9 3 (more or less)

ERIC CONRADO DE SOUZA

COMUNICAÇÃO CRIPTOGRAFADA ATRAVÉS DA TEORIA DE SISTEMAS CAÓTICOS

Trabalho de Formatura apresentado à
Escola Politécnica da Universidade de
São Paulo

São Paulo
Dezembro de 2000

ERIC CONRADO DE SOUZA

COMUNICAÇÃO CRIPTOGRAFADA ATRAVÉS DA TEORIA DE SISTEMAS CAÓTICOS

Trabalho de Formatura apresentado à
Escola Politécnica da Universidade de
São Paulo

Orientadores:
Marcelo Godoy Simões
Sílvio Szafir

São Paulo
Dezembro de 2000

Agradecimentos

Agradeço à Deus, sempre em primeiro lugar, pela ajuda e companhia, à Quem eu devo o meu esforço e dedicação.

Agradeço também ao Prof. Marcelo Godoy Simões pela oportunidade de ser seu orientado.

Ao Silvio Szafir, sempre disposto na sua ajuda e orientação.

Ao Prof. Newton Maruyama, por emprestar a EVM necessária à implementação.

Ao Nilson Noris Franceschetti e Newton César Omune.

Dedicatória

Gostaria de dedicar este estudo às pessoas que estivam mais próximas durante este trabalho e que sempre estarão do meu lado:

À Patrícia (musa inspiradora)

e a minha família

(Albino, Elaine, Aline, Andréa e Erisson).

Sumário

Introdução

No início da década de 90, quando a sincronização de sistemas caóticos foi observada pela primeira vez por Pecora – Carroll (1990), o reconhecimento do potencial destes sistemas para a comunicação levou ao surgimento de um campo distinto na área da dinâmica não – linear. Desde então, pesquisadores tem se dedicado no estudo e na aplicação de sistemas caóticos para comunicação segura, entre outras aplicações.

A teoria desenvolvida pelo estudo do Caos, através das ultimas décadas, permitiu ao homem uma nova visão da natureza. Inúmeros campos da matemática, física e astronomia, entre outras ciências, se beneficiaram das ferramentas surgidas com o estudo do caos. As aplicações desta teoria se estendem às áreas de telecomunicações (CDMA), fenômenos de convecção (transferência de calor), estudo do laser, biologia, química; e possivelmente para futuras aplicações, comunicação criptografada (*chaotic masking*).

O objetivo deste projeto é a implementação física do Sistema Caótico de Lorenz para Transmissão e Recepção, visando a comunicação segura de sinais. Essa implementação torna necessária a análise do quão adequado esse sistema se apresenta frente à sua utilização para comunicação.

Este relatório tem por finalidade apresentar as diversas análises, considerações, os estudos e resultados obtidos durante o projeto de implementação. A exposição dos itens abordados é realizada através de uma divisão didática em duas partes, à saber, o Estudo de Viabilidade - realizado para o cumprimento da primeira etapa do projeto do Trabalho de Formatura,

correspondente à disciplina PMC 580 – Projeto Mecânico I e o Projeto Básico - representando a segunda etapa do projeto do trabalho de Formatura, correspondente à disciplina PMC 581 – Projeto Mecânico II.

Na primeira parte será apresentada uma descrição da teoria envolvida, como também uma descrição dos requisitos de projeto e, por último, os resultados de simulações numéricas. A Segunda parte consiste no prosseguimento do Estudo de Viabilidade, com escopo voltado à implementação física do Sistema de Lorenz para comunicação, como também descrições sobre os aspectos tecnológicos em questão, sempre com a preocupação do seu desenvolvimento com base na Metodologia de Projetos [1].

O trabalho proposto foi baseado no artigo *Synchronization of Lorenz – Based Chaotic Circuits, with Applications to Communications* publicado no periódico mensal *IEEE Transactions on Circuits and Systems* de outubro de 1993 [2].

1. Fundamentos Teóricos

Nos parágrafos abaixo, uma caracterização do caos e de seus elementos será realizada. O propósito desta caracterização é somente chamar atenção do leitor para as diferenças entre os sistemas caóticos e outros sistemas.

O caos pode ser definido como um comportamento dinâmico limitado e que não possui solução em equilíbrio, solução periódica ou solução quasi – periódica [3].[1]

1.1 Sistemas Caóticos

Um sistema caótico pode ser representado por um sistema de equações matemáticas; e a solução deste a cada instante, ou para os valores de entrada da função, compõem os pontos ou estados da trajetória do sinal caótico num espaço – estado. Tem – se o interesse no estudo e análise deste sinal para que ele atenda ao uso de transmissão segura de mensagens.

Sistemas Caóticos podem ser definidos como sistemas dinâmicos não - lineares, determinísticos e imprevisíveis (*unpredictable)*, [4]. A não - linearidade é resultado da não aplicação do princípio da superposição. A qualidade de serem determinísticos se fundamenta na idéia de que dois sistemas caóticos partindo de estados ou condições inicias ligeiramente diferentes, possuirão comportamentos divergentes e imprevisíveis à priori. Mantidas, entretanto, as condições iniciais, o comportamento sempre será o mesmo. Estes sistemas são, portanto, sensíveis às

[1] As definições de sinais em equilíbrio, periódicos e quasi – periódicos são feitas no Apêndice B.

condições iniciais mas uma vez estabelecidas, a impossibilidade de se prever a evolução dinâmica do sistema é fato e por isso são imprevisíveis [1], [4] e [5].

Os sinais caóticos correspondem às respostas dos sistemas utilizados e podem ser empregados numa determinada atividade. São, também, determinísticos, possuindo uma taxa de redundância infinita e uma taxa de entropia finita e positiva [6].

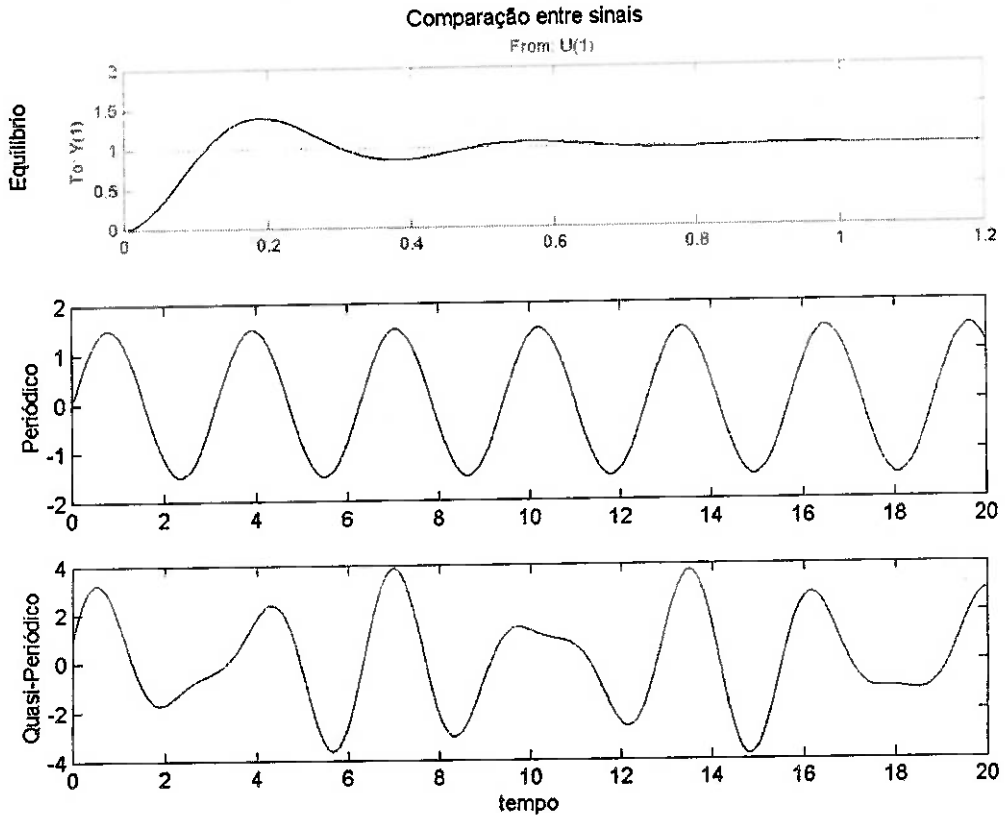


Figura 1: Soluções em equilíbrio, periódica e Quasi - periódico.

Em contraste com o espectro de sinais periódicos e quasi – periódicos, Figura 1, o sinal caótico possui um caráter contínuo de banda - larga (*broadband)*; um exemplo ilustrativo é mostrado na Figura 2. Juntamente com essa componente de banda – larga, o espectro de um sinal caótico freqüentemente apresenta

pontas (ou saltos) que indicam a freqüência predominante do sinal; e sua dinâmica (*chaotic motion*) é a superposição de várias dinâmicas periódicas instáveis. O sistema caótico pode, por exemplo, permanecer por um breve intervalo de tempo com uma dinâmica periódica e depois passar para outra, diferente da anterior, Figura 2. Por esta razão, pela contínua evolução de comportamentos dinâmicos que o sistema caótico pode apresentar, num intervalo relativamente longo, uma impressão de randomissidade é produzida; apesar dos intervalos curtos em que o sistema permanece com uma dinâmica fixa e possuir uma certa ordem [3].

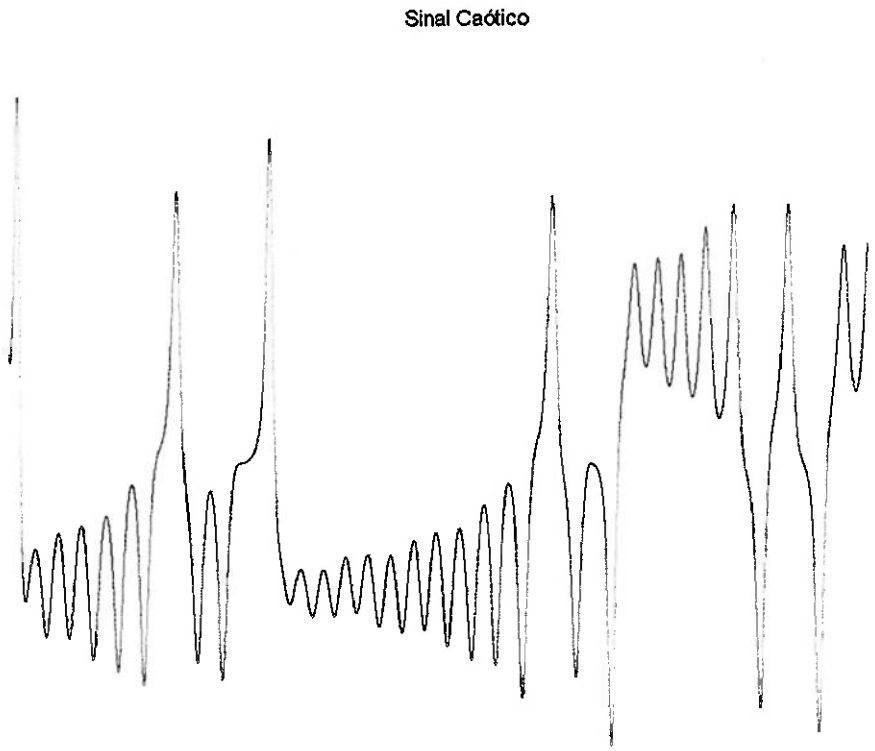


Figura 2: Sinal Caótico.

1.2 Sincronização

Existem grupos particulares de sistemas caóticos que se caracterizam por possuir uma propriedade especial denominada **Auto – sincronização**. Essa propriedade é observada quando o sistema pode ser decomposto em pelo menos dois sub – sistemas, à saber, um sub – sistema "acionador" ou transmissor (*drive system*) e um receptor (*response system)* que é sincronizado ao primeiro quando "acionado" pelo sinal de saída do sistema transmissor [2]. Atentar para o fato do primeiro sub – sistema possuir comportamento dinâmico independente do segundo, embora o segundo seja dependente do primeiro. Estes dois sub – sistemas podem ser combinados de modo a formar um sistema com igual dimensão de estados e com mesma resposta que o sistema original.

O resultado da sincronização é convergência dos valores de uma variável de estado do sistema transmissor e da variável correspondente ao sistema receptor. Em outras palavras, a teoria de sincronização de sistemas caóticos permite que dois sistemas distintos possuam mesma trajetória no espaço – estado quando o tempo tende ao infinito. Esta configuração é denominada Sincronização *Master – Slave* por Pecora - Carroll (1990).

Em particular, o sistema de **Lorenz** possui uma auto - sincronização robusta dos sub – sistemas mencionados; permitindo - a com condições iniciais relativamente diferentes.

O sistema de Lorenz consiste num conjunto de três equações diferenciais ordinárias (três dimensões no espaço de estados) dados por:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = r * x - y - x * z \\ \dot{z} = x * y - b * z \end{cases} \qquad (1)$$

Os símbolos σ, **r** e **b** são parâmetros que podem ou não ser constantes [7]. Estes parâmetros determinam o comportamento dinâmico do sistema. Com os valores mostrados abaixo, o comportamento do sistema de Lorenz se torna caótico:

$$\begin{cases} \sigma = 16 \\ r = 45{,}6 \\ b = 4 \end{cases}$$

Para o sistema de Lorenz, decompõe – se os sub – sistemas:

$$\begin{cases} \dot{y}2 = rx - y2 - xz2 \\ \dot{z}2 = xy2 - bz2 \end{cases} \qquad (2)$$

e

$$\begin{cases} \dot{x}1 = \sigma(y - x1) \\ \dot{z}1 = x1y - bz1 \end{cases} \qquad (3)$$

Observa – se que se $x(t)$, de(1), for a entrada de (2), e sua resposta, $y2(t)$ for usada como entrada para (3) então a nova saída $x(t)$ é semelhante à primeira entrada.

Para implementar a comunicação faz - se necessário um sistema transmissor e um receptor. Do sistema transmissor pode - se utilizar uma das variáveis de estado como gerador do sinal caótico que servirá como "onda portadora" do sinal ou mensagem a ser transmitida (a ser criptografada).

O sinal **m(t)** presente no sistema transmissor da figura abaixo, inserida no equacionamento, representa o sinal a ser transmitido ao receptor – o sinal a ser criptografado.

Os sistemas que serão implementados são mostrados no quadro abaixo:

Sistema Transmissor

$$\begin{cases} \dot{x}_t = \sigma(y_t - x_t) \\ \dot{y}_t = r * (x_t + m(t)) - y_t - (x_t + m(t)) * z_t \\ \dot{z}_t = (x_t + m(t)) * y_t - b * z_t \end{cases}$$

Sisema Receptor

$$\begin{cases} \dot{x}_r = \sigma(y_r - x_r) \\ \dot{y}_r = r * x_r - y_r - x_r * z_r \\ \dot{z}_r = x_r * y_r - b * z_r \end{cases}$$

O diagrama de blocos mostrado abaixo ilustra a configuração *Master – Slave* utilizada para comunicação essa idéia:

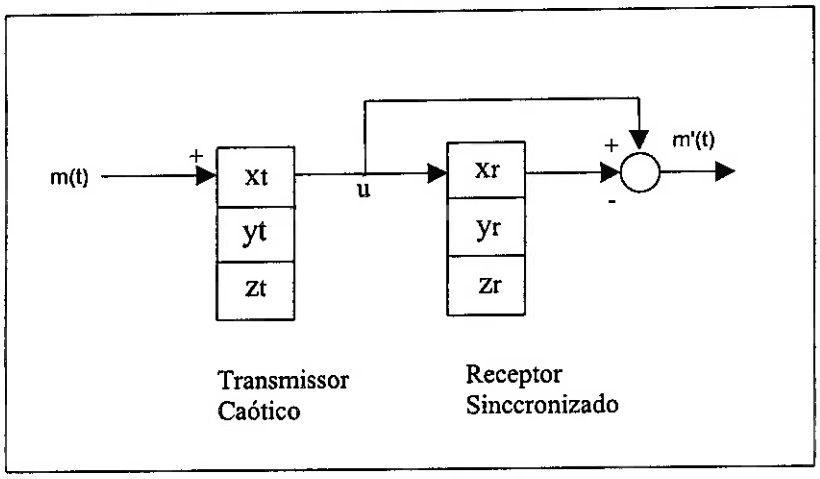


Figura 3: Sistema de Criptografia utilizando o Sistema de Lorenz.

Na figura acima, o sistema de recepção caótica é acionado pelo sinal proveniente do sistema de transmissão *u*. Nele sinal é embutida a mensagem *m(t)* – resultando num sinal composto. O receptor recebe esta composição de sinais e o utiliza para atingir a sincronização com sistema de transmissão. Após um

transitório inicial a sincronização é estabelecida e a componente sincronizada do sistema receptor é subtraída do sinal composto, reconstruindo, assim, uma mensagem semelhante à original – *m'(t)*.

Mas o que garante a sincronização? E em situações ela ocorre? Devido à complexidade na natureza não linear presente no sistema, não se sabe ao certo, o porquê do sincronismo entre os sub – sistemas. Entretanto, sua ocorrência, pode ser comprovada analiticamente através do erro dinâmico entre os sistemas transmissor e receptor.

Definindo o erro dinâmico do sistema de transmissão e recepção como

$$\begin{cases} X = x_r - x_t \\ Y = y_r - y_t \\ Z = z_r - z_t \end{cases}$$

encontra – se:

$$\begin{cases} \dot{X} = \sigma(Y - X) \\ \dot{Y} = -Y - Z(m(t) + x_t) \\ \dot{Z} = -bZ + Y(m(t) + x_t) \end{cases}$$

Escolhendo uma função de Liapunov:

$$V(X(t), Y(t), Z(t)) = 2X^2 + \sigma Y^2 + \sigma Z^2,$$

prova – se que:

$\frac{d}{dt} V(X(t), Y(t), Z(t)) \leq -\rho V(X(t), Y(t), Z(t))$, onde ρ é constante real e positivo [9].

Considerando que a função V não se anula para $t = 0$ e garantindo as condições para que a desigualdade se mantenha pode – se integrar a última expressão no tempo:

$$\Rightarrow \ln[V(X(t),Y(t),Z(t))]-\ln[V(X(0),Y(0),Z(0))]\leq -\rho t$$

$$\Rightarrow V(X(t),Y(t),Z(t))\leq e^{-\rho t}V(X(0),Y(0),Z(0))$$

Observa – se pela última expressão que a função de Liapunov fica limitada superiormente por uma exponencial com expoente negativo. Nota – se, também, que para o início da contagem do tempo, a função de Liapunov é igual à condição inicial. Portanto, o erro entre as variáveis dos sistemas vão à zero com o crescimento do valor do tempo, ou seja:

$V(X(t),Y(t),Z(t))\rightarrow 0, t\rightarrow\infty \Rightarrow$ Sincronização.

Em termos da mensagem, sinal a ser criptografado, tem – se:

$$m_r(t)=[x_t(t)+m(t)]-x_r(t)$$
$$\Rightarrow m_r(t)=[x_t(t)-x_r(t)]+m(t)$$

Quando da sincronização, a diferença entre variável utilizada como portadora da mensagem e sua correspondente no sistema receptor é nula. Logo:

$$\Rightarrow m_r(t)\cong m(t), t>>1.$$

A sincronização também é observada quando os autovalores da Matriz Jacobiana dos sub – sistemas forem menores que zero.

Uma importante pergunta pode ser colocada: o que faz do sistema de caótico, em especial de Lorenz, possível de aplicação na comunicação segura?

A sincronização é sem dúvida o grande trunfo, pois através dela pode - se extrair a mensagem do sinal portador caótico. Ademais, por produzir um sinal imprevisível, passando de um estado de equilíbrio para outro, fica claro que o sistema caótico não é um simples gerador de padrões (*patterns*), tornando complicada a sua decodificação por sistemas não - caóticos, Apêndice A. Na

tentativa de se "quebrar" essa criptografia por um sistema caótico, deve - se atentar para o fato de que várias configurações de codificação caótica são possíveis, seção 3.2, com vários níveis de complexidade, e muitas outras estão em estudo para torná-la comercialmente viável.

2. Projeto de Viabilidade

A metodologia empregada para este projeto segue as recomendações da disciplina PMC 475 – Metodologia de Projeto [1]. As várias etapas previstas pela metodologia são discutidas abaixo.

2.1. Estabelecimento da Necessidade

Diversas atividades hoje realizadas pelo homem, profissionais em especial, necessitam da preservação do conteúdo de sua comunicação em relação à terceiros. A manutenção da confiabilidade de informações constitui, portanto, em uma necessidade **real**.

No campo empresarial por exemplo, muitas vezes a exigência de privacidade de comunicação, entre matriz e filiais, se tornam imprescindíveis. Fica claro perceber que além de reais, as necessidades são **implícitas**.

2.2. Síntese das Necessidades

A necessidade a ser satisfeita é atender à comunicação segura de dados. Essa comunicação pode ser empregada pelos diversos canais, como a televisão por assinatura, rádios, telefonia convencional ou celular, Internet, comunicação entre empresas ou mesmo dentro de uma empresa...

O público a ser atendido pela implantação desta tecnologia é o mais abrangente possível, mais especificamente, a parcela da população que faz uso dos meios de comunicação acima mencionados.

A evolução desse tipo de comunicação frente o mercado deverá ser de constante aprimoramento nas técnicas de criptografia. Olhando o mercado com referência ao produto; as possíveis inovações de produtos já existentes ou mesmo novos produtos que porventura sejam lançados, poderão se utilizar da transmissão caótica.

2.3. Formulação do Projeto

2.3.1. Especificações Técnicas

A determinação dos requisitos, com relação ao produto - **sistema de criptografia** - composto por software e hardware, a serem atendidos são feitas abaixo:

2.3.1.1. Funcionais

Para aplicações em telefonia, em que a mensagem é a voz, o sistema de criptografia deve permitir a codificação de mensagens com freqüência de até 4kHz. A banda freqüência compreendida entre 100Hz e 4kHz corresponde ao da voz humana, e portanto. Estima – se que a freqüência de aquisição deve ser não menor que 8kHz, o que corresponde a uma velocidade mínima de 64kbits/s, utilizando 8 bits de resolução.

A velocidade de transmissão dependerá da modulação do sinal codificado e é função da aplicação. Para telefonia celular, a velocidade chega a 2Mbits/s.

O sistema não deve representar qualquer risco ao operador, ao que se refere à saúde ou vida, quando em operação; deve oferecer níveis de segurança iguais ou melhores outros meios ou métodos de comunicação. Em caso do

sistema operar de forma indesejada e / ou apresentar defeitos, o mesmo tem como função garantir segurança ao usuário.

2.3.1.2. Operacionais

A durabilidade ou vida útil é limitada à durabilidade dos componentes eletrônicos utilizados (conforme soluções do Relatório Parcial).

