        {
           n=n-1;
        }
        else
        {
```

```
            n=n+1;
        }
        y++;
    _sleep(5000);
    }while ((fabs(dif3)>= 2) && (n < 313));//Fechamento do 2 do while;
    //Lógica de uma parte estrutura continua da função case;
        cout << "\n\n";
        inic_dados = inic_dados + 1;
    if ((fabs(dif)>= 2) || y > 312)//Significa que não foi possível recuperar
o bit por algum motivo;
    {
    goto continu ; /*Caso a mensagem de um bit não consiga ser recuperada
    ou algum erro acontecer o goto manda voltar para o case 3 que é o continue
*/
        dt=n;
        inic_dados = inic_dados;
    }
    else
        bit = bit + 1;
        inic_dados = tam + inic_dados;
        cout << "Inicio de dados : " << inic_dados << endl;
        cout << "Numero de bits analisado : " << bit << endl;
    case 4:
        cout <<"-Erro- \n";

    case 5:
        cout <<"-Fechamento- \n";

        cout << "\n\n";
    //if ((fabs(dif)<=0) || (y > 313))
    //{
       if (valor_abspositivo > valor_abspositivo2)
          bita[z] = 1;
       else
          bita[z] = 0;
    //}
    cout << "Valor Do BIT recuperado :  " << bita[z];
    cout << "\n\n";
    z++;
}//fechamento do switch;
    }while (bit != 47);//Fechamento do 1 do while;

   for (z=0;z<48;z++)
   cout << bita[z];
   cout << "\n";
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

O programa utiliza dois arquivos cabeçalhos, a seguir será demonstrado apenas um desses que é o fase1cont3.h, que é utilizado como uma subrotina do programa principal, já que o outro arquivo msgteste3.h, serve apenas como mensagem para testar o programa.

Apêndice 5. Subrotina do programa principal, que gera a onda padrão em fase positiva e negativa.

```
int onda_quadradapos(int i)
{
    int x;
    if ( i <= (313)/2)
       x= 1;
    else if ((313)/2 < i <=313)
       x= -1;
    return(x);
}

int onda_quadradaneg(int i)
{
    int x;
    if ( i <= (313)/2)
       x= -1;
    else if ((313)/2 < i <=313)
       x=1;
    return(x);
}
```