Para possuir confiabilidade de operação a criptografia deve representar difícil decodificação por terceiros (qualitativo).

2.3.1.3. Construtivas

As dimensões físicas máximas devem permitir a utilização do sistema com aparelhos de telefones celulares, *pagers*, rádios, *palm – tops*, etc. Estima – se que o sistema de comunicação possa ser implementado em um circuito integrado com 20mm x 20mm (largura x comprimento). O mesmo pode ser dito sobre o peso máximo; estima – se 200g.

A relação custo / benefício, para o usuário, de deve ser igual ou menor quando comparado com outros sistema de comunicação.

Todas as especificações mencionadas acima devem ser previstas para o sistema de comunicação caótica com igual ou melhor intensidade que os sistemas hoje existentes.

2.3.2. Técnica de Formulação de Características

Nesta etapa estuda – se o sistema ou produto de criptografia segundo suas entradas e saídas, conforme a Figura 4:

Figura 4: Sistema Caótico, suas entradas e saídas.

Existe a necessidade de determinar e quantificar as especificações técnicas através da verificação de entradas e saídas desejáveis ou não – desejáveis.

2.3.2.1. Entradas Desejáveis

- Energia elétrica;

- Transmissor:
  - Indicação de funcionamento pela habilitação do usuário e / ou da rede de comunicação: aquisição da mensagem e codificação;
  - Sinais ou mensagem com banda de freqüência abaixo da banda da voz;
- Receptor:
  - Indicação de funcionamento pela habilitação da operadora de telecomunicações, usuário e / ou rede de comunicação: recepção da mensagem criptografada;
  - Sinais com banda de freqüência abaixo da banda do sistema caótico.

2.3.2.2. Saídas Desejáveis

- Transmissor:
  - Mensagem criptografada com velocidade de transmissão correspondente ao meio de comunicação em utilização;

- Receptor:
  - Mensagem decodificada.

2.3.2.3. Entradas Indesejáveis

- Choques físicos ou térmicos;
- Líquidos ou umidade;
- Interferência eletromagnética.

2.3.2.4. Saídas Indesejáveis

- Consumo excessivo de energia elétrica (aparelhos portáteis);
- Calor;
- Radiação nociva ao homem.

2.4. Síntese de Soluções

As várias soluções levantadas no Relatório Parcial são discutidas neste item com mais detalhes além da ênfase na preocupação de atender aos requisitos estipulados nos itens anteriores. Os recursos necessários à implementação das soluções são, também, apresentados.

2.4.1. Hardware

Para implementação da comunicação caótica faz – se necessário a utilização de um dispositivo físico capaz de executar o sistema de criptografia e manipular os sinais gerados por este sistema, assim como a mensagem de interesse a ser comunicada. Esta função pode ser desempenhada tanto por um computador pessoal como por um sistema dedicado, como um circuito eletrônico analógico ou através do uso de microprocessadores (circuitos digitais).

A opção que se utiliza da eletrônica analógica representa uma solução que oferece pouca flexibilidade quanto à mudança de parâmetros do sistema caótico, uma vez que os elementos semicondutores, como resistores, capacitores e / ou diodos, haveriam de ser trocados eventualmente para testes ou mesmo quando do uso de algoritmos com variação de parâmetros.[2] Outro inconveniente se deve ao aumento de complexidade quando, por ventura, se deseje recuperar um sinal, como através de arquivo no PC; ou ainda realizar um processamento do sinal, como, por exemplo, uma análise espectral do sinal gerado pelo sistema.

As vantagens do uso da eletrônica digital em relação à analógica são resumidas abaixo:

- Aumento da densidade e velocidade e ao mesmo tempo diminuição do custo e tamanho dos CI's;
- Possibilidade de criar e modificar funções de processamento de sinais via software. Em sistemas analógicos seria necessário a modificação de hardware (resistores, capacitores, amp. op's, etc.) para haver mudanças significativas do funcionamento do sistema;
- Menos sensíveis às condições ou mudanças ambientais;
- Insensíveis às tolerâncias de componentes: mesma entrada corresponde a uma mesma saída (resposta). O comportamento dos sistemas analógicos: mesma entrada corresponde à saída ligeiramente diferente.

[2] Em [7] é mostrado um estudo da Sincronização dos Sistemas Caóticos com Variação dos Parâmetros.

Confrontando as outras duas soluções, uso do computador pessoal e o de microprocessadores dedicados, nota – se que o primeiro se sobressai devido à grande versatilidade de operação. Entretanto, os sistemas dedicados hoje existentes no mercado prevêem uma gama variada de funções pré – programadas, o que facilita consideravelmente a implementação. Além disso, a implementação do sistema caótico nestes sistemas constitui, senão o primeiro passo, pelo menos uma boa estimativa do funcionamento real de novos métodos de comunicação pelos diversos canais hoje existentes: telefones celulares, *pagers*, rádios, comunicação através da Internet com *palm – tops* (e – mails), comunicação intra ou inter – empresarial, etc. Esta afirmação não só é verdadeira como, também, a grande maioria destes equipamentos faz uso de processadores dedicados ao processamento de sinais, os DSP – Processadores Digitais de Sinais, discutidos nos próximos itens.

2.4.1.1. Estudo de Processadores

Este estudo foi realizado com o intuito de se determinar o processador que melhor se adequa ao conjunto de especificações técnicas estabelecidas mas que não, ao mesmo tempo, estivesse sobre – dimensionado em relação às mesmas especificações.

Com o rápido avanço tecnológico ocorrido, os Microcontroladores constituem em uma possibilidade real de escolha. Estes processadores incorporam algumas funções de I/O, contadores (*timers*) e funções específicas para manipulação de dados, entre outras, e que não estão presentes na maioria processadores. São desenvolvidos como processadores de propósito geral (PPG) e, portanto, seriam pouco eficientes no caso de serem utilizados para um

aplicação específica em tempo real, por exemplo, pois não são previsíveis quanto ao tempo de execução.

Um outro grupo de processadores é apresentado à seguir.

2.4.1.2. DSP

Os DSP são microprocessadores projetados especificamente para desempenhar o processamento de sinais digitais. O que se entende por processamento de sinais digitais é o conjunto de operações matemáticas necessárias à manipulação de sinais. Muitas vezes esses sinais devem ser processados em tempo real, como na telefonia celular digital, em modems, controles servo de *disk drives*, multimídia nos computadores pessoais, *home theater*, etc. Em vista disto, os DSP são arquitetados para permitirem execuções rápidas, através do paralelismo de atividades permitindo maior número de instruções por ciclo de instrução. Possuem:

- Unidade MAC – Multiplicação e Acumulação em um ciclo de instrução;
- MAM – Múltiplo Acesso à Memória, em que o operando pode ser carregado ao mesmo tempo em que se faz acesso à memória para busca de uma *word* de instrução;
- Funções de endereçamento especiais à memória (SMA – *Special Memory Addressing*), de controle de execução do programa (PFC – *Program Flow Control*) e de aceleração de operações repetitivas;
- Periféricos *on – chip*, interface eficiente com DAC, ADC e memórias.

O emprego destes processadores cresce rapidamente desde a chegada destes no mercado no início da década de 80, os DSP da primeira geração. Em contra partida, o inverso ocorre com o custo de aquisição destes; com valores

cada vez mais baixos e comparáveis com os custos de processadores de propósito geral ou micro controladores. Ao mesmo tempo, passam por uma contínua e rápida evolução – estando já na quinta geração – permitem sua utilização em aplicações cada vez mais variadas no processamento de sinais.

A Tabela 1, mostrada abaixo, foi elaborada com características correspondentes dos dois processadores já comentados:

Tabela 1: Diferenças entre DSP e Micro controladores.

	DSP	PPG
Path de Dados	Todas operações aritméticas feitas em 1 ciclo	Multiplicações são realizadas em mais de 1 ciclo
	Suporte de Hardware para shits, guard bits e saturação	Shifts, rounding, saturação >1 ciclo
Conjunto de Instruções	Especializado, instruções complexas	Instruções de Propósito Geral
	Operações múltiplas por instrução	Tipicamente uma instrução por ciclo
Arquitetura de Memória	Harvard	Von Neumann
	2 – 4 buscas por ciclo	Tipicamente 1 busca por ciclo
Endereçamento	SRAM *on – chip*	Uso de *caches*
	AGU (gerador de endereços) dedicado	Sem unidade de geração de endereço dedicado

Pelo apresentado, observa – se que os DSP oferecem maiores recursos e funções de processamento necessário ao sistema de comunicação caótica. Além disso possuem instruções específicas para o tratamento de sinais, permitindo códigos simples e concisos.

Descartando os micro controladores como candidatos, fica estabelecida a adoção dos DSP como solução. Os critérios de comparação para escolha do modelo são listados abaixo:

- Desempenho (ou Performance): Combinação do tempo de execução de uma determinada tarefa levando – se em conta o uso da memória e consumo de energia;

- Custo;
- Consumo: geralmente para aplicações em comunicação sem fio, celulares ou *pagers*, que se utilizam de baterias, o consumo se torna um critério de importância. Em função disto os produtores de DSP tem reduzido as tensões de alimentação como também incluído uma série de dispositivos incorporados: divisores de freqüência, controle dos periféricos em uso (permitindo o desligamentos destes quando não utilizados) e *sleep modes*;
- Facilidade de Programação: por oferecer aplicativos específicos de simulação e depuração (*debugger*), como também, placas de desenvolvimento;
- Formato Numérico dos dados manipulados:
  - ponto fixo: são mais baratos, precisa – se conhecer o extensão dinâmica dos dados e determinar a resolução necessária através de simulações ou de análises analíticas;
  - ponto flutuante: mais caro pois necessitam de maior complexidade de hardware na unidade de processamento;
- Resolução dos dados:
  - ponto flutuante sempre 32 bits: mais caro pois influenciam no número de pinos e no tamanho do CI, assim como o tamanho dos dispositivos de memória conectados ao DSP;
  - ponto fixo: 16, 20 ou 24 bits;
- Configuração da Memória
  - Von Neumann: somente um barramento para dados e endereço;
  - Harvard: barramentos e memórias independentes.

Apresenta – se abaixo uma tabela comparativa de alguns critérios de vários modelos de DSP e agrupados de acordo com o fabricante.

Tabela 2: Fabricantes e Modelos de DSP no mercado.

	Aritmética: Ponto...	Represent. de Dados / Instrução*	*Clock Speed* (MHz)	Espaço para Endereçamento Dados / Prog.	Tensão de Alimentação (volts)	Preço US$
Analog Devices						
ADSP-21xx	Fixo	16 / 24	75	16K / 16K	2.5 - 5.0	5 - 79
ADSP-2106x	Flutuante	40 / 48	60	16M / 4G	3.3; 5.0	20 - 381
ADSP-2116x	Flutuante	40 / 48	100	4G / 4G	2.5; 3.3	138
Hitachi						
SH-DSP	Fixo	16 / 32	66	4M / 128K	3.0	25
IBM						
C54XDSP	Fixo	16 / 16	66	-	-	-
Lucent Technologies						
DSP16xx	Fixo	16 / 16	120	64K / 64K	3.0; 3.3; 5.0	5 - 75
DSP16xxx	Fixo	16 / 32	100	1M - 1M	3.3	58
Motorola						
DSP560xx	Fixo	24 / 24	47.5	64K / 128K	5.0	8 - 21
DSP563xx	Fixo	24 / 24	100	16M / 32M	3.3	20- 53
DSP566xx	Fixo	16 / 24	70	64K / 128K	1.8 ;2.5; 3.0	15 - 60
DSP568xx	Fixo	16 / 16	35	64K / 64K	3.0; 3.3	6 - 10
Texas Instruments						
TMS320C1x	Fixo	16 / 16	8.8	4K / 256	3.3; 5.0	-
TMS320C2x	Fixo	16 / 16	12.5	64K / 64K	5.0	-
TMS320C2xx	Fixo	16 / 16	40	64K / 64K	5.0	5 - 16
TMS320C27xx	Fixo	16 / 16	50	4M / 4M	3.3	-
TMS320C3x	Flutuante	32 / 32	40	16M / 16M	3.3; 5.0	10 - 180
TMS320C4x	Flutuante	32 / 32	30	4G (ambos)	5.0	69 - 177
TMS320C5x	Fixo	16 / 16	50	64K / 64K	3.3; 5.0	11 - 35
TMS320C54x	Fixo	16 / 16	100	64K / 64K	2.5; 3.3; 5.0	20 - 27
TMS320C62xx	Fixo	16 / 32	200	16M / 16M	1.8; 2.5; 3.3	90 - 121
TMS320C67xx	Flutuante	32 / 32	167	16M / 16M	1.8; 3.3	143

2.4.2. Aplicativos

Os aplicativos de simulação e de análises matemática e gráfica serão utilizados para verificar tanto a veracidade como a eficiência do comportamento do sistema de comunicação caótica. As ferramentas de auxílio no desenvolvimento, como ambientes de programação e depuração, serão empregadas.

Com a escolha dos DSP como solução, é importante mencionar que os aplicativos de compilação e *linkagem* de programas ou mesmo o ambiente de programação e depuração são específicos ao modelo (ou família) utilizado.

3. Testes e Simulações

3.1. Software

Uma consideração muito importante a ressaltar é com respeito ao algoritmo numérico utilizado, necessário à integração das variáveis de estado dos sistemas caótico de transmissão e recepção.

Para tanto empregou - se integração numérica segundo o método de Runge - Kutta de 4ª ordem. Trata - se de um método de passo simples, em que o conhecimento apenas da informação local se faz necessário, ou seja, um (1) intervalo de integração compreendido em $[x_i , x_{i+1}]$; requerendo apenas derivadas de primeira ordem. Esse método, embora exija um número relativamente maior de operações se comparadas com os de 2ª e 3ª ordem, permite aproximações precisas com erros de truncamento, erros do algoritmo numérico, da ordem de h^4, onde h é o passo de integração.

Verificou - se também , através das simulações, a não necessidade do uso de controle sobre o passo de integração, como o critério de Collatz, por exemplo.

A implementação no ambiente MatLab, em primeira instância, teve como principal conseqüência a verificação da sincronização do uso dos sistemas caóticos como algoritmos adequados à comunicação, criptografando o sinal a ser enviado ao receptor. Além disso o ambiente permitiu tanto a utilização de uma sintaxe fácil como também de rotinas de visualização gráfica.

O diagrama de Nassi - Schneiderman do algoritmo de simulação numérica é apresentado abaixo:

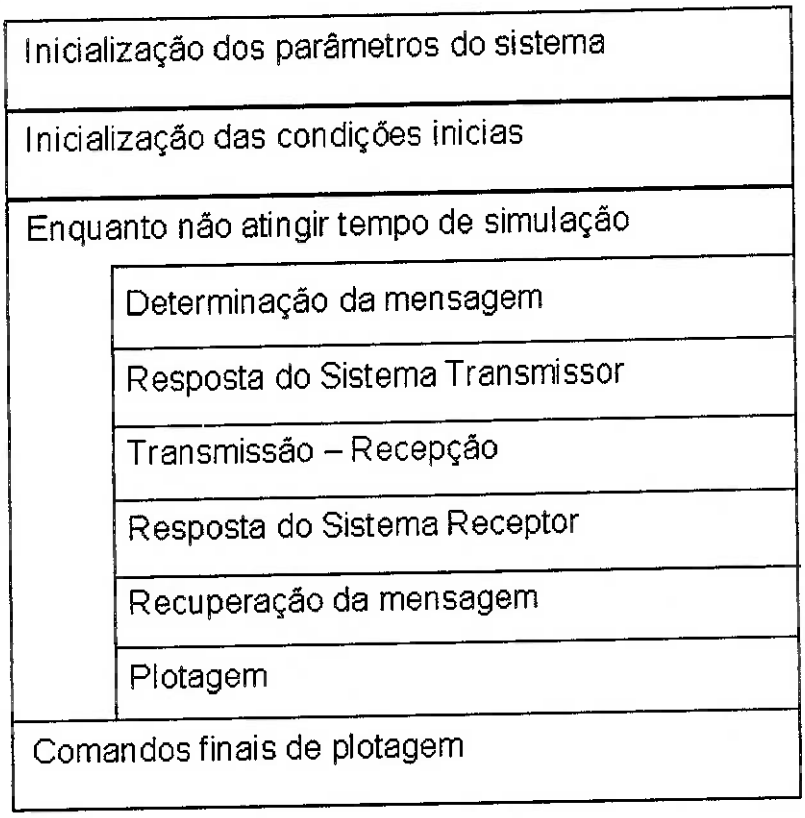


Figura 5: Diagrama de Nassi - Schneiderman.

Na ilustração à seguir, pode - se verificar de maneira mais detalhada do como a implementação foi realizada com o algoritmo acima. Nesta figura, os blocos verdes corresponde a integração da variável *x*; os vermelhos, y e os azuis, z. A mensagem é inicialmente introduzida no equacionamento do sistema transmissor (esquerda), somada ao sinal portador caótico, formando o sinal composto, e então transmitida. No lado direito, o sinal é sincronizado no sistema receptor e como resultado, a recomposição da mensagem.

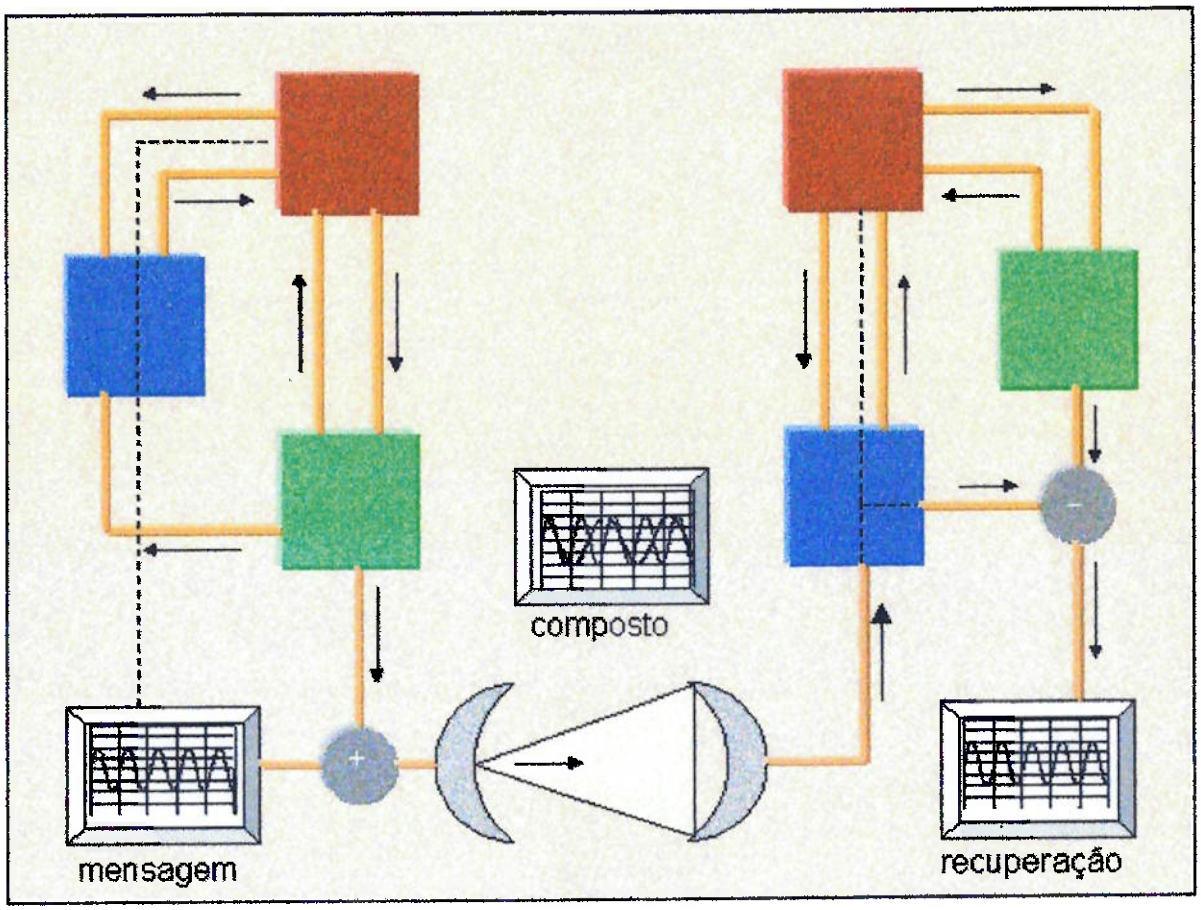


Figura 6: Sistema de Comunicação Caótica.

3.2. Resultados

O algoritmo testado contém os sistemas de transmissão e de recepção. Nota - se as condições iniciais ligeiramente diferentes entre os dois sistemas: para o transmissor $(x_i, y_i, z_i) = (10.0;1.5;-6.5)$ e para o receptor $(x_i, y_i, z_i) = (1;1;1)$.

Os resultados da sincronização e da recuperação da mensagem original são, agora, apresentados.

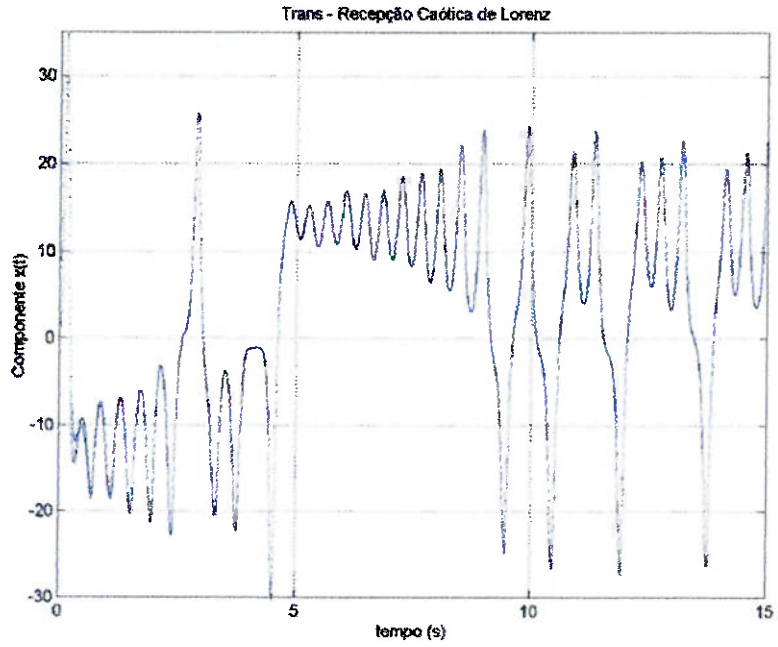


Figura 7: Sincronização dos sinais composto e gerado no receptor, passo de 0.002.

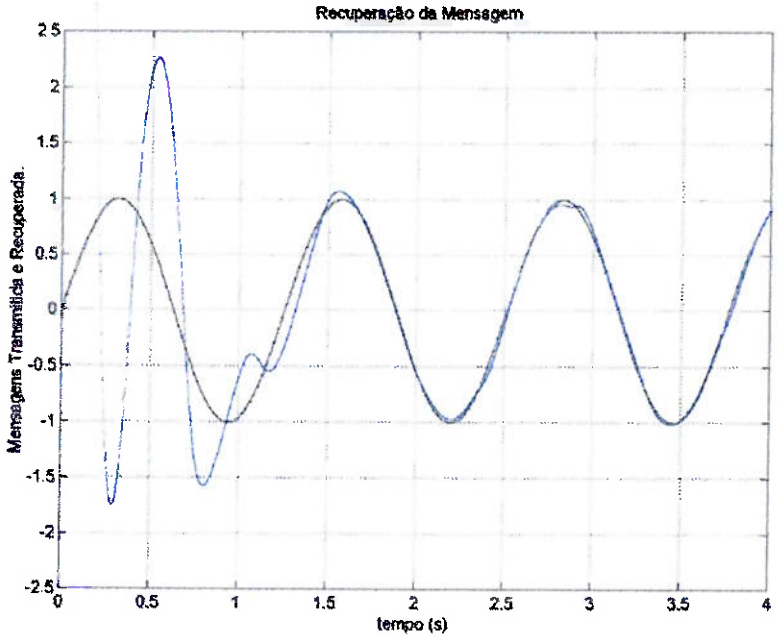


Figura 8: Recuperação da mensagem transmitida, passo de 0.0005.

Observa - se o transitório, comentado na seção 1.2, ou tempo necessário para consolidar a sincronização. Foi necessários aproximadamente 3000

interações, com as condições iniciais e passo mencionadas acima, para o seu estabelecimento.

Um outro esquema do sistema de Lorenz pode ser usado para o mesmo fim, segundo Pecora - Carroll. Nesta abordagem além do sinal composto transmitido um outro sinal, o *drive signal*, que corresponde a uma outra variável daquela utilizada para compor o sinal com a mensagem, é transmitido. Deste modo, a transmissão fica mais robusta pois os sistema transmissor e receptor não são iguais, como no caso anterior, precisando reconstruir o subsistema para decodificar a mensagem, figura abaixo.

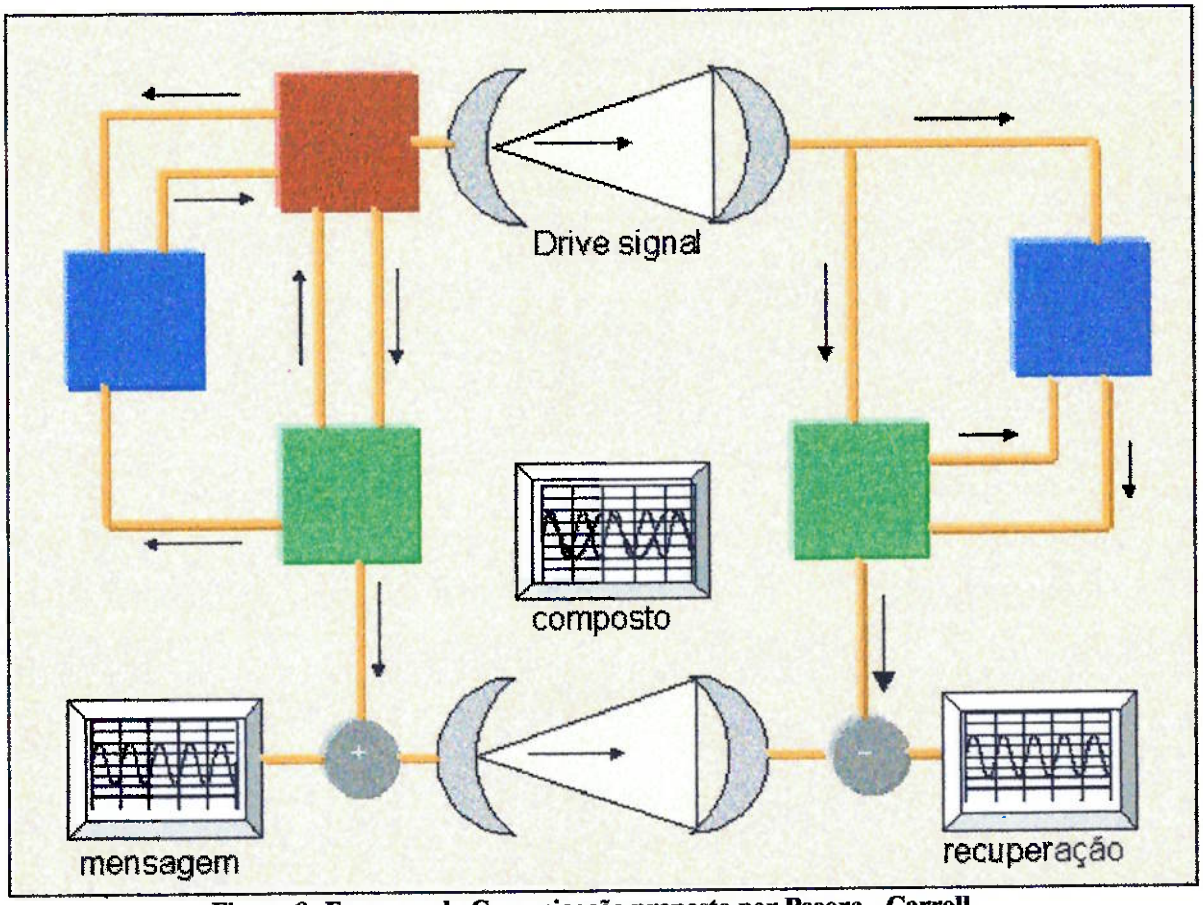


Figura 9: Esquema de Comunicação proposto por Pecora - Carroll.

Fazendo uma analogia ao uso de senhas com o *login*, esta configuração funciona semelhantemente; precisa de um sinal de acionamento para reconstruir o

subsistema, sincronizar e poder extrair a mensagem. Os resultados da simulação são apresentados a seguir, com mesmas condições iniciais:

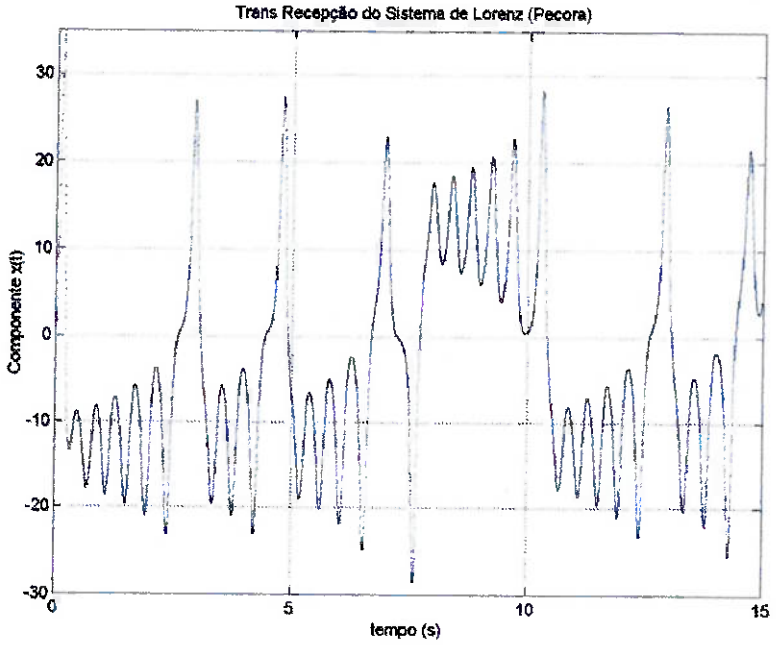


Figura 10: Sincronização dos sinais composto e gerado no receptor, passo de 0.002.

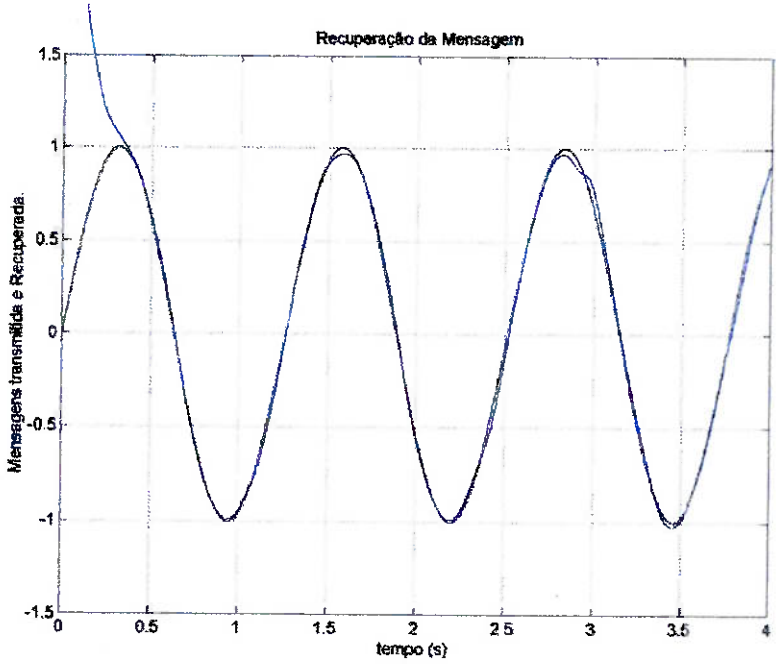


Figura 11: Recuperação da mensagem transmitida, passo 0.0005.

Pode - se observar que neste caso o transitório é visivelmente menor. Sincronizando com aproximadamente 700 interações, 5 vezes mais rápido que o primeiro esquema.

Em seguida uma análise da resposta em freqüência é realizada o fim de verificar se o sistema atende aos requisitos funcionais.

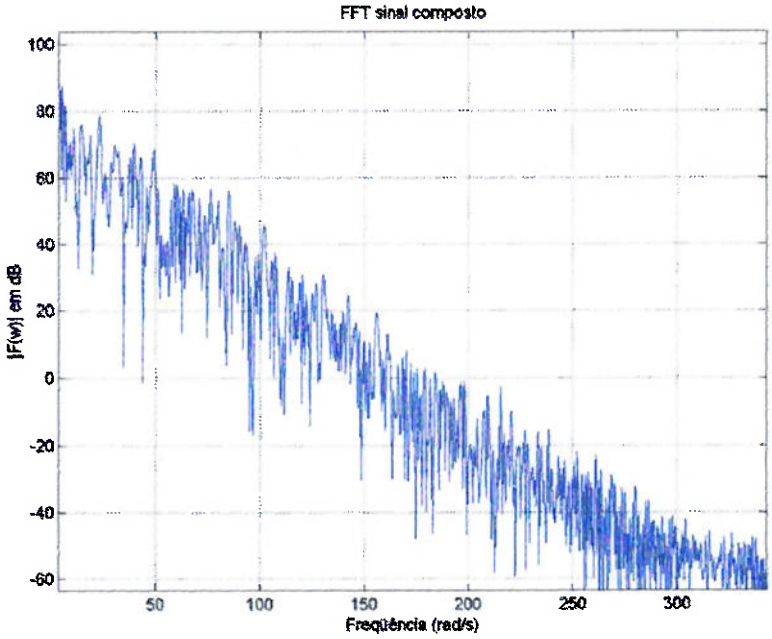


Figura 12: FFT do Sistema de Lorenz.

Comparando - se o resultado acima com o obtido por Kuomo - Oppenheim, observa - se que a resposta em freqüência deste último é maior segundo uma relação, aproximada, de 400 : 1.

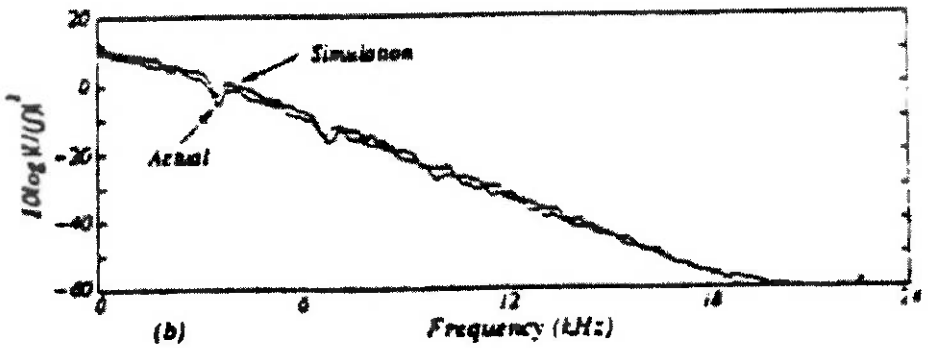


Figura 13: Resultado obtidos por Kuomo - Oppenheim.

Foi feita um linearização do sistema de Lorenz em torno da posição de equilíbrio com a finalidade de se saber a resposta em freqüência. O resultado é mostrado abaixo. Observa - se, a título de comparação, que para uma freqüência de ~100 rad/s (16Hz) ganho correspondente é de 40dB negativos.

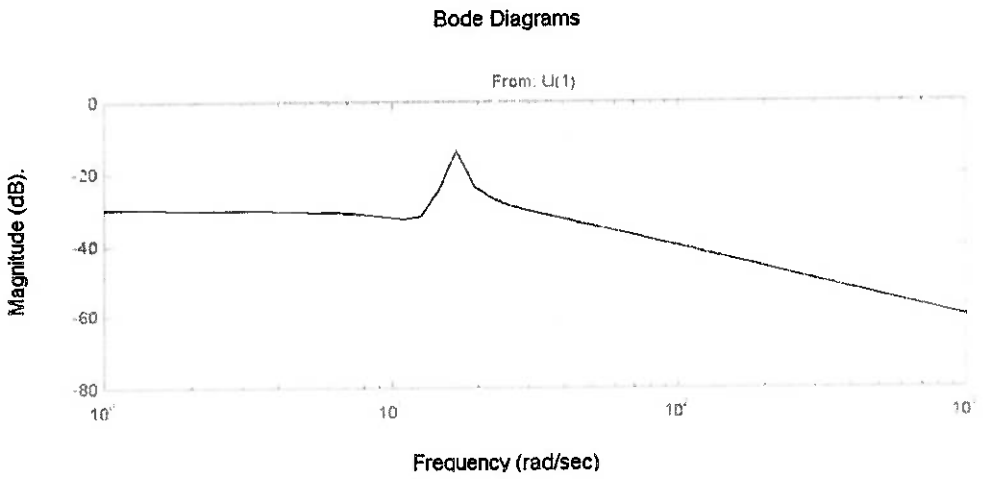


Figura 14: Resposta em freqüência do Sistema de Lorenz Linearizado.

Comparando - se as últimas três figuras, fica evidente que o resultado da linearização e do diagrama de bode possuem respostas de mesma ordem de grandeza. Estes dois gráficos estão aquém do obtido por Kuomo - Oppenheim; tem - se interesse na comparação destes com resultados práticos apresentados nos itens à seguir.

4. Projeto Básico

Uma vez definido a utilização por processador digital de sinais (DSP) como elemento processador, nesta parte serão abordados a análise e seleção do modelo e fabricante do DSP que será empregado na implementação do sistema como também as várias alternativas de como essa implementação será realizada.

4.1. Seleção do DSP

Algumas características são desejáveis no DSP, ente elas pode - se citar o formato numérico e resolução de dados convenientes, facilidade de programação - que envolve um suporte à programação através de um Ambiente de Programação adequado como também do uso de *kits* de avaliação do processador. Outro atributo importante é a velocidade de processamento - que fornece uma idéia da performance do DSP - através de uma comparação em capacidade de instruções por segundo utilizando como referência pelo mercado.

A facilidade de programação consiste no mais importante dos critérios de seleção. Isto se deve principalmente porque o programador deseja que o tempo dedicado ao desenvolvimento de sua aplicação seja o menor possível, o que acarretará, em parte, em menores custos de projeto. Além disso um bom Ambiente de Programação torna automático o desenvolvimento de algoritmos em que não se exige conhecimento profundo da linguagem montadora do DSP, ao mesmo tempo que auxilia no criação de códigos mais elaborados.

O formato numérico e a resolução dos dados do DSP estão relacionados com o custo do processador e suas ferramentas na grande maioria dos casos.

Quanto maior o número de bits empregado na representação de dados ou no caso do formato de *ponto flutuante* maior será o custo de aquisição do processador. No entanto, um processador com formato numérico em *ponto fixo* consegue manipular algebricamente números em ponto flutuante, graças a rotinas de transformação presente no Ambiente de Programação, assim como um processador *ponto flutuante*. Ademais, conforme verificado através de simulações numéricas realizadas no capítulo anterior, verificou – se que os dados variam entre os limites [-30;30]. Adotando – se uma resolução de dados de 16 bits consegue – se uma precisão de dados até a terceira casa decimal, o que pode ser considerado, neste projeto, como aceitável.

A velocidade de processamento consiste noutro ponto importante de consideração. Muitos fabricantes de processadores indicam os valores de *Clock Speed*, Mips ou Mops para a determinação de potência consumida de seus produtos. No entanto estas medidas não são adequadas para comparação da velocidade de diferentes modelos de processadores uma vez que o número de operações para cada instrução e o tempo de execução (freqüência de operação) necessário para uma instrução varia de modelo para modelo. Para uma determinação adequada da velocidade de processamento de um DSP outros atributos e procedimentos são analisados, entretanto este estudo situa – se fora do escopo deste trabalho.

Outros atributos, como consumo e custo - levados em conta no projeto de um produto, possuem menor importância em relação à natureza acadêmica deste trabalho e não serão comentados.

Considerando os vários aspectos apresentados acima e analisando vários fabricantes de DSP e seus respectivos modelos, define – se o modelo em questão. A título de ilustração uma Matriz de Decisão, tabela abaixo, poderia ser empregada no auxílio de escolha. Nesta tabela, a solução que tivesse maior soma de nota seria, teoricamente, a solução escolhida.

Tabela 3: Matriz de Decisão de Soluções - famílias de processadores de 16bits.

Atributo	peso	Alt. A		Alt. B		Alt. C		Alt. D		Alt. E	
		Texas Instr. TMS320C27xx		Texas Instr. TMS320C2x		Texas Instr. TMS320C5x		Motorola 568xx		Analog Devices ADSP-21xx	
		nota	n x p	nota	n x p	nota	n x p	nota	n x p	Nota	n x p
Custo	0,15	8		9		8		10		8	
Clock Speed (MHz)	0,15	8		7		8		7		10	
Formato Numérico	0,15	10		10		10		10		10	
Resolução de Dados	0,15	8		8		8		8		8	
Facilidade de Programação	0,30	9		9		9		9		7	
Memória para Desenvolvim.	0,10	8		8		8		8		8	
soma	1,00										

Com base na discussão acima e na consulta de fabricantes determina – se a escolha da família DSP56800 da Motorola; mais especificamente o modelo DSP56L811.

O DSP56L811 é um processador de propósito geral com uma arquitetura híbrida, reunindo características de um processador de sinais, através da existência em seu núcleo da arquitetura da família DSP56800 e incluindo a flexibilidade de instruções e módulos periféricos internos, encontrado nos microcontroladores. O DSP escolhido é um processador de 16 bits ponto fixo para representação de dados e endereços e opera numa freqüência máxima de 40MHz. Levando em conta que uma Possui uma arquitetura otimizada para

execução de suas instruções priorizando a utilização da linguagem C no desenvolvimento de aplicações. Ele é um dispositivo HCMOS de alta densidade possuindo pinos de saída padrão TTL e pinos de entrada CMOS. Algumas de suas características principais e especificações são mostradas abaixo:

- Núcleo de 16 bits – ponto fixo, 20 milhões de instruções por segundo (MIPS) a 40 MHz;
- Arquitetura Harvard, permitindo operações ou instruções com até três acessos simultâneos às memórias de programa e de dados;
- Duas portas, B e C de 16 pinos, de Propósito Geral (GPIO);
- Três *timers* programáveis de 16 – bit;
- Suporte de Interface Serial de Periféricos (SPI) – porta C;
- Suporte de Interface Serial Síncrona (SSI) – porta C;
- Até dez pinos reservados para interrupções externas;
- Porta para depuração independente – sem interferir no processamento do processador – do tipo JTAG/ OnCE.

4.2. Recursos de Desenvolvimento

Feita a escolha do processador procede – se a descrição de outros recursos necessários à implementação do sistema de comunicação.

4.2.1. Placa EVM

A placa EVM – *Evaluation Module* – para o DSP56L811 foi desenvolvida pela Motorola visando a facilidade de programação e a utilização de todo potencial de recursos disponíveis no processador; permitindo ao usuário – programador, um desenvolvimento descomplicado e eficiente de aplicações para o DSP. A placa

EVM contém, além de outros dispositivos, 64K de memória RAM – destinada ao desenvolvimento de programas e um codec linear de 13 bits.

A comunicação com a placa EVM através do PC pode ser feita de duas formas. Uma maneira é com protocolo serial de interface RS – 232, através do processador MC68HC705C4 presente na EVM. A EVM possui um conector DB9 permitindo sua conexão com terminal COM1 ou COM2 do PC. A outra maneira é através da porta JTAG da EVM utilizando a porta paralela do PC.

Para a implementação analógica da comunicação, foram utilizados os projetos das placas ADDA I e ADDA II (veja Anexos), através dos quais foi possível a interface entre a EVM e os conversores de sinais.

4.2.2. Ambiente de Programação e Depuração

O ambiente de programação *CodeWarrior* da *MetroWerks* foi escolhido para o desenvolvimento das rotinas e aplicações necessárias à implementação. Trata – se de um ambiente integrado composto por um editor de gráfico, ambiente de compilação, depuração, simulação e de auxílio na produção de códigos em linguagem C e assembly da família 56800.

4.3. Seleção da Melhor Alternativa de Implementação

Dentre algumas variações possíveis a implementação dos sub-sistemas transmissor e receptor, os canais de comunicação de dados podem ser analógico ou digital. No caso de se utilizar a comunicação analógica faz - se necessário o uso de conversores analógico - digital externos a EVM (ou através do codec, presente na placa, mas que não será utilizado). No caso da comunicação digital pode - se escolher entre os protocolos de comunicação seriais pré - existentes no

DSP, sendo eles: Interface Serial Síncrona - SSI, Interface Serial de Periféricos – SPI, Interface GPIO com as portas B/C ou emular a serial RS-232, ver manual do usuário do DSP56811.

Quanto aos conversores sua resolução dos conversores pode ser 8, 12 ou 16 bits de representação de dados. Entre as ligações possíveis dos conversores estas podem ser paralelas, podendo - se fazer uso da porta B GPIO do DSP ou através da memória mapeada na porta A, ou seriais, com o protocolo SSI, por exemplo.

5. Análises

A modelagem matemática realizada no Estudo de Viabilidade será utilizada para as análises à seguir.

5.1. Análise de Sensibilidade

Um sistema pode ser estudado por um conjunto de variáveis, chamadas de parâmetros de projeto. Uma descrição do comportamento do sistema pode ser realizada com a análise destes parâmetros juntamente com as variáveis de entrada e saída. De acordo com [1], a análise de sensibilidade tem por finalidade de determinar quão sensível é o desempenho do sistema ao ajuste dos parâmetros.

Esta análise foi realizada considerando os parâmetros de análise como os mesmos que identificam o comportamento do Sistema de Lorenz como caótico, ou seja, os parâmetros σ, **r** e **b**. Estes valores, são:

$$\begin{cases} \sigma = 16 \\ r = 45{,}6 \\ b = 4 \end{cases}.$$

O sistema de Lorenz foi simulado com outros valores. Para certos valores a resposta de sincronização de sinais foi mais lento do que o observado com simulações realizadas com os valores apresentados acima. Para outra gama de valores, o sistema de Lorenz não apresentava regime caótico, o que é indesejável visto a qualidade de "randomissidade produzida" não estar presente no sinal.

Logo, o sistema de Lorenz é sensível com relação aos valores dos parâmetros do sistema; apresentando dinâmicas de diferentes classificações, conforme discutido em [4].

5.2. Análise de Compatibilidade

A função desta análise é definir faixas de valores para os parâmetros de modo a assegurar compatibilidade funcional e compatibilidade dimensional.

Dividindo - se o sistema de comunicação em um sub - sistema de transmissão e outro de recepção verifica - se que os parâmetros em questão não são os mesmos da última análise. Aqueles devem ser iguais para ambos os sub - sistemas para garantir a sincronização e correta decodificação da mensagem. Os parâmetros nesta análise são os que permitem a conexão elétrica entre os circuitos dos sub - sistemas transmissor e receptor. Neste caso a compatibilidade de tensão e corrente elétrica entre os sub - sistemas devem ser atendidas.

Visto, os dois sub - sistemas, possuírem o mesmo processador, DSP, os requisitos de conexão entre ambos resumem - se na compatibilidade dos conversores analógico - digital de sinais. Este fato deve ser analisado quando da escolha do modelo dos conversores.

5.3. Análise de Estabilidade

O objetivo desta análise consiste na determinação das faixas de valores dos parâmetros para assegurar estabilidade intrínseca e extrínseca. Os parâmetros nesta análise são considerados os mesmos parâmetros matemáticos da análise de sensibilidade.

Serão considerados os valores para os parâmetros mostrados acima, como foi discutido na análise de Sensibilidade, para os quais o sistema de Lorenz apresenta resposta satisfatória.

Outro parâmetro, que a princípio não seria tão evidente, é o passo de integração. Para certos valores deste a sincronização é deficiente a tal ponto que a mensagem não é recuperada adequadamente. Simulações numéricas mostram que valores maiores que 0.008 de passo não são desejáveis.

6. Modelagem

Nas simulações realizadas no Estudo de Viabilidade, empregou - se o método de Runge - Kutta de 4ª ordem para resolver numericamente as equações diferenciais ordinárias do sistema de Lorenz. Apesar da validade dos resultados segundo esta modelagem, prevê - se um grande esforço computacional quando da implementação em sistema dedicado com o DSP. Tendo em vista esta previsão procurou - se uma modelagem menos elaborada e que apresentasse resultados tão eficazes quanto os apresentados com o uso do Método de Runge - Kutta.

Depois de sucessivas tentativas e avaliações a modelagem segundo equações de diferenças se mostrou adequada ao desejado. Os resultados obtidos foram analisados conforme a eficiência da recuperação da mensagem, velocidade de recuperação e discrepância em relação à mensagem original. Esta análise se reflete como conseqüência da sincronização entre os sub - sistemas transmissor e receptor. Através de simulações realizadas no MatLab verificou - se velocidade igual da recuperação de mensagem, se comparada com o modelagem original, mas com erros entre a mensagem original e decodificada de aproximadamente 2%. O algoritmo utilizado na simulação foi implementado segundo as relações, em tempo discreto, abaixo:

$$x[n+1] = x[n] + h*[\sigma*(y[n]-x[n])]$$
$$y[n+1] = y[n] + h*[r*x[n]-y[n]-y[n]*z[n]]$$
$$z[n+1] = z[n] + h*[y[n]*x[n]-b*z[n]]$$

O DSP56811 é um processador que se utiliza da representação de dados em ponto fixo. Isto quer dizer que o ponto decimal tem a sua posição fixa em relação aos bits da *palavra* de 16 bits. Além disso, caso seja necessário trabalhar com números com representação em ponto flutuante, ponto decimal com posição variável – possuindo o número parte inteira e fracionária, será indispensável a utilização das rotinas presentes na biblioteca *MSL C 56800.lib* e *FP56800.lib*, ou rotinas de manipulação de dados personalizadas.

Para a representação de dados em formatos ponto fixo o DSP utilizado faz uso de aritmética com *inteiros* e *fracionários*. Para maiores detalhes ver referência [8] e bibliografia recomendada. O formato fracionário será preterido sob recomendação do manual da família DSP56800.

Feita esta consideração, resume – se no próximo item conceitos relativos à programação em ponto fixo.

6.1. Aritmética de ponto fixo

Numa programação com dados e variáveis em ponto fixo algumas regras, diferentes de representações em ponto flutuante, devem ser cuidadosamente atendidas tanto para sua aplicação correta como para otimização do algoritmo.

A representação de um número em ponto fixo, ponto decimal não varia de posição com relação aos bits, pode ser entendida como a de um número inteiro escalonado, ou dividido, por uma potência de dois. Seja uma variável inteira não escalonada P e uma variável em ponto – fixo escalonada p segundo um valor de b bits, pode – se dizer, então, que

$$p = \frac{P}{2^b}.$$

Considerando um número de representação binária com *N* bits e um com representação em complemento de dois (*signed*), *A*, pode – se apresentar a notação

$$A(a,b),$$

onde $N = a + b + 1$ e sendo *b* o número de bits à direita do ponto decimal e conforme a potência da relação acima. Por exemplo, na representação de um número com 8 bits de *palavra* como $A(5,2)$, a localização do ponto decimal é mostrada conforme o número binário abaixo:

$$b_5 b_4 b_3 b_2 b_1 b_0 . b_{-1} b_{-2}.$$

Nesta representação o primeiro bit b_5 é reservado ao sinal do número.

Alguns conceitos sobre matemática de precisão finita devem ser levados em conta num projeto do sistema de processamento digital. Estes estão resumidos abaixo.

- Precisão: definido como sendo o número *N* de bits da *palavra*;
- Resolução: é o menor valor possível de ser representado pelo processador, ou seja 2^{-N};
- Extensão: É a diferença dos números limites, mínimo e máximo, do intervalo de valores possível. Traduzindo em notação matemática, o atributo *extensão* de um formato de representação com intervalo $-2^a \le p \le 2^a - 2^{-b}$ fica $2 * 2^a - 2^b$;

➢ Extensão Dinâmica: é a quantidade número diferentes possíveis de serem representados com *N* bits. Em outras palavras, 2^N ou $2*\frac{2^a}{2^{-b}}$.

Para o processador escolhido e descrito nos itens anteriores, o DSP possui palavras de 16 bits da dados. Pode – se considerar dois tipos de números em pauta. O primeiro consiste nos números reais correspondentes aos simulados e sendo representados através de variáveis do tipo *float* (ponto flutuante); o outro grupo é relativo aos números fracionários em ponto – fixo. A tabela apresentada abaixo resume os atributos acima, com relação ao DSP56811, e levando em conta os dois tipos de números de interesse.

Tabela 4: Comparação enter as duas Representações de dados de interesse.

ATRIBUTO	**Representação de Números Fracionários (ponto fixo)**	**Representação de Números Reais (ponto flutuante)**
Precisão	16 bits	16 bits
Resolução	Para $A(0;15)$, $2^{-15}=0.0000305$	Para $A(4;11)$, $2^{-11}=0.0004882$
Extensão	$-2^0 \le p \le 2^0-2^{-15}$ $(-1 \le p \le 0.9999695)$	$-2^4 \le p \le 2^4-2^{-11}$ $(-16 \le p \le 15{,}9995117)$
Extensão Dinâmica	$2^{16}=65536$	$2^{16}=65536$

Para implementação do algoritmo de resolução das equações diferenciais, somente as operações aritméticas de adição / subtração e multiplicação foram necessárias.

Numa operação de adição, os dois números precisam estar escalonados de igual modo para que a operação forneça resultados corretos. Ou seja, para a adição, ou subtração, de $X(c;d)$ com $Y(e;f)$, é necessário que $X=Y$ (os dois

números são do mesmo tipo – com ou sem sinal – no caso, complemento de dois), e $c=e, d=f$. O resultado da soma de dois números $X(c;d)$ é um número do tipo $X(c+1;d)$.

Na operação de multiplicação de dois números com *N* bits, além dos multiplicandos serem do mesmo tipo, deve – se atentar para o fato do resultado ser um número *2N*. Para a multiplicação $A(a_1;b_1)*A(a_2;b_2)$ tem – se um resultado do tipo $A(a_1+a_2+1;b_1+b_2)$. Como o DSP utilizado possui 16 bits para representação de dados, este necessita de um acumulador de, no mínimo, 32 bits para efetuar as operações corretamente e sem perda de resolução. O DSP56811 possui acumulador de 36 bits (mais quatro bits de extensão). O resultado da multiplicação precisa ter seu ponto decimal deslocado pois os dois números multiplicados foram escalonados de *b* e o resultado desta, fica escalonado de *2b*. Portanto, é necessário deslocar o ponto decimal de *b* bits para a direita. Este valor, intermediário, é arredondado para representação em 16 bits e, então, colocado no registrador ou numa posição de memória – ambos de 16 bits – finalizando a operação.

Durante o desenvolvimento do final do algoritmo, especial atenção foi dedicada com a finalidade de representar todos os valores finais como intermediários, em cada iteração, dentro da extensão disponível, evitando, assim, *overflow*. Para tanto, o algoritmo passou por refinamentos sucessivos, num total de 5, visando este objetivo. O quadro abaixo mostra, de maneira simplificada, o estudo envolvido, apresentando somente quatro etapas.

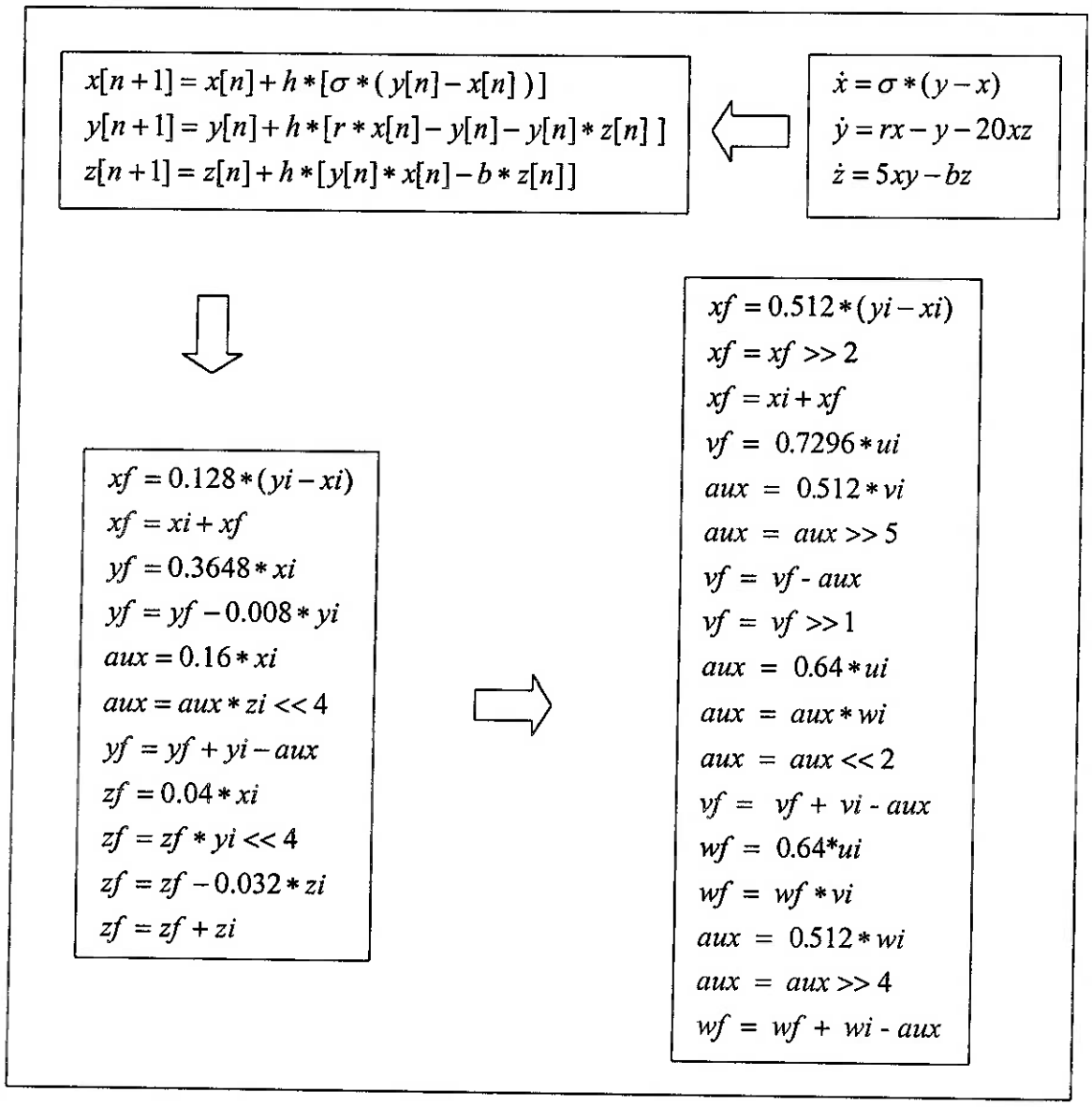


Figura 15: Refinamentos Sucessivos do algoritmo.

O próximo passo consiste na implementação do Sistema de transmissão e recepção Caótico. Essa implementação é composta de duas etapas; a primeira, através de uma comunicação dados digital, e a segunda, com uma comunicação analógica.

7. Implementação Digital

A programação da comunicação digital foi implementada com o uso da porta B, programada como entrada e saída de propósito de geral (GPIO).

7.1. Esquema de Montagem - Hardware

Um esquema simplificado de montagem do circuito eletrônico implementado para a transmissão paralela digital é mostrado abaixo.

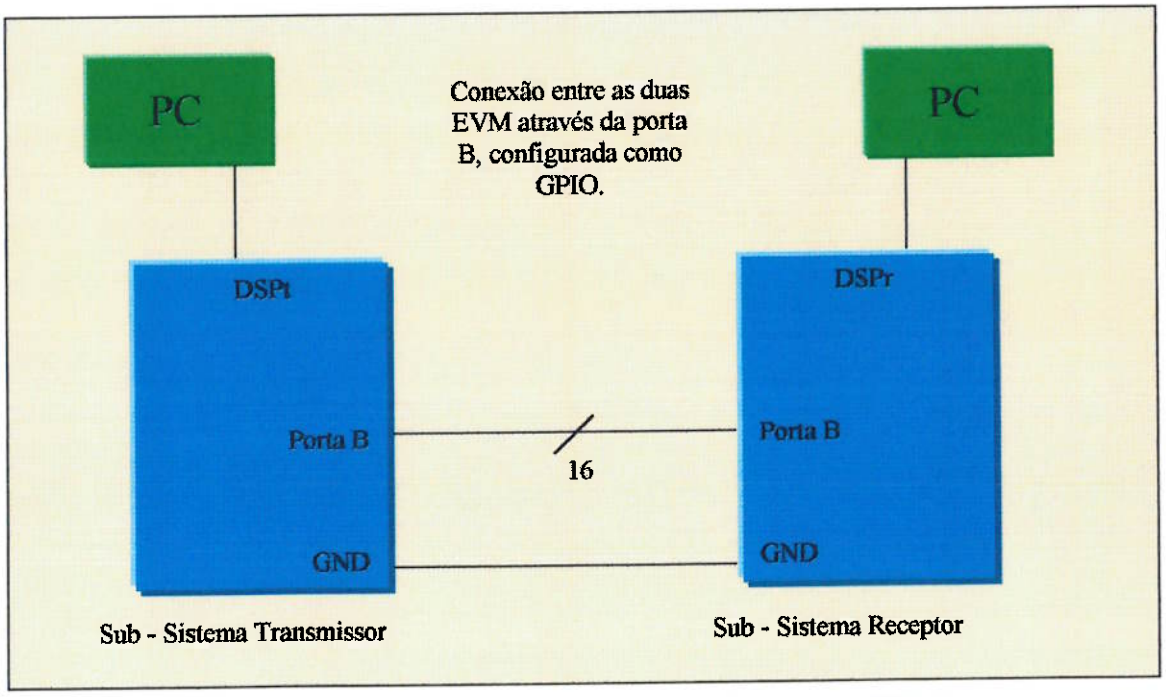


Figura 16: Implementação da Comunicação Digital entre as EVM's.

7.2. Programação - Software

Todos os 16 pinos (16 bits) da porta B da EVM transmissora foram programadas como saída enquanto que os pinos da EVM receptora, como entrada. Os diagramas de Nassi – Schneiderman para a transmissão e recepção são mostrados à seguir:

Inicialização Transmissão porta B
Inicialização RTI
Enquanto não atingir tempo de simulação
Determinação da mensagem
Resposta do Sistema Transmissor
Tratamento do sinal antes da comunicação (para transmissão Analógica)
Transmissão sinal composto (Digital / Analógica – Porta B)
Comandos finais de plotagem (opcional)

Figura 17: Diagramas de Nassi – Schneiderman para a transmissão.

Inicialização Recepção porta B
Inicialização RTI
Enquanto não atingir tempo de simulação
Recepção sinal composto (Digital / Analógica – Porta B)
Tratamento do sinal (para transmissão Analógica)
Resposta do Sistema Receptor
Determinação da mensagem
Comandos finais de plotagem (opcional)

Figura 18: Diagramas de Nassi – Schneiderman para a recepção.

Notar que ambos possuem estrutura análoga para comunicação digital e analógica.

7.3. Testes e Resultados

A abordagem utilizada nos testes foi a de Oppenheim modificado.

O sistema de comunicação foi testado em blocos. Um algoritmo foi desenvolvido com base na programação do módulo de Interrupção em Tempo Real do DSP (RTI - *Real Time Interrupt*). Deste modo, a cada intervalo de tempo programado o sub - sistema de transmissão calcularia o valor atual da componente "onda portadora" a ser enviada ao sub - sistema receptor. O passo seguinte foi determinar o tempo necessário para execução do *loop* do sistema de Lorenz.

Tomando - se como base as especificações funcionais do Estudo de Viabilidade, estima - se que para uma representação de sinais com espectro em freqüência de até 3kHz, considera - se como parâmetro de projeto uma referência com taxa de amostragem de 6kHz ou 0,167ms.

Nos testes preliminares o tipo básico das variáveis utilizadas nestes testes foi o *float*, de 32 bits. Com o auxílio de um bit da porta C, pôde - se determinar o tempo necessário para os cálculos de um *loop* completo, medindo - se o intervalo de tempo em que o sinal proveniente permanecesse em nível lógico "1". O resultado desta medida foi de 840μs (sem rotinas de transmissão). Este valor é 5 vezes maior do que o requisito funcional para o projeto, que especifica uma amostragem de dados de 167μs (ou, 6kHz). O algoritmo foi, então, escrito

utilizando variáveis fracionárias de 16 bits ponto fixo, visando diminuir o período de *loop*.

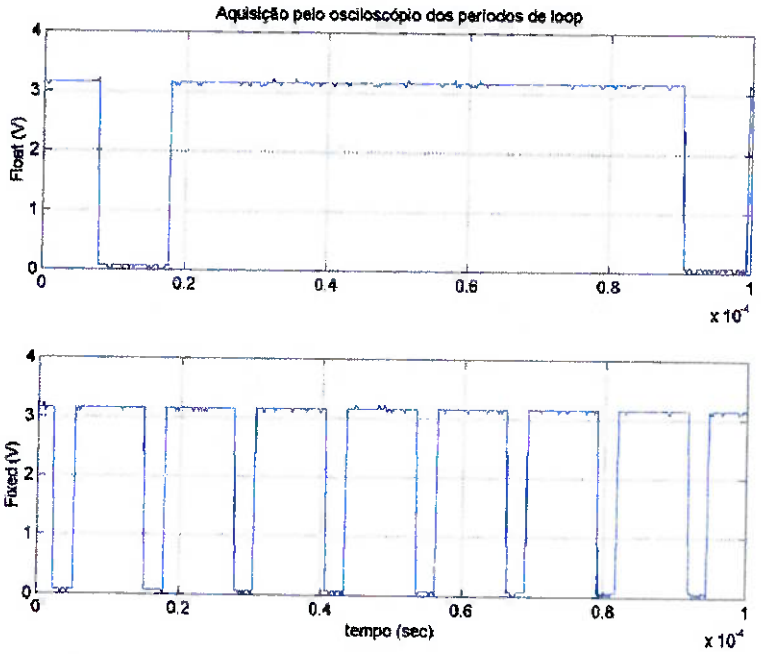


Figura 19: Tempos de *loop* para algoritmos em ponto flutuante e ponto fixo.

Na figura acima, o período do algoritmo programado em ponto fixo é de 128 μs (com rotinas de transmissão). Os resultados de simulações com o MetroWerks e uma EVM do algoritmo de comunicação digital se mostraram satisfatórias e são mostradas abaixo.

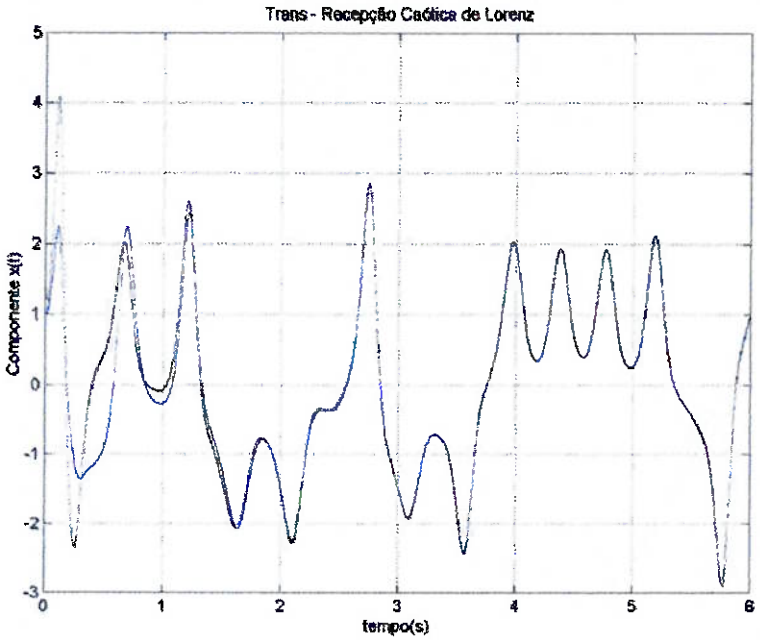


Figura 20: Sinais sincronizados - Simulação DSP56811.

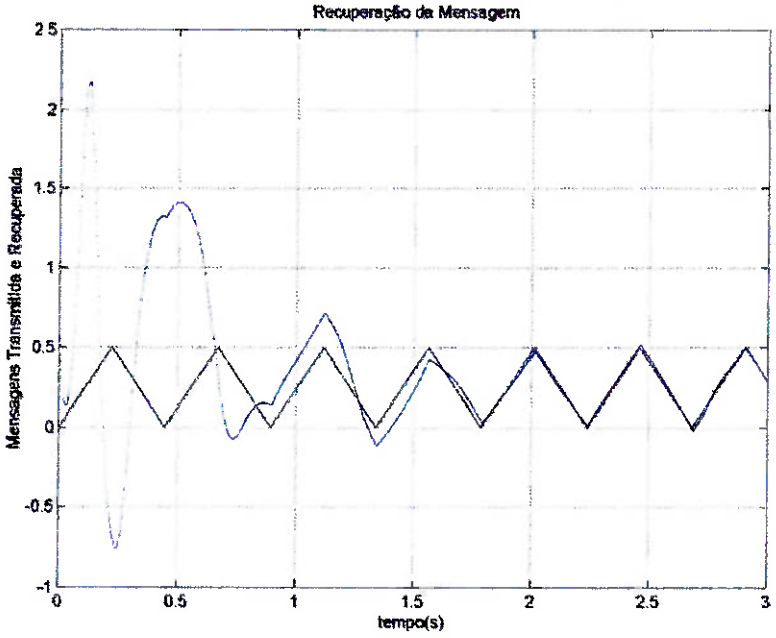


Figura 21: Mensagem decodificada – com mensagem original.

Os resultados dos testes realizados são mostrados nas próximas figuras. Os dados transmitidos pelo DSP na ADDA I, sincronizados pelo DSP na ADDA II

assim como a decodificação da mensagem criptografada podem ser analisadas através de comparações com os resultados numéricos do item 3 e de acordo com as simulações das figuras anteriores. A aquisição dos sinais foi realizada com o conversor DA da placa ADDA II; enviando, ao osciloscópio, um sinal correspondente aos oito bits mais significativos das variáveis de integração.

A mensagem utilizada para comunicação foi uma codificação dos caracteres da palavra "*hello*" em ASCII.

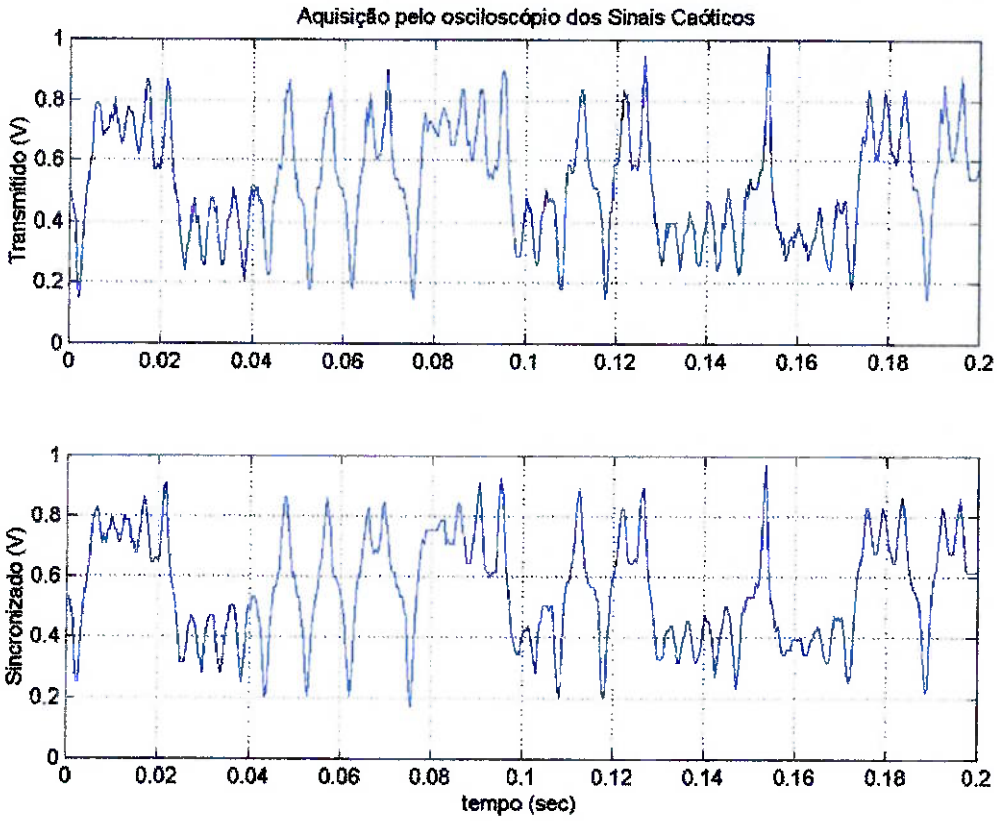


Figura 22: Sinais Sincronizados - Comunicação Digital.

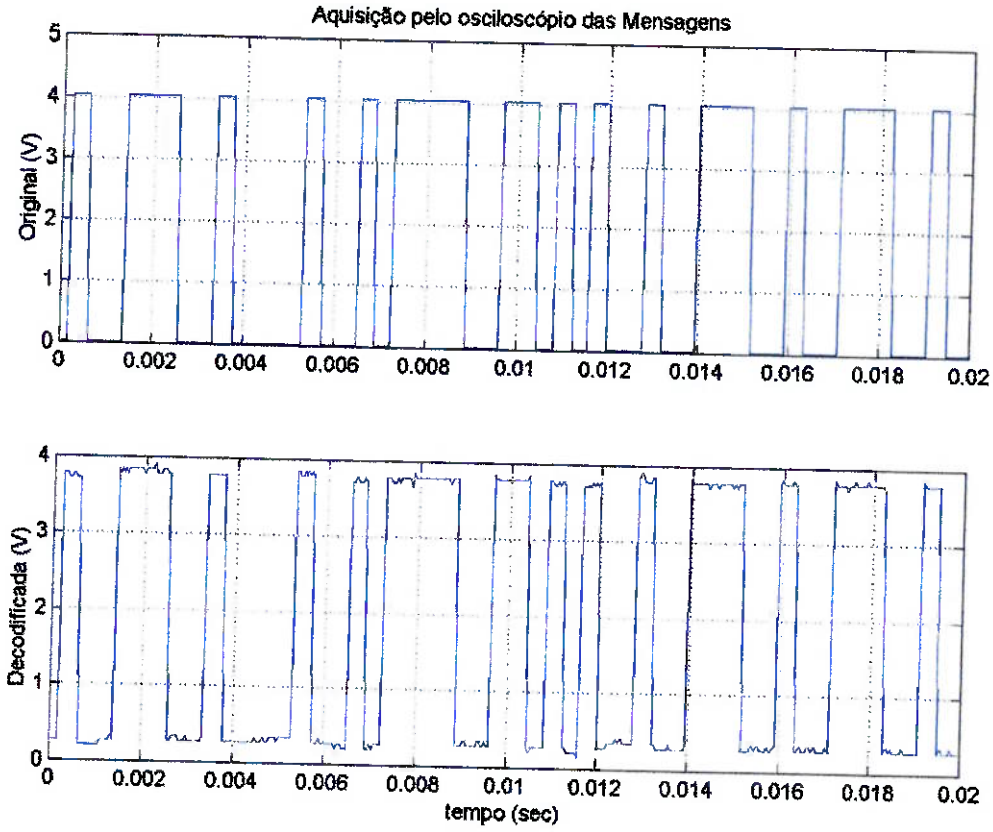


Figura 23: Recuperação da Mensagem: "hello" em ASCII.

8. Implementação Analógica

Após a implementação da comunicação digital, o próximo passo é efetivar comunicação analógica.

8.1. Esquema de Montagem – Hardware

Além das placas EVM, o uso de conversores D/A e A/D é necessário à comunicação de sinais analógicos entre os dois sub - sistemas. O esquema de montagem é mostrado abaixo:

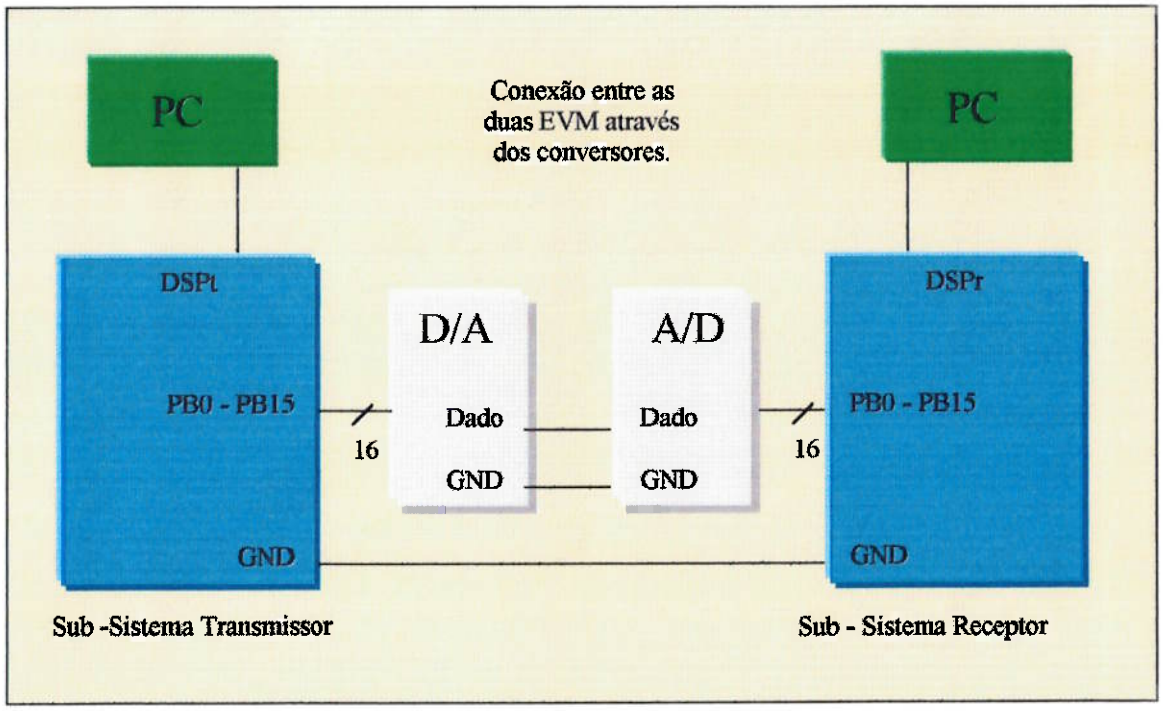


Figura 24: Implementação Analógica de Comunicação, EVM - DSP-PC.

Na figura acima não foram representados os CI's responsáveis pela conversão de tensões que garantem compatibilidade elétrica entre o DSP (3,3V) e os conversores A/D e D/A (5,0V).

Os modelos dos conversores D/A e A/D presentes nas placas AADA I e II e utilizados para implementação estão descritos à seguir.

D/A

Modelo: TLC7225C, conversor em paralelo;

Fabricante: Texas Instruments;

Características: Quatro Conversores D/A de 8 bits com tensões de referência independentes, TTL/CMOS compatível.

A/D

Modelo: AD7828, conversor em paralelo;

Fabricante: Analog Devices;

Característica: LC^2MOS – de alta velocidade Oito entradas (canais) analógicas.

O AD7828 trabalha com a conversão de sinais analógicos em digitais com representação em 8 bits. Não necessita de sinal de *clock* para operação.

O conversor AD consegue amostragens a uma taxa máxima de 50kHz. Uma taxa de amostragem de, no mínimo, 6kHz é, portanto, atendida. Este pode assimilar sinais analógicos de entrada com taxas de até 157mV/μs (*slew rate*), ou seja, um sinal analógico com freqüência de até 10kHz pode ser amostrado sem a necessidade de circuitos externos de retenção (*sample and hold*).

A operação deste conversor prevê dois modos de funcionamento, à saber Modos 0 e 1. O primeiro modo é usado somente para o microprocessadores que se fazem uso de estados de espera (*wait states*), em que o ciclo de instrução de leitura - READ – é estendida para se adequar ao uso de memórias lentas. Já o

Modo 1, é usado quando o microprocessador não é forçado em modo de espera. O modo 1 foi utilizado.

Os dois conversores são compatíveis com os níveis de tensão do padrão TTL de 5V, entretanto o DSP56L811 opera com 3,6V. Existe, portanto, a necessidade de um dispositivo que faça a interface de níveis lógicos de tensão entre o processador e os conversores. O dispositivo utilizado para este fim foi o QS3245 - chaveador de barramento de 8 bits (*8 bit bus switch*); realizando a conversão de tensão entre 5 e 3,3V.

8.2. Programação – Software

Antes da comunicação propriamente dita, procurou – se estudar a necessidade da comunicação, entre as duas placas, com o envio de dados de 16 bits do DSP ao conversor DA e do conversor AD ao DSP para uma representação de dados adequada à sincronização dos sinais. Caso uma comunicação com apenas 8 bits fosse permitida (possibilitando a sincronização e recuperação da mensagem), isto acarretaria numa diminuição do custo do protótipo. O algoritmo testado foi executado no simulador para o DSP56811 com as seguintes linhas entre a passagem da variável do sistema transmissor ao receptor:

```
sinal = sinal >> 8;                 // transforma em dado 8 bits
sinal = sinal << 8;                 // volta a 16 bits
sinal = sinal + 0.003902;           // soma 127
```

A primeira linha seria executada pelo sistema transmissor, que enviaria um sinal com apenas os 8 bits mais significativos (a informação do byte menos significativo é perdida). A variável "sinal", na segunda linha, é transformada novamente em 16 bits e somada com 127 ou 0x7F; permitindo reduzir o erro em,

no máximo, duas vezes. O resultado deste teste pode ser observado abaixo; sugerindo sua aplicação na comunicação de fato.

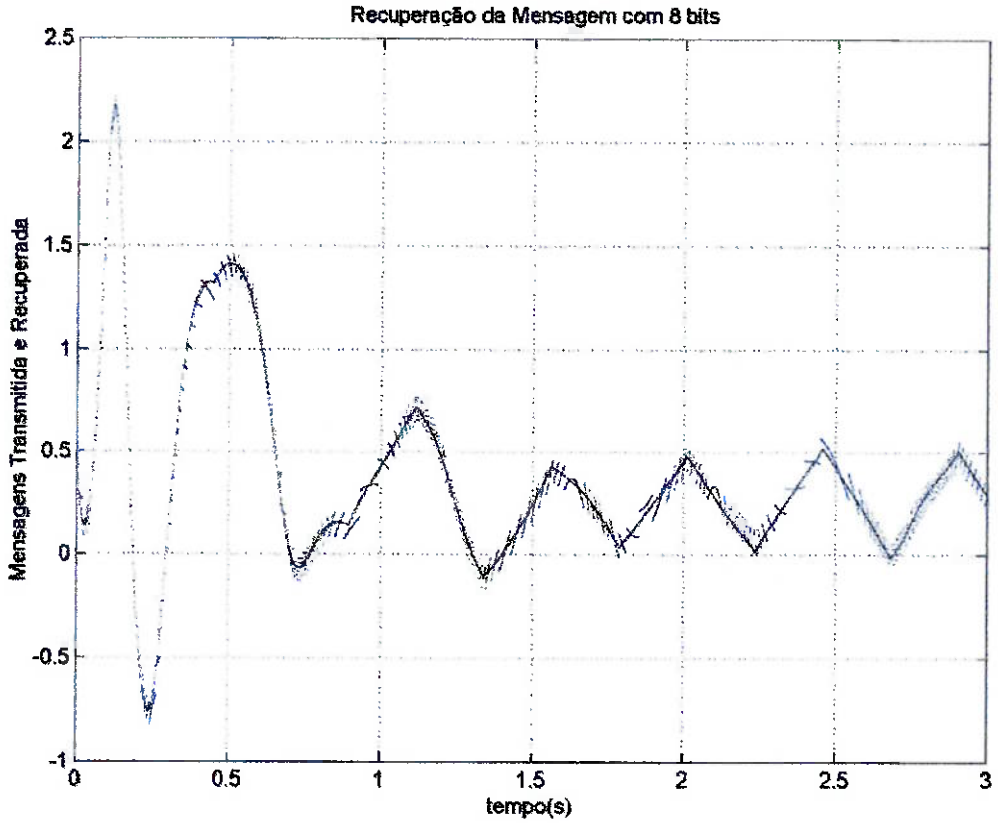


Figura 25: Recuperação da Mensagem com Simulação de transmissão com 8 bits.

Nota – se da figura acima que houve a introdução de uma componente de freqüência elevada na mensagem decodificada. Isto poderia ser contornado com uma filtragem (passa – baixas) da mensagem e que poderia ser implementada digitalmente.

8.3. Testes e Resultados

Os resultados obtidos, com condições iniciais para o transmissor $(x_i, y_i, z_i) = (1.5; -0.5; -1.1)$ e para o receptor $(x_i, y_i, z_i) = (1;1;1)$ e com passo h = são mostrados abaixo.

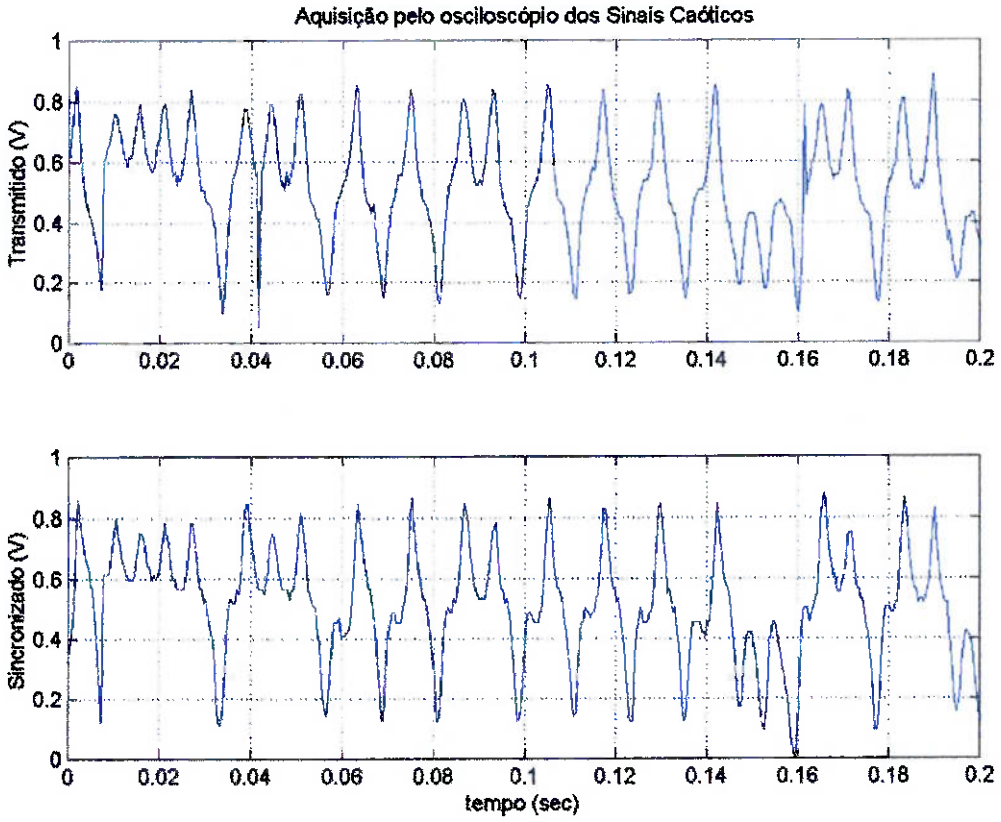


Figura 26: Sinais sincronizados - Comunicação Analógica.

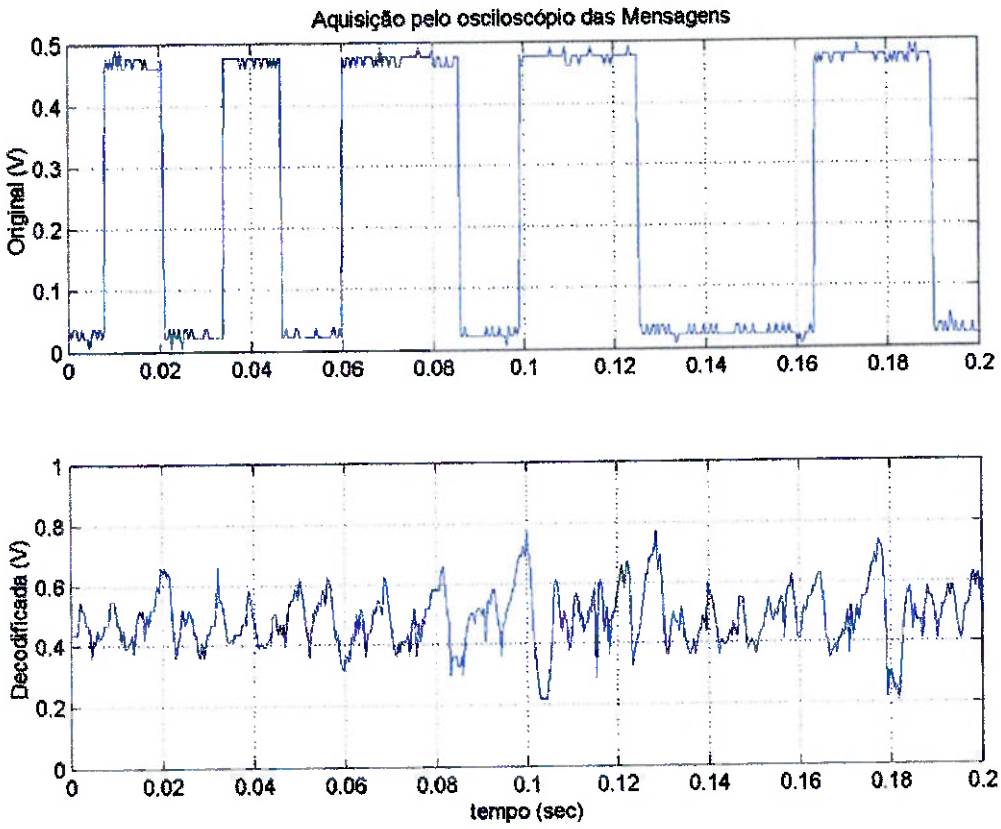


Figura 27: Mensagem decodificada – com mensagem original.

Como pode ser observado pela Figura 26 os dois sinais caóticos, do transmissor e receptor, tendem à sincronização, mas que não ocorre de fato. Consequentemente a recuperação da mensagem original fica comprometida, Figura 27. Visto os algoritmos de codificação e decodificação serem os mesmos dos utilizados nas simulações e as funções de transmissão e recepção dos conversores estarem funcionando corretamente, procurou – se investigar a razão para a não verificação do comportamento esperado para o sistema. O sinal analógico entre os conversores não apresentou sensibilidade à interferência eletromagnética mas, testes realizados com a recepção do sinal pelo ADC e passado diretamente ao DAC, na placa receptora, indicavam existir diferenças

entre os sinais transmitido e recepcionado. Além disso a placa ADDA II possui um circuito que fornece tensões de referência estáveis para os seus conversores; que não existe na placa ADDA I. Logo, um valor digital convertido para analógico na placa transmissora pode não ser convertido de maneira análoga nos conversores da placa receptora, gerando erros independentes de programação. Estes erros são amplificados 256 vezes quando recebidos pois o sinal transmitido corresponde ao byte mais significativo do sinal. Consequentemente, erros de 10%, numa extensão de valores de –3 a 3 (ver Figura 20), entre os sinais transmitido e sincronizado, foram observados. Pelo fato do sistema de Lorenz ser sensível tanto às condições iniciais de integração como à mensagem de transmissão o erro presente na comunicação impossibilitou uma decodificação satisfatória. Com base nos tempos de *loop* apresentados acima, pode – se realizar uma estimativa da resposta em freqüência do sinal caótico, sinal composto transmitido.

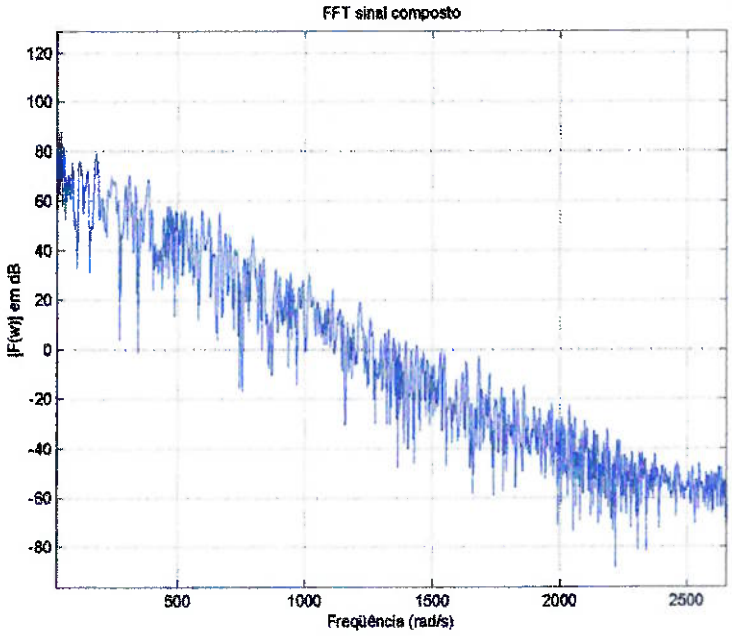


Figura 28: Estimativa da FFT do sinal composto.

9. Considerações Finais

Nota - se pelos resultados obtidos através de simulações numéricas no item 3 que a criptografia caótica de fato codifica o sinal ou mensagem de interesse. Observou - se, também que com valores de passo abaixo de 0.001 a recuperação da mensagem é satisfatória, não apresentando diferenças notáveis nos gráficos obtidos. Comentou - se, também, a respeito do transitório, que exige um número elevado de iterações para ser superado. Este fator pode representar um obstáculo para aplicações comerciais e um estudo para minimiza - lo se faz necessário.

Observou - se que uma transmissão, quando da representação de dados como variáveis de 32bits (tipo *float*), possui uma freqüência de amostragem máxima de 1100Hz. Este valor corresponde a quase 20% da freqüência necessária. Em vista disto, estudou - se a viabilidade de um algoritmo que se utilize de variáveis do tipo inteiros (*short*); com resultados satisfatórios. Isto reduziu consideravelmente o tempo de *loop* considerado pois as operações algébricas seriam com variáveis de 16bits, eliminando a necessidade de duas operações de busca (*fetchs*) na memória de dados.

Outro aspecto a comentar é relativo à baixa resposta em freqüência do sistema para simulações numéricas. Para sua aplicação em casos práticos de implementação, notou – se que a resposta em freqüência se aproxima daquela obtida por Oppenheim – Kuomo. Isto acontece por causa do passo utilizado para a integração do sistema de equações ser de 0.001, mesmo com o período em que é solicitada interrupção ser de ~100μs. Isto acarreta numa rápida resolução das equações diferenciais no tempo e consequentemente no aumento da resposta em

freqüência. Este processamento, entretanto, não é rápido o suficiente para se equiparar com os resultados da implementação analógica obtidas por Kuomo – Oppenheim. Portanto, pode – se resumir que a escolha do processador, para esta implementação, se mostrou adequada quanto aos requisitos de funcionamento – sincronização e decodificação de sinais e taxa de amostragem de 6kHz – mas que, sob o aspecto da resposta em freqüência do sistema caótico, não se mostrou adequada. Por causa disto, a criptografia de sinais com componentes de freqüência máxima de 3kHz, por exemplo, e segundo a implementação adotada neste estudo, não são possíveis. Isto é devido a baixa resposta de 400Hz (a 60 dB negativos) do sistema caótico, o que possibilitaria a identificação da componente de freqüência da mensagem, com banda maiores que 400Hz, numa análise do espectro de potência.

Outro aspecto a comentar é com relação à implementação digital da qual se obteve resultados satisfatórios e condizentes com as simulações prévias quando analisadas em relação à sincronização dos sinais e decodificação das mensagens.

A comunicação Analógica, por outro lado, não apresentou o mesmo resultado em virtude das considerações acima relatadas.

Quando do Projeto Executivo, para viabilização comercial deste projeto, algumas considerações devem ser levadas em conta. Dependendo da aplicação da criptografia, como mensagens de voz, por exemplo, o DSP utilizado deve possuir maior velocidade de processamento, de forma a garantir uma banda do sinal caótico acima de 3kHz. O algoritmo numérico de resolução das equações de Lorenz deve ser capaz de atender requisitos, em geral, conflitantes, como baixa exigência de processamento e erros de truncamento pequenos. A linguagem

adotada na programação poderia ser implementada completamente em assembly, ajudando atender os requisitos referentes ao tempo de processamento. Os conversores poderiam possuir maior precisão que os 8 bits utilizados, eliminando operações de deslocamento do ponto fracionário. Um maior cuidado deve ser tomado com relação à tensão de referência dos conversores; causa maior da não sincronização de sinais quando da implementação analógica.

A criptografia aqui apresentada é senão uma de várias aproximações possíveis com o sistema de Lorenz pois existem outras configurações que podem ser implementadas, como a proposta por Pecora – Carroll, além de outros sistemas caóticos. Atualmente as aplicações de Sistemas Caóticos para a comunicação criptografada estão em estudo. Mesmo que um dia sua viabilidade comercial seja comprovada é importante que seja comparada com a viabilidade tecnológica e econômica dos métodos de criptografia já existentes.

Deve – se ressaltar, todavia, que o estudo realizado e aqui documentado possui importância acadêmica e que o maior objetivo atingido foi o aprendizado do estudante.

10. Anexos

As fotos da bancada de instrumentação bem como das placas de ADDA I e ADDA II utilizadas para implementação são mostradas abaixo.

Figura 29: Bancada utilizada no desenvolvimento do projeto.

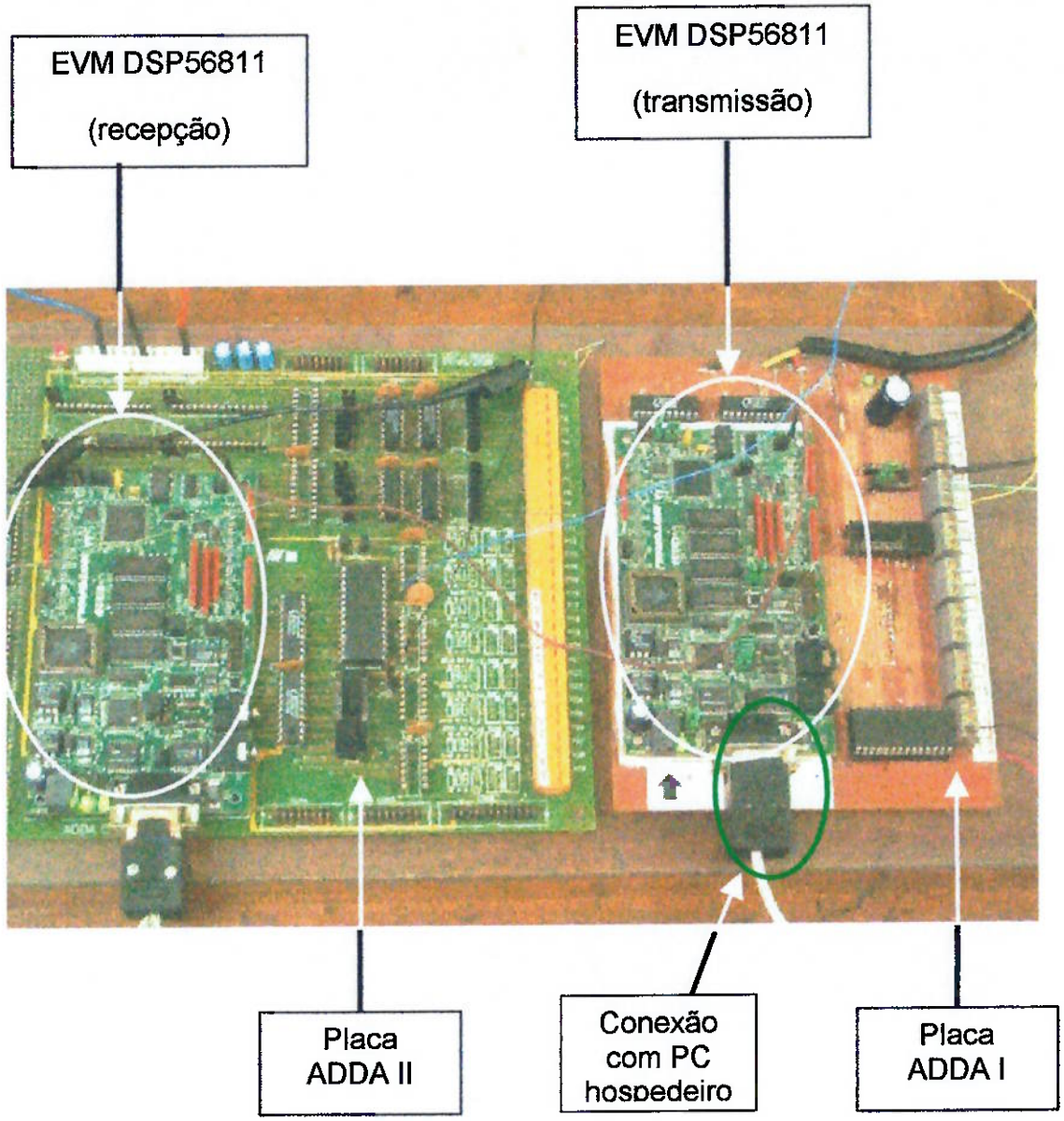


Figura 30: Placas ADDA I e II, com as respectivas placas EVM.

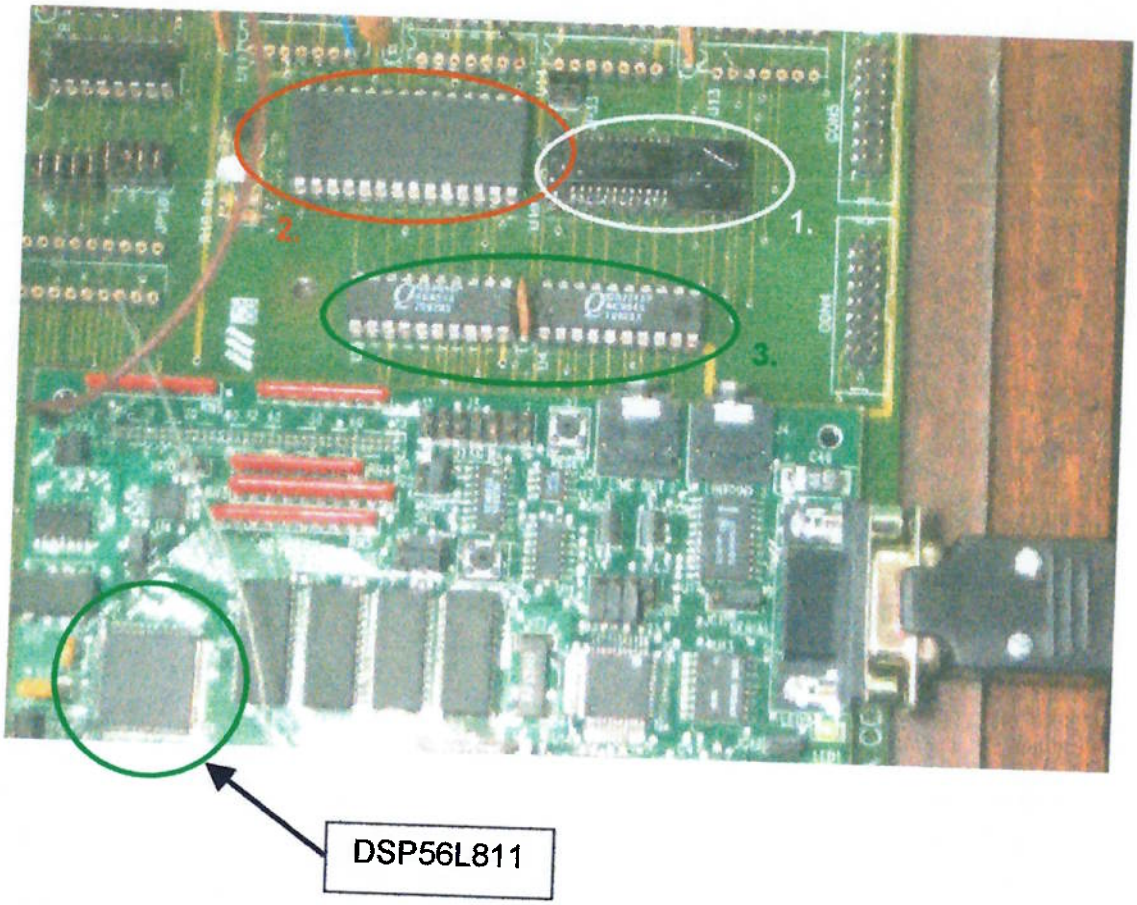


Figura 31: Principais componentes da placa ADDA II: 1 - Conversor Digital - Analógico de 8bits (DAC); 2- Conversor Analógico - Digitalde 8bits (ADC); 3- Lógica de Apoio (Glue logic).

Referências Bibliográficas

[1] – Alves, Marcelo A. L., Massarani, M., Kaminski, P. C., Ramos Jr., Roberto, Salvagni, Ronaldo B.; Apostila de Aula **PMC – 475 Metodologia de Projeto**, São Paulo, 1999;

[2] - Cuomo, Kevin M.; Oppenheim, Alan V, **Synchronization of Lorenz – Based Chaotic Circuits, with Applications to Communications**, *IEEE Transactions on Circuits and Systems*, outubro – 1993;

(ou Artigo: **Chaotic signals and systems for communications**)

http://www.causalproductions.com/TEMP/INDEX/IC93S304.HTM#p3137;

[3] – Ali H. Nayfeh, Balakumar Balachandran, **Applied Nonlinear Dynamics *Analytical, Computational and Experimental Methods***; A Wiley – Interscience Publicatios , John Wiley & Sons, Inc.;

[4] – Holborn, Robert C., **Chaos and Nonlinear Dynamics *an introduction for scientists and engineers***; Oxford University Press, Inc., 1994;

[5] - Hasler, M. Home Page: **Synchronization of chaotic systems and transmission of information**,

http://www.worldscientific.com/journals/ijbc/84_5/hasler.html;

[6] - Kubin Gernot, Artigo: **What is a Chaotic Signal?**;

http://www.nt.tuwien.ac.at/dspgroup/gkubin.html

http://poseidon.csd.auth.gr/Workshop/papers/p_5_8.html

[7] – Gamieiro, Márcio F., Rodrigues, Hildebrando M., **Synchronization in Communication systems. Robustness with Respect to Parameter Variation.**, 49º Seminário de Análise, separatas 1999;

[8] - Yates, Randy; **Fixed-Point Arithmetic: An Introduction**

26 de Julho de 2000, Artigo, http://207.87.184.178/papers.htm

Bibliografia

1. Motorola Semiconductors: http://www.motorola.com;
2. Berkeley Design Technology Inc.: http://www.bdti.com;
3. Motorola Inc, **DSP56L811 User's Manual**, Motorola Semiconductor Products Sector, DSP Division, Austin, TX, 1996;
4. Motorola Inc, **DSP56800 Family Manual**, Motorola Semiconductor Products Sector, DSP Division, Austin, TX, 1996;
5. Metrowerks Inc., **CodeWarrior Targeting DSP56800**, Release 3.5, Metrowerks Codewarrior Documentation, Austin, TX, 1996.
6. Notas Teóricas de James Carr da Universidade Estadual da Florida da página na Internet:

   http://www.scri.fsu.edu/~jac/MAD3401/Backgrnd/ieee.html

Apêndice A – Projeção do Espaço - Estado

Observando - se a trajetória de um sistema caótico, nota - se, pela plotagem das projeções nos planos *x - y* e *x - z* a ocorrência de regiões com características particulares. Nestas regiões, a trajetória do sistema caótico fica limitada externamente a uma região como se estivessem sendo atraída por um ponto no espaço de estados denominando *atratores estranhos*. Pode – se determinar analiticamente as coordenadas do ponto bem como a sua natureza: se pontos de atração ou repulsão, ou pontos de sela.

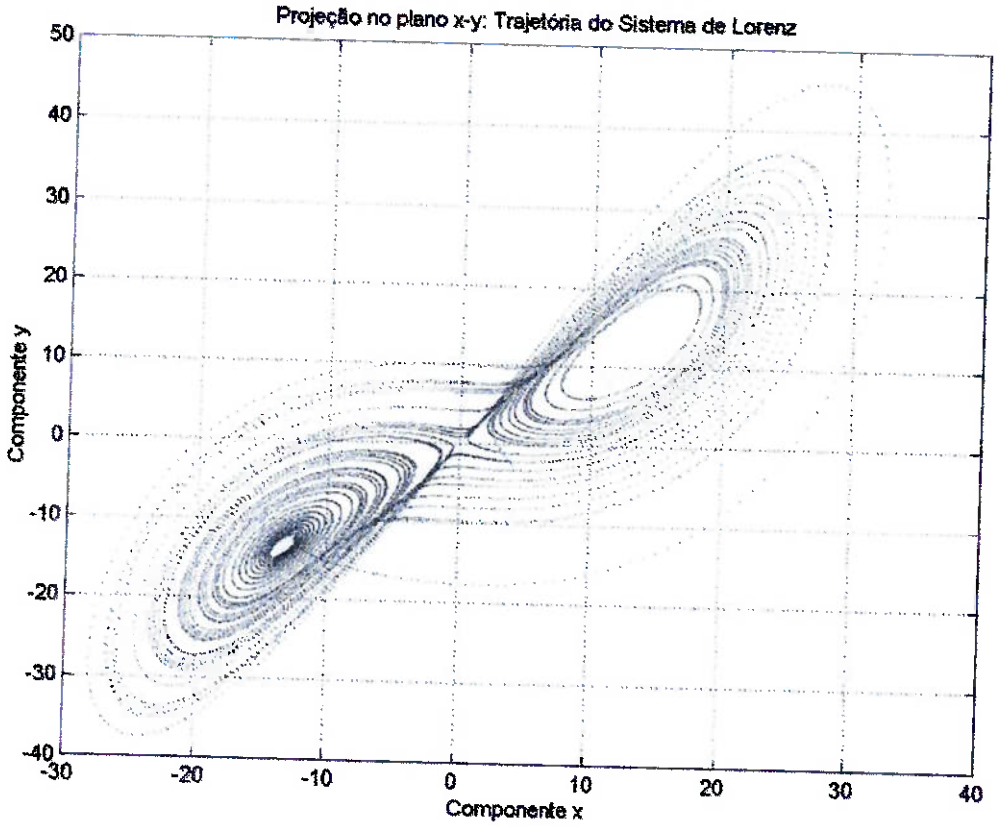


Figura 1: Projeção no plano x - y da trajetória do Sistema de Lorenz no Espaço - Estado.

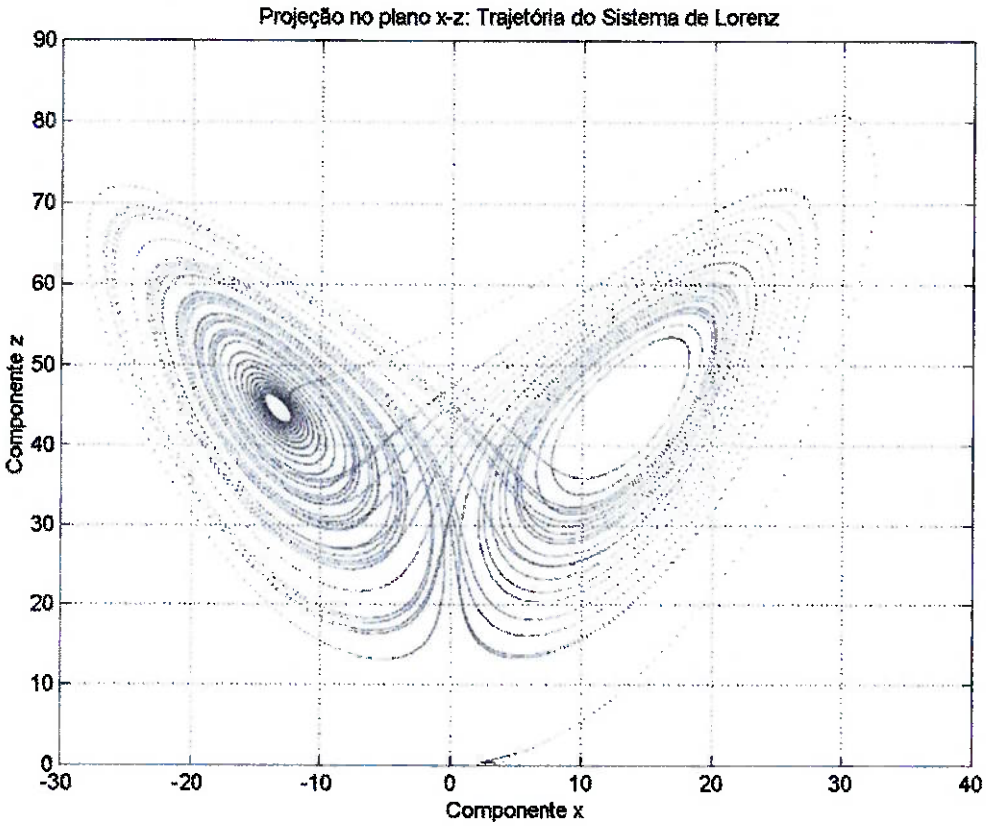


Figura 2: Projeção no plano x - z da trajetória do Sistema de Lorenz no Espaço - Estado.

As Figuras 1 e 2 acima mostram as projeções da trajetória do sistema de Lorenz nos planos x-y e y-z, respectivamente. Foram construídas com passo h = 0.002 e condições iniciais $(x_i, y_i, z_i) = (4.3, -3.0, 0.5)$.

Apêndice B – Definições de Soluções

Definições de sinais ou soluções em equilíbrio, periódicos e quasi – periódicos são feitas abaixo.

Soluções em Equilíbrio

Para sistema autônomos, que são independentes do tempo, como o abaixo:

$$\dot{x} = F(x;M),$$

onde *M* representa o conjunto de parâmetros, define - se os pontos fixos como sendo o ponto, ou os pontos, em que:

$$F(x;M) = 0$$

Pontos fixos são, também, chamados de soluções estacionárias, pontos críticos. Fisicamente, esses pontos correspondem a uma posição de equilíbrio do sistema.

Soluções Periódicas

Um solução $x = X(t)$ de sistema em tempo contínuo é periódico se $X(t+T) = X(t)$ e $X(t+\tau) \neq X(t)$ para $0 < \tau < T$, onde *T* é o período da solução.

Soluções Quasi – Periódicas

Uma solução é dita quasi – periódica se possui duas ou mais freqüências chamadas de incomensuráveis. Duas freqüências são incomensuráveis se a razão delas resulta em um número irracional. Para *m* freqüências de incomensuráveis pode – se escrever a equação:

$$n_1w_1 + n_2w_2 + \cdots + n_mw_m = 0$$

Esta equação é satisfeita somente se cada *ni* inteiro for nulo.

Logo, uma função quasi – periódica de k períodos possui a forma:

$x = x(w1t, w2t, \ldots, wkt)$.

Apêndice C – Simulações Numéricas

Código fonte da simulação no MatLab do sistema de transmissão – recepção.

```
% Parâmetros de entrada:passo (h).

function y=orig3(h)

figure

%Parâmetros e inicializações...
sig = 16;    %Estes valores para "sig","r" e "b" tornam o sistema de
lorenz
r = 45.6;    %caótico.
b = 4;
t=0;      %Variável da contagem do Tempo recebe zero (s).

ui = 10;
vi = 1.5;
wi = -6.5;

uri = 1;
vri = 1;
wri = 1;

%Método de Runge-Kutta 4a. ordem
while (t<3.0)            %Condição de parada.

mes = sin(500*t);

%Transmissor [x,y,z].
 k1 = sig*(vi - ui);
 q1 = -vi - (ui + mes)*wi + r*(ui + mes);
 u1 = -b*wi + (ui + mes)*vi;

 k2 = sig*((vi + q1*h/2) - (ui + k1*h/2));
 q2 = -(vi + q1*h/2) - (ui + k1*h/2 + mes)*(wi +u1*h/2) + r*(ui + k1*h/2
+ mes);
 u2 = -b*(wi + u1*h/2) + (ui + k1*h/2 + mes)*(vi + q1*h/2);

 k3 = sig*((vi + q2*h/2) - (ui + k2*h/2));
 q3 = -(vi + q2*h/2) - (ui + k2*h/2 + mes)*(wi +u2*h/2) + r*(ui + k2*h/2
+ mes);
 u3 = -b*(wi + u2*h/2) + (ui + k2*h/2 + mes)*(vi + q2*h/2);

 k4 = sig*((vi + q3*h) - (ui + k3*h));
 q4 = -(vi + q3*h) - (ui + k3*h + mes)*(wi +u3*h) + r*(ui + k3*h + mes);
 u4 = -b*(wi + u3*h) + (ui + k3*h + mes)*(vi + q3*h);

   uf = ui + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
   vf = vi + (h/6)*(q1 + 2*q2 + 2*q3 + q4);
```

```
    wf = wi + (h/6)*(u1 + 2*u2 + 2*u3 + u4);

    ui = uf;
    vi = vf;
    wi = wf;

sinal = uf + mes; %Sinal embutido a ser transmitido.
t = t + h;      %O tempo é incrementado de h. É mostrado a cada interação.

%Receptor [ur,vr,wr].
 a1 = sig*(vri - uri);
 b1 = -vri - sinal*wri + r*sinal;
 c1 = -b*wri + sinal*vri;

 a2 = sig*((vri + b1*h/2) - (uri + a1*h/2));
 b2 = -(vri + b1*h/2) - sinal*(wri +c1*h/2) + r*sinal;
 c2 = -b*(wri + c1*h/2) + sinal*(vri + b1*h/2);

 a3 = sig*((vri + b2*h/2) - (uri + a2*h/2));
 b3 = -(vri + b2*h/2) - sinal*(wri +c2*h/2) + r*sinal;
 c3 = -b*(wri + c2*h/2) + sinal*(vri + b2*h/2);

 a4 = sig*((vri + b3*h) - (uri + a3*h));
 b4 = -(vri + b3*h) - sinal*(wri +c3*h) + r*sinal;
 c4 = -b*(wri + c3*h) + sinal*(vri + b3*h);

 urf = uri + (h/6)*(a1 + 2*a2 + 2*a3 + a4);
 vrf = vri + (h/6)*(b1 + 2*b2 + 2*b3 + b4);
 wrf = wri + (h/6)*(c1 + 2*c2 + 2*c3 + c4);

 uri = urf;
 vri = vrf;
 wri = wrf;

%Comandos de "plotagem" a cada interação.

subplot(2,1,1)
plot(t,uf,'k')
plot(t,urf,'b-')
hold on

subplot(2,1,2)
plot(t,mes,'k:')
plot(t,sinal-uri,'b:')
hold on

end    %while

subplot(2,1,1)
grid on;
xlabel('tempo (s)')
ylabel('Componente x(t)')
title('Trans - Recepção Caótica de Lorenz')
axis([0 10 -30 35]);

subplot(2,1,2)
grid
```

```
xlabel('tempo (s)')
ylabel('Componente x(t)')
title('Mensagens Original e Recuperada')
axis([0 3 -2 2.5]);
```

Esquema proposto por Pecora - Caroll.

```
% Abordagem de Criptografia Caótica proposto por Pecora - Caroll.
% Parâmetros de entrada:passo (h).

function y=pecora(h)

%Parâmetros e inicializações...
sig = 16;    %Estes valores para "sig","r" e "b" tornam o sistema de
r = 45.6;    % lorenz caótico.
b = 4;
t=0;      %Variável da contagem do Tempo recebe zero (s).

ui = 10;
vi = 1.5;
wi = -6.5;

uri = 1;
vri = 1;
wri = 1;

figure

%Método de Runge-Kutta 4a. ordem
while (t<10.0)        %Condição de parada.

%Transmissor [x,y,z].
 k1 = sig*(vi - ui);
 q1 = -vi - (ui )*wi + r*(ui );
 u1 = -b*wi + (ui )*vi;

 k2 = sig*((vi + q1*h/2) - (ui + k1*h/2));
 q2 = -(vi + q1*h/2) - (ui + k1*h/2 )*(wi +u1*h/2) + r*(ui + k1*h/2 );
 u2 = -b*(wi + u1*h/2) + (ui + k1*h/2)*(vi + q1*h/2);

 k3 = sig*((vi + q2*h/2) - (ui + k2*h/2));
 q3 = -(vi + q2*h/2) - (ui + k2*h/2 )*(wi +u2*h/2) + r*(ui + k2*h/2 );
 u3 = -b*(wi + u2*h/2) + (ui + k2*h/2 )*(vi + q2*h/2);

 k4 = sig*((vi + q3*h) - (ui + k3*h));
 q4 = -(vi + q3*h) - (ui + k3*h )*(wi +u3*h) + r*(ui + k3*h );
 u4 = -b*(wi + u3*h) + (ui + k3*h)*(vi + q3*h);

   uf = ui + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
   vf = vi + (h/6)*(q1 + 2*q2 + 2*q3 + q4);
   wf = wi + (h/6)*(u1 + 2*u2 + 2*u3 + u4);

   ui = uf;
   vi = vf;
   wi = wf;
```

```
   sinal = uf + sin(500*t);   %Sinal embutido a ser transmitido.
   t = t + h;      %O tempo é incrementado de h. É mostrado a cada
interação.
   vri = vf;

%Receptor [ur,vr,wr].
a1 = sig*(vri - uri);
c1 = -b*wri + uf*vri;

a2 = sig*(vri - (uri + a1*h/2));
c2 = -b*(wri + c1*h/2) + uf*vri;

a3 = sig*(vri - (uri + a2*h/2));
c3 = -b*(wri + c2*h/2) + uf*vri;

a4 = sig*(vri - (uri + a3*h));
c4 = -b*(wri + c3*h) + uf*vri;

urf = uri + (h/6)*(a1 + 2*a2 + 2*a3 + a4);
wrf = wri + (h/6)*(c1 + 2*c2 + 2*c3 + c4);

uri = urf;
wri = wrf;

  %plot(t,urf,'r-')        %Comandos de "plotagem" a cada interação.
  %plot(t,uf,'k')
  plot(t,sin(5*t),'k');
  plot(t,sinal-uri,'b-');
  hold on   %Idem

end    %while

grid              %Comandos de "plotagem" finais.
xlabel('tempo (s)')
ylabel('Componente x(t)')
title('Recuperação da Mensagem')
%axis([0 10 -30 35]);    % Usado para plotar sinal de composto
axis([0 3 -1.5 1.5]);    % Usado para plotar recuperação da mensagem
```

Apêndice D – Linearização

Neste item procede – se com a Linearização do Sistema, passo a passo, do comunicação caótica segundo aproximação linear de Taylor até o termo de 1ª ordem. Definindo – se funções correspondentes às equações do sistema de Lorenz, pode – se escrever:

$$\begin{cases} f_1(x,y,z) = \dot{x} = \sigma(y-x) = 0 \\ f_2(x,y,z) = \dot{y} = rx - y - xz = 0 \\ f_3(x,y,z) = \dot{z} = xy - bz = 0 \end{cases}$$

Os pontos de equilíbrio encontrados, incluindo a solução trivial, são:

$(x_o, y_o, z_o) = (\pm 13.3566, \pm 13.3566, 44.6)$

$(x_o, y_o, z_o) = (0,0,0)$

Pelos resultados das simulações obtidos no Apêndice A, observa – se que um dos atratores, ou ponto de fixos de equilíbrio, possui as coordenadas: $(x_o, y_o, z_o) = (-13.3566, -13.3566, 44.5)$.

$$\frac{\partial f_1}{\partial x}\Big|_{(xo,\, yo,\, zo)} = -\sigma = -16$$

$$\frac{\partial f_1}{\partial y}\Big|_{(xo,\, yo,\, zo)} = \sigma = 16$$

$$\frac{\partial f_1}{\partial z}\Big|_{(xo,\, yo,\, zo)} = 0$$

O mesmo pode ser feito para as duas outras funções:

$$\frac{\partial f_2}{\partial x}\Big|_{(xo,\,yo,\,zo)} = 45.6 - z = 1$$

$$\frac{\partial f_2}{\partial y}\Big|_{(xo,\,yo,\,zo)} = -1$$

$$\frac{\partial f_2}{\partial z}\Big|_{(xo,\,yo,\,zo)} = -x = 13.3566$$

$$\frac{\partial f_3}{\partial x}\Big|_{(xo,\,yo,\,zo)} = y = -13.3566$$

$$\frac{\partial f_3}{\partial y}\Big|_{(xo,\,yo,\,zo)} = x = -13.3566$$

$$\frac{\partial f_3}{\partial z}\Big|_{(xo,\,yo,\,zo)} = -b = -4$$

Encontra – se, então, a matriz Jacobiana, que também é a matriz *A* quando se escreve a Linearização na forma de estado:

$$J = A = \begin{bmatrix} -16 & 16 & 0 \\ 1 & -1 & 13.3566 \\ -13.3566 & -13.3566 & -4 \end{bmatrix}$$

Linearização na forma de equações de Estado:

$$[\dot{X}] = A[X] + B[u]$$
$$[y] = C[X] + D[u]'$$

onde *A*, *B* e *C* são conforme o código abaixo.

Código fonte da simulação da análise de resposta em freqüência no MatLab.

```
% LINEARIZAÇÃO do Sistema de LORENZ
%
% Linearização por Taylor em torno dos pontos de equilibrio
% (fazendo as derivadas da variáveis de estado =0), cálculo
% da matris Jacobiana, análise espaço-estado.

clear all;

figure
% 1ª Variável
A=[-16 16 0;1 -1 13.3566;-13.3566 -13.3566 -4]
B=[1; 0; 0]
C=[1 0 0]
```

```
D=[0]
bode(A,B,C,D)              % Diagrama de Bode

figure
% 2ª Variável
A=[-16 16 0;1 -1 13.3566;-13.3566 -13.3566 -4]
B=[0; 1; 0]
C=[0 1 0]
D=[0]
bode(A,B,C,D)              % Diagrama de Bode

figure
% 3ª Variável
A=[-16 16 0;1 -1 13.3566;-13.3566 -13.3566 -4]
B=[0; 0; 1]
C=[0 0 1]
D=[0]
bode(A,B,C,D)              % Diagrama de Bode
```

A análise pela transformada rápida de Fourier foi feita com o algoritmo abaixo:

```
% Função transs_fft -simulação do sistema de lorenz sem sinal de mensagem
% e com analise do espectro em frequencia - FFT (com messagem).
%
% Frequência de Amostragem: (tmax / N)^-1 -> Hz
%
N=4*2048;       %número de pontos discretizados é potência de 2;
                %aumento da velocidade de computação da FFT.
index = 1;
tmax = 20;
t = linspace(0,tmax,N);
h = tmax / N;  % Periodo de amostragem - ou passo h.

%Parâmetros e inicializações...
sig = 16;    %Estes valores para "sig","r" e "b" tornam o sistema de
lorenz
r = 45.6;    %caótico.
b = 4;

xi = 10;     %Condicoes iniciais.
yi = 1.5;
zi = -6.5;

x=0;

%Método de Runge-Kutta 4a. ordem
while (index <= N)          %Condição de parada.

 mes = sin(10*x);

 k1 = sig*(yi - xi);
 q1 = -yi - (xi+ mes)*zi + r*(xi+ mes);
 u1 = -b*zi + (xi+ mes)*yi;
```

```
 k2 = sig*((yi + q1*h/2) - (xi + k1*h/2));
 q2 = -(yi + q1*h/2) - (xi + k1*h/2+ mes)*(zi +u1*h/2) + r*(xi + k1*h/2+
mes);
 u2 = -b*(zi + u1*h/2) + (xi + k1*h/2+ mes)*(yi + q1*h/2);

 k3 = sig*((yi + q2*h/2) - (xi + k2*h/2));
 q3 = -(yi + q2*h/2) - (xi + k2*h/2+ mes)*(zi +u2*h/2) + r*(xi + k2*h/2+
mes);
 u3 = -b*(zi + u2*h/2) + (xi + k2*h/2+ mes)*(yi + q2*h/2);

 k4 = sig*((yi + q3*h) - (xi + k3*h));
 q4 = -(yi + q3*h) - (xi + k3*h+ mes)*(zi +u3*h) + r*(xi + k3*h+ mes);
 u4 = -b*(zi + u3*h) + (xi + k3*h+ mes)*(yi + q3*h);

   xf = xi + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
   yf = yi + (h/6)*(q1 + 2*q2 + 2*q3 + q4);
   zf = zi + (h/6)*(u1 + 2*u2 + 2*u3 + u4);

   xi = xf;
   yi = yf;
   zi = zf;
   f(index) = xf;
   index = index + 1;
   x=x+h;
end    %while

F=fft(f);

Ws=2*pi/h;          % Frequencia de amostragem.

Fp=F(1:N/2+1)*h;    % Parte positiva de F e multiplica pelo peiodo de
                    % amostragem para estimar F(w).
W=Ws*(0:N/2)/N;     % Intervalo das abcissas - frequencia...

figure
Fp = 10*log(abs(Fp.*Fp));
plot(W,Fp)          % Plotagem da FFT em dB.
grid on
xlabel('Freqüência (rad/s)')
ylabel('|F(w)| em dB')
title('FFT sinal composto')
```

Apêndice E – Programas de Comunicação

O código mostrado abaixo corresponde às rotinas em linguagem *assembly* utilizadas tanto pelo algoritmo de comunicação analógica como para a digital; de utilização na placa ADDA II.

```
            SECTION rotinas

            GLOBAL Finicia_c
   Finicia_c:
            MOVE  #$0000,X:$FFED     ; General Purpose I/O Pins na porta
C.
            move  #$FFFF,X:$FFEE     ; Sets all bits as output
            ;bfclr      #$00FF,X:$FFED     ; clears only the 8 first LSB -
port C GPIO
            ;bfset      #$00FF,X:$FFEE     ; sets only the GPIO bits as
output
            RTS

            GLOBAL Finicia_b
   Finicia_b:
            ;MOVE    #$0000,X:$FFF9       ;0 wait states
            MOVE    #$0000,x:$FFEA        ;desabilita interrupcao
            MOVE    #$FF00,x:$FFEB        ;seleciona PB0-PB15 como saidas
            RTS

            GLOBAL Frti
   Frti:
            move    #$5555,X:$FFF0   ; sequencia de setup RTI
            nop
            move    #$AAAA,X:$FFF0
            nop
            move    #$5555,X:$FFF0
            nop
            bfclr   #$0800,X:$FFF1   ; clear RTE bit in COPCTL
            bfset   #$0400,X:$FFF1   ; set RTIE bit in COPCTL
            bfclr   #$0100,X:$FFF1   ; clr RP bit in COPCTL => div by 1
            bfset   #$0004,X:$FFF1   ; set DV[3] => division by 8
            bfset   #$0800,X:$FFF1   ; set RTE bit in COPCTL
            move    #$AAAA,X:$FFF0   ; disables write on COPCTL
            nop
            rts

            GLOBAL Fpwrite
   Fpwrite:
            MOVE  Y1,R0              ; Put address in pointer register
            NOP                      ; Must stall a cycle
            MOVE  Y0,P:(R0)+         ; Store
            rts

            GLOBAL Fpll
```

```
   Fpll:
            move  #$04C0,X:$FFF3    ; PS div by 256  -  CLKOUT Disabled
            move  #$7FE0,X:$FFF2    ; mult by 1024
            bfset #$4000,X:$FFF3    ; enable PLL
            rts

            GLOBAL Fenableints      ;This routine enables maskable
interrupts.
   Fenableints:
            BFCLR   #$0200,SR
            ;BFSET   #$8000,X:$FFFB ;para com. GPIO - portB
            ;BFSET   #$0200,X:$FFFB ;para com. SSI
            BFSET   #$4000,X:$FFFB  ;para RTI
            rts

            GLOBAL Fdisableints     ;This routine disables maskable
interrupts
   Fdisableints:
            BFSET   #$0200,SR
            ;BFCLR   #$8000,X:$FFFB ;para com. GPIO - port B
            ;BFCLR   #$0200,X:$FFFB ;para com. SSI
            BFCLR   #$4000,X:$FFFB  ;para RTI
            rts

            GLOBAL Fleitura
   Fleitura:
            move    #$3F00,X:$FFEB  ; PB0-PB6 = input e PB7-PB15 = output
            move    #$3F00,X:$FFEC  ; inicializa a PBD, tendo PB8..PB15 em
"1s"
            ;bfclr   #$0700,X:$FFEC ; define canal 1, com valor A0 = A1 =
A2 = "0"
            ;bfclr   #$0600,X:$FFEC ; define canal 2, com valor A0 = "1",
A1 = A2 = "0"
            ;bfclr   #$0500,X:$FFEC ; define canal 3, com valor A0 = "0",
A1 = "1", A2 = "0"
            ;bfclr   #$0400,X:$FFEC ; define canal 4, com valor A0 = A1 =
"1", A2 = "0"
            ;bfclr   #$0300,X:$FFEC ; define canal 5, com valor A0 = A1 =
"0", A2 = "1"
            ;bfclr   #$0200,X:$FFEC ; define canal 6, com valor A0 = "1",
A1 = "0", A2 = "1"
             bfclr   #$0100,X:$FFEC  ; define canal 7, com valor A0 = "0",
A1 = A2 = "1"
            ;bfclr   #$0000,X:$FFEC ; define canal 8, com valor A0 = A1 =
A2 = "1"
            bfclr   #$0800,X:$FFEC  ; abaixa (clear) sinal *CS A/D
            bfclr   #$1000,X:$FFEC  ; abaixa (clear) sinal *RD A/D
;
            nop         ; pequeno pulso de *RD, pois na borda de subida
                  ; habilita o latch de DB0..DB7
                  ; no tempo de tRD = 80 nS
                  ; e tACC1 e o tempo de acesso apos o sinal *RD
                  ; que e de 110 nS

            bfset   #$1000,X:$FFEC  ; levanta (set) sinal *RD A/D
            move    #20,x0    ; intervalo entre leituras, tP = 500 nS
(tip.)
```

```
   CONT:          ; entre termino sinal *INT e proxima conversao
                  decw  x0    ; + tCRD = 2,4 (tip.) perfazem um tempo medio
                  bgt   CONT; de 2,9 uS entre abaixar sinal *CD entre leituras

                  ; Estranhamente utilizou - se inicialmente 20 nop's
                  ; (mais que isso dava erro na compilacao) para gastar
                  ; os 3micro seg.; so gastou 2.3micro seg....

                  bfclr   #$1000,X:$FFEC  ; abaixa (clear) sinal *RD A/D
                  nop                     ; pequeno delay
                  move    X:$FFEC,Y0      ; valor da leitura do A/D

                  bfset   #$1000,X:$FFEC  ; levanta (set) sinal *RD A/D
                  ;bfset   #$0700,X:$FFEC  ; define canal 8, com valor A0 = A1
= A2 = "1"
                  bfset   #$0800,X:$FFEC  ; levanta (set) sinal *CS A/D

andc    #$00FF,Y0       ; mascara para X0, apenas os 8 bits do A/D
RTS

                  GLOBAL Fleitura2
   Fleitura2:

                  move    #$3F00,X:$FFEB  ; PB0-PB6 = input e PB7-PB15 = output
                  move    #$3F00,X:$FFEC  ; inicializa a PBD, tendo PB8..PB15
em "1s"

                  ;bfclr   #$0700,X:$FFEC  ; define canal 1, com valor A0 = A1
= A2 = "0"
                  bfclr   #$0600,X:$FFEC  ; define canal 2, com valor A0 = "1",
A1 = A2 = "0"
                  bfclr   #$0800,X:$FFEC  ; abaixa (clear) sinal *CS A/D
                  bfclr   #$1000,X:$FFEC  ; abaixa (clear) sinal *RD A/D

                  nop                     ; pequeno pulso de *RD, pois na borda
de subida
                  ; habilita o latch de DB0..DB7
                  ; no tempo de tRD = 80 nS
                  ; e tACC1 e o tempo de acesso apos o sinal *RD
                  ; que e de 110 nS
                  bfset   #$1000,X:$FFEC  ; levanta (set) sinal *RD A/D

                  move    #20,x0     ; intervalo entre leituras, tP = 500 nS
(tip.)
   CONT2:         ; entre termino sinal *INT e proxima conversao
                  decw    x0   ; + tCRD = 2,4 (tip.) perfazem um tempo medio
                  bgt     CONT2     ; de 2,9 uS entre abaixar sinal *CD entre
leituras

                  bfclr   #$1000,X:$FFEC  ; abaixa (clear) sinal *RD A/D
                  nop                     ; pequeno delay
                  move    X:$FFEC,Y0      ; valor da leitura do A/D

                  bfset   #$1000,X:$FFEC  ; levanta (set) sinal *RD A/D
                  ;bfset   #$0700,X:$FFEC  ; define canal 8, com valor A0 = A1
= A2 = "1"
                  bfset   #$0800,X:$FFEC  ; levanta (set) sinal *CS A/D
```

```
            andc    #$00FF,Y0       ; mascara para X0, apenas os 8 bits
do A/D
            RTS

            GLOBAL Fescrita
   Fescrita:
            move    #$3FFF,X:$FFEB    ; Pinos PB0-PB15 como output
            bfset   #$3F00,X:$FFEC
            bfclr   #$0800,X:$FFEC    ; abaixa (clear) sinal *LDAC D/A

            bfclr   #$0700,X:$FFEC    ; define canal 1, com valor A0 = A1
= "0"
            ;bfclr   #$0600,X:$FFEC    ; define canal 2, com valor A0 =
"1", A1 = "0"
            ;bfclr   #$0500,X:$FFEC    ; define canal 3, com valor A0 =
"0", A1 = "1"
            ;bfclr   #$0400,X:$FFEC    ; define canal 4, com valor A0 =
A1 = "1"

            bfclr   #$2000,X:$FFEC    ; abaixa (clear) sinal *WR A/D

            BFCLR   #$FF00,Y0
            move    Y0,X:$FFEC

            bfset   #$2000,X:$FFEC    ; levanta (set) sinal *WR D/A
            bfset   #$2F00,X:$FFEC    ; levanta (set) todos os outros
sinais
            RTS

            ENDSEC
```

O código fonte para o sub – sistema transmissor é mostrado abaixo.

```
#include <stdlib.h>
#include <stdio.h>

#define h 0.001

__fixed__ bin[50];
int k = 0;
int index=0;
int e=0;
float t = 0;
static char mesg[10] = "hello";


__fixed__ uf,ui;
__fixed__ vf,vi;
__fixed__ wf,wi;
__fixed__ urf,uri;
__fixed__ vrf,vri;
__fixed__ wrf,wri;

__fixed__ aux;
__fixed__ temp1;
__fixed__ temp2;
```

```
__fixed__ mes = 0;

__fixed__ s[448]={      //pontos de onda triangular só para h=0.008;
...                     //pontos divididos por 16
}

typedef unsigned short WORD;

//Prototypes

void inicia_b();
//void inicia_c();
void escrita(__fixed__);      //Funcão de saida pelo CI D/A.
void dac0(__fixed__);
void dac1(__fixed__);
void HextoBin(int);

void Sist_Trans();            //Algoritmo calcula dado a ser transmitido

void pwrite(WORD, WORD);      // writes to P memory
void rti(void);               // setup for RTI
void pll(void);               // setup Prescaler for division: 20MHz / 256
void enableints(void);
void disableints(void);

/*************************************************************************/
/* Funcao ASCII                                                          */
/*************************************************************************/
__fixed__ ASCII(){

if(e==1000){
  if(k==40)
    k = 0;
  k = k + 1;
  e=0;
}
  e++;
  return bin[k];
}

void t_mesg(void){

int i=0;

while(i < 5){
  HextoBin(mesg[i] / 16);
  index = index + 4;
  HextoBin(mesg[i] % 16);
  index = index + 4;
  i++;
}
}

void HextoBin(int info){
int i;

 if(info == 0xF)
```

```
    for(i=0;i<4;i++)
     bin[index+i] = 0.007805;
  if(info == 0xE)
    for(i=0;i<3;i++)
     bin[index+i] = 0.007805;
  if(info == 0xD){
    bin[index] = 0.007805;
    bin[index+1] = 0.007805;
    bin[index+3] = 0.007805;}
  if(info == 0xC){
    bin[index] = 0.007805;
    bin[index+1] = 0.007805;}
  if(info == 0xB){
    bin[index] = 0.007805;
    bin[index+2] = 0.007805;
    bin[index+3] = 0.007805;}
  if(info == 0xA){
    bin[index] = 0.007805;
    bin[index+2] = 0.007805;}
  if(info == 0x9){
    bin[index] = 0.007805;
    bin[index+3] = 0.007805;}
  if(info == 0x8)
    bin[index] = 0.007805;
  if(info == 0x7){
    bin[index+1] = 0.007805;
    bin[index+2] = 0.007805;
    bin[index+3] = 0.007805;}
  if(info == 0x6){
    bin[index+1] = 0.007805;
    bin[index+2] = 0.007805;}
  if(info == 0x5){
    bin[index+1] = 0.007805;
    bin[index+3] = 0.007805;}
  if(info == 0x4)
    bin[index+1] = 0.007805;
  if(info == 0x3){
    bin[index+2] = 0.007805;
    bin[index+3] = 0.007805;}
  if(info == 0x2)
    bin[index+2] = 0.007805;
  if(info == 0x1)
    bin[index+1] = 0.007805;
}


/*************************************************************************/
/* Funcao Seno                                                           */
/*************************************************************************/

__fixed__ seno(){
  if(k==447)
     k = 0;
  k = k + 1;

  return s[k];
}
```

```
/*************************************************************************/
/* Funcao Principal                                                      */
/*************************************************************************/
void main(void){

int i,j;
j=0;

  pwrite((WORD)0xE9C8,(WORD)0x0016);                  // Write JSR
  pwrite((WORD)Sist_Trans,(WORD)0x0017); // Write actual vector

  pll();

  inicia_b();
  //inicia_c();
  /*
  printf("%s = ",mesg);
  printf("%x ",mesg[0]);
  printf("%x ",mesg[1]);
  printf("%x ",mesg[2]);
  printf("%x ",mesg[3]);
  printf("%x\n",mesg[4]);
  */
  for(i=0;i<50;i++)
    bin[i] = 0;

  t_mesg();
/*
  for(i=0;i<40;i++){
  printf("%d ",bin[i]);
  j++;
  if(j==8){
    printf("\n");
    j=0;}
  }
*/

ui = 0.09375;           // 1.5 / 16
vi = -0.03125;          //-0.5 / 16
wi = -0.06875;          //-1.1 / 16

  //printf("Transmition Program Initiated!\n");
  //printf("%f\n",h);

  rti();
  enableints();

//  while (t<10.0){                //Condicao de parada.
  while(1){
  //  Sist_Trans();
  }   //while

  //printf("End of Program!\n");
}
```

```
/*************************************************************************/
/* Funcao Sist_Trans                                                     */
/*************************************************************************/
void Sist_Trans(){

disableints();
//asm {bfset      #$00FF,X:$FFEF}

mes = ASCII();
//mes = seno();

uf = 0.512*(vi - ui);
uf = (uf>>5) + ui;

ui = ui + mes;

vf = 0.7296*ui;
aux = 0.512*vi;
aux = aux>>5;
vf = vf - aux;
vf = vf>>4;
aux = 0.64*ui;
aux = aux*wi;
aux = aux>>1;
vf =  vf + vi - aux;

wf = 0.64*ui;
wf = wf*vi;
wf = wf>>3;
aux = 0.512*wi;
aux = aux>>7;
wf = wf + wi - aux;

ui = uf;
vi = vf;
wi = wf;

dac0(mes);
temp1 = ui + mes + 0.22;      // + 7208 para evitar dados negativos
temp2 = temp1 >> 8;
dac1(temp2);

t=t + h;                      //O tempo eh incrementado de h.

//asm {bfclr      #$00FF,X:$FFEF}
enableints();

}           //end funcao transmissor.

void  dac0(__fixed__){
asm
      {
            move  mes,x0;           //    dado para x0
            rol   x0
            rol   x0
            rol   x0
            rol   x0
```

```
            rol   x0
            rol   x0
            rol   x0
            rol   x0
            andc  $FF00,x0;          // limpa bits de ctrl de x0
            orc   #$00F0,x0          // soma bits ctrl com dado em X0
            move  x0,X:$FFEC         // x0 para Porta B
            move  #$FF3F,X:$FFEB     // direcao dados Output
            bfclr #$0020,X:$FFEC     // abaixa sina *WR
            nop
            nop
            bfset #$0021,X:$FFEC
            rts
      }
}

void  dac1(__fixed__){
asm
      {
            move  temp2,x0;          //    dado para x0
            rol   x0
            rol   x0
            rol   x0
            rol   x0
            rol   x0
            rol   x0
            rol   x0
            rol   x0
            andc  #$FF00,x0;         // limpa bits de ctrl de x0
            orc   #$00F8,x0          // soma bits ctrl com dado em X0
            move  x0,X:$FFEC         // x0 para Porta B
            move  #$FF3F,X:$FFEB     // direcao dados Output
            bfclr #$0020,X:$FFEC     // abaixa sina *WR
            nop
            nop
            bfset #$0021,X:$FFEC
            rts
      }
}
```

O algoritmo de recepção é mostrado simplificadamente abaixo.

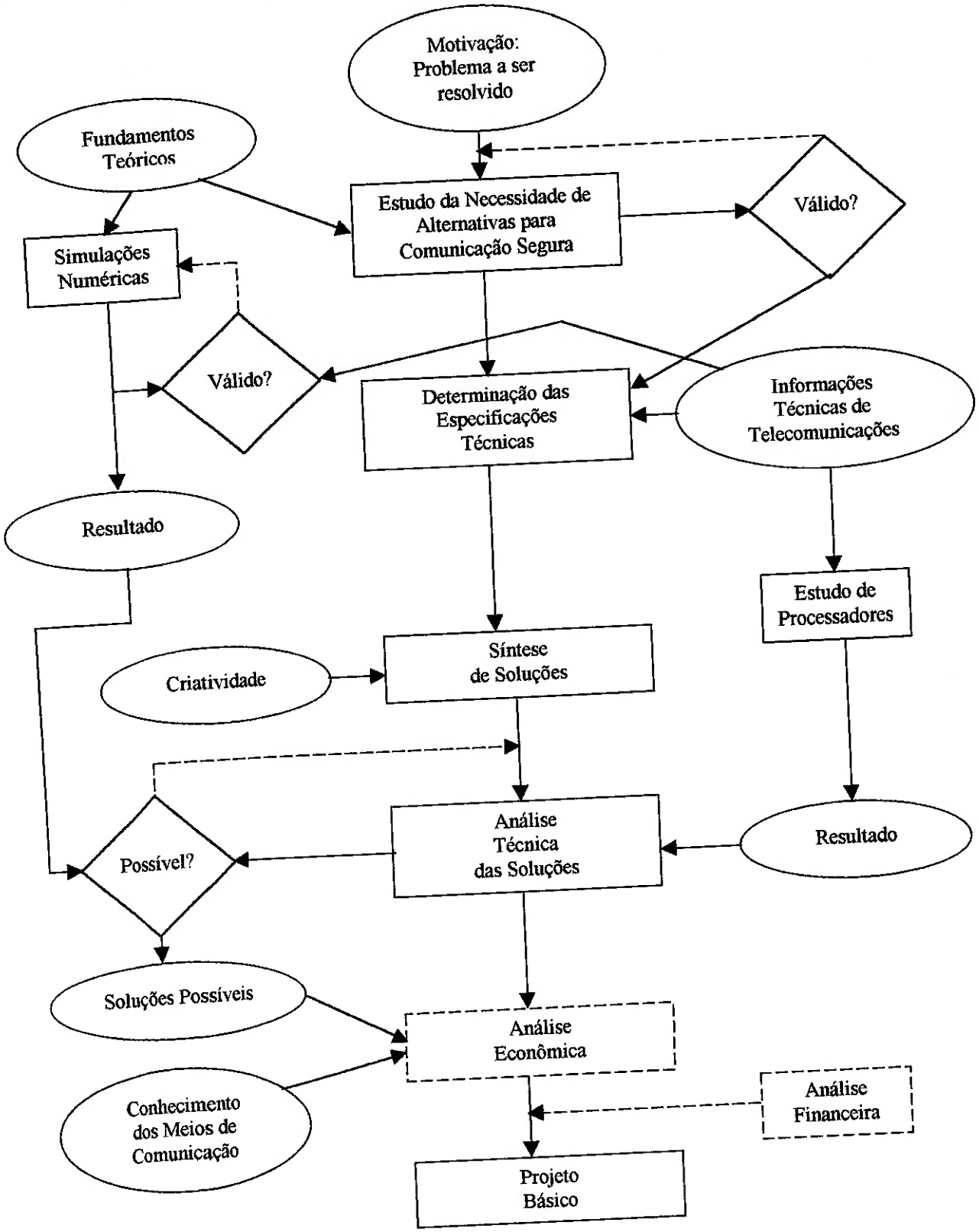
```
#include <stdlib.h>
#include <stdio.h>

#define h 0.001

float t = 0;
int i;

__fixed__ urf,uri;
__fixed__ vrf,vri;
__fixed__ wrf,wri;

__fixed__ aux;
__fixed__ temp;
```

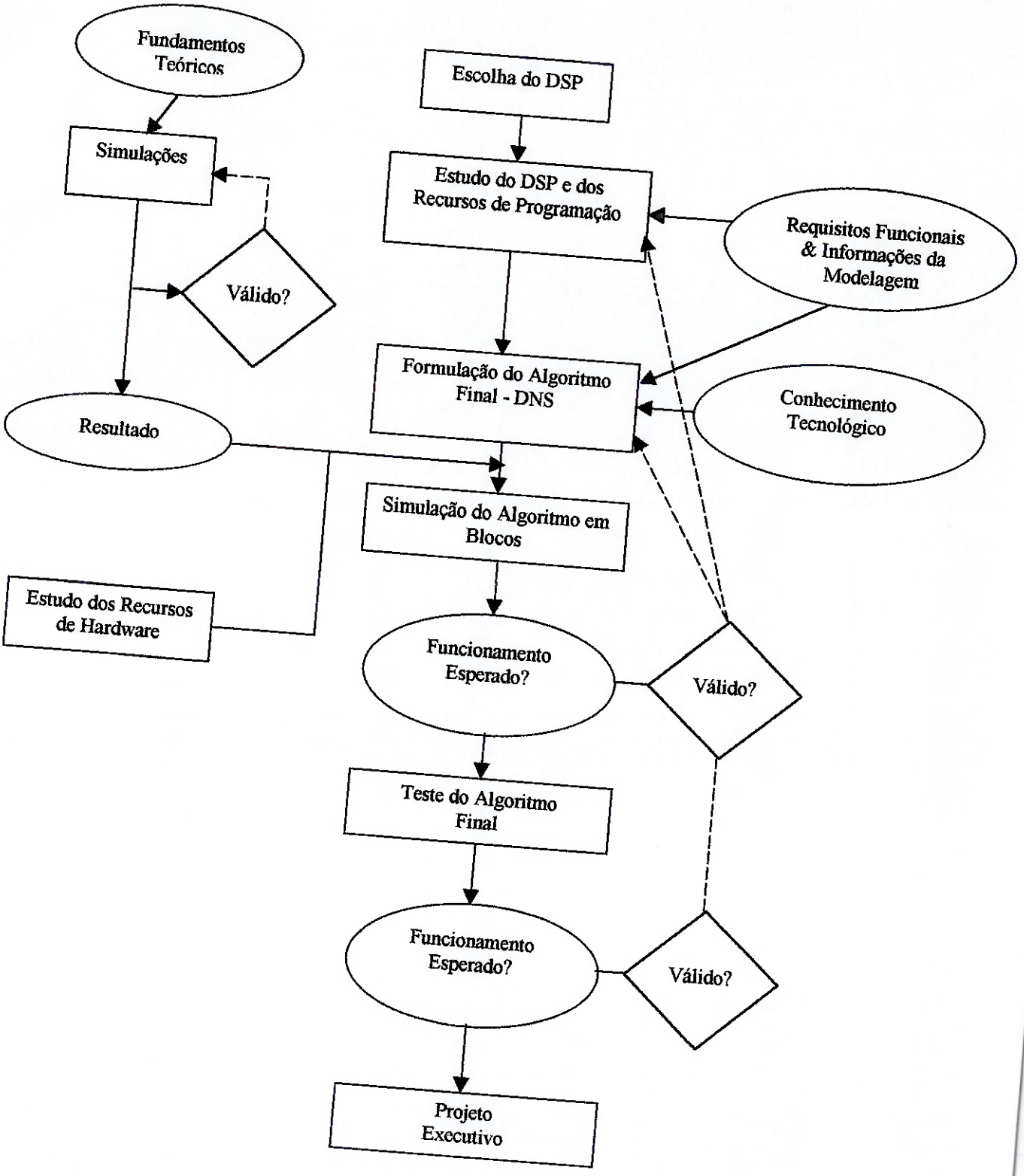
```
typedef unsigned short WORD;

//Prototypes

void inicia_b();
void inicia_c();
__fixed__ leitura(void);      //Funcão com bits de entrada PB8-PB15 com o
sinal de READ.
void escrita(__fixed__);      //Funcão de saida pelo CI D/A.

//__fixed__ leitura2(void);   //Funcão com bits de entrada PB8-PB15 com o
sinal de READ.

void Sist_Receiv(void);

void pwrite(WORD, WORD);      // writes to P memory
void rti(void);               // setup for RTI
void pll(void);               // setup Prescaler for division: 20MHz / 256
void enableints(void);
void disableints(void);

/*********************************************************************/
/* Funcao Principal                                                  */
/*********************************************************************/
void main(void){

  inicia_b();
  inicia_c();

  //pwrite((WORD)0xE9C8,(WORD)0x0016);          // Write JSR
  //pwrite((WORD)Sist_Receiv,(WORD)0x0017);     // Write actual vector

  //pll();

uri = 0.0625;           // 1 / 16
vri = 0.0625;           // 1 / 16
wri = 0.0625;           // 1 / 16

  //printf("Reception Program Initiated!\n");
  //printf("%f\n",h);

  //rti();
  //enableints();

  //while (t<10.0){               //Condicao de parada.
  while(1){

  Sist_Receiv();

  }   //while

  //printf("End of Program!\n");

}
```

```
/*****************************************************************/
                                                                 */
/* Funcao Sist_Receiv
/*****************************************************************/
void Sist_Receiv(){

//disableints();
asm {bfset  #$00FF,X:$FFEF}
//asm {eorc #$00FF,X:$FFEF}

temp = leitura();
//temp = leitura2();


temp = temp << 8;
temp = temp - 0.216098;       //-0.22 (eliminar negativos) + 0.003902
(mais 127)

urf = 0.512*(vri - uri);
urf = (urf>>5) + uri;

vrf = 0.7296*temp;
aux = 0.512*vri;
aux = aux>>5;
vrf = vrf - aux;
vrf = vrf>>4;
aux = 0.64*temp;
aux = aux*wri;
aux = aux>>1;
vrf =  vrf + vri - aux;

wrf = 0.64*temp;
wrf = wrf*vri;
wrf = wrf>>3;
aux = 0.512*wri;
aux = aux>>7;
wrf = wrf + wri - aux;

uri = urf;
vri = vrf;
wri = wrf;

t=t + h;                          //O tempo eh incrementado de h.

//temp = uri;
temp = temp - uri;
temp = temp + 0.22;
temp = temp >> 8;

escrita(temp);
//printf("%d\n",);

for(i=0;i<42;i++);

asm {bfclr  #$00FF,X:$FFEF}
//enableints();

}           //end receptor.
```

Apêndice F - Fluxograma do Projeto

Um fluxograma das atividades realizadas pode ser elaborado. No primeiro fluxograma, as setas pontilhadas significam que houve um processo interativo de análise. No caso dos bloco de análise econômica e financeira o pontilhados indica que não houve qualquer tipo de estudo de responsabilidade.


Motivação:
Problema a ser
resolvido
Fundamentos
Teóricos
Estudo da Necessidade de
Alternativas para
Comunicação Segura
Válido?
Simulações
Numéricas
Válido?
Determinação das
Especificações
Técnicas
Informações
Técnicas de
Telecomunicações
Resultado
Estudo de
Processadores
Síntese
de Soluções
Criatividade
Análise
Técnica
das Soluções
Possível?
Resultado
Soluções Possíveis
Análise
Econômica
Conhecimento
dos Meios de
Comunicação
Análise
Financeira
Projeto
Básico


Fundamentos Teóricos
Escolha do DSP
Simulações
Estudo do DSP e dos Recursos de Programação
Requisitos Funcionais & Informações da Modelagem
Válido?
Formulação do Algoritmo Final - DNS
Conhecimento Tecnológico
Resultado
Simulação do Algoritmo em Blocos
Estudo dos Recursos de Hardware
Funcionamento Esperado?
Válido?
Teste do Algoritmo Final
Funcionamento Esperado?
Válido?
Projeto Executivo