MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-13006-PRE/8283

SISTEMA DE CONTROLE DE APONTAMENTO PARA A ANTENA DA ESTALAÇÃO TT&C DE NATAL

Kurios Iuri Pinheiro de Melo Queiroz*

*Bolsista UFRN

Relatório Final de Projeto de Iniciação Científica (PIBIC/CNPq/INPE), orientado pelo Dr. Manoel Jozeane Mafra de Carvalho

INPE
São José dos Campos
2005

SISTEMA DE CONTROLE DE APONTAMENTO PARA A ANTENA DA ESTAÇÃO TT&C DE NATAL

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

Kurios Iuri Pinheiro de Melo Queiroz (UFRN, Bolsista PIBIC/CNPq)
E-mail: kurios@crn.inpe.br

Manoel Jozeane Mafra de Carvalho (INPE, Orientador)
E-mail: manoel@crn.inpe.br

COLABORADORES

Dr. Francisco das Chagas Mota (DCA/UFRN)

Maio de 2005

Resumo

Este trabalho consiste na pesquisa e desenvolvimento de um sistema de controle para uma antena, no Instituto Nacional de Pesquisas Espacias (INPE) em cooperação com a Universidade Federal do Rio Grande do Norte (UFRN), cuja finalidade é o rastreio de satélites. A sua concepção vai desde o acionamento e proteção do sistema, até as técnicas de controle empregadas para solução do problema. O projeto tem como base a estação SACI (atualmente desativada) cuja estrutura física (cabos, antena, motores, inversores, etc.) é totalmente aproveitada. Toda a parte de software é desenvolvida utilizando-se a plataforma GNU/Linux e os pacotes do projeto Comedi, responsável pelos drives da placa conversora analógica-digital (AD/DA), e pelas bibliotecas de programação na linguagem C. Testes preliminares com um controlador PI já foram realizados com resultados satisfatórios. Algoritmos mais elaborados como MRAC (Model Reference Adaptive Control)e o VS-MRAC (Variable Structure Model Reference Adaptive Control) para plantas com parâmetros desconhecidos ou conhecidos com incerteza, já estão sendo estudados para posterior aplicação. A robustez do sistema também é alvo de estudos, prevendo dinâmica não-modelada e situações inesperadas, como rajadas de vento e chuva constante. Apenas o controle de posição da antena é implementado, pois inversores realizam o controle de velocidade dos motores.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O sistema de controle de posição em desenvolvimento tem como finalidade a substituição da antiga estação de rastreio, SACI, aproveitando-se boa parte de sua estrutura física e conceitos fundamentais. Um novo bloco de controle será criado, englobando as funções de proteção do sistema (sensores de fim de curso, defeito nos inversores, etc.) e controle de posição, que será realizado inicialmente por controlador PID digital. Para efeito didático, a estação de rastreio anterior será chamada de SACI, e o sistema atual em pesquisa de CITOSINA.

A figura 1 apresenta o diagrama de blocos do sistema, com suas malhas de controle de posição (mais externa) e velocidade (mais interna). Este último, não é realizado pelo CITOSINA, uma vez que o inversor já trata desse aspecto com um PID junto ao motor. Já o controle de posição será o alvo principal desse estudo, com a finalidade de prover meios para construção de um algoritmo de rasteio, em que a antena deve seguir a trajetória de um satélite.

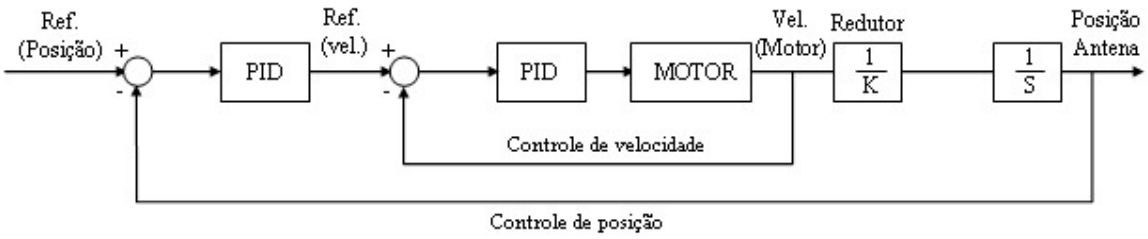

Figura 1: Diagrama de blocos do sistema

A metodologia utilizada na concepção do CITOSINA corresponde a modelagem do problema com base na estação anterior, estudando seus manuais e demais itens, para uma solução mais rápida e comercial possível. Em virtude do porte do projeto, foi adotado o sistema operacional Linux, que reduz os custos com licenças para *softwares* e aumenta sua confiabilidade.

O projeto atual pode ser dividido da seguinte forma:

- Plataforma operacional
- Posicionamento
- Acionamento
- Algoritmos de controle

A seção "Plataforma operacional" descreve a estação de trabalho e os *softwares* utilizados na construção dos algoritmos. As seções seguintes, "Posicionamento" e "Acionamento", abordam respectivamente o sistema de posição da antena e sua parte de potência (motores, inversores, etc.). Em "Algoritmos de controle", são descritas as técnicas de controle utilizadas e estudos para implementações futuras. Por fim, em "Conclusões e considerações finais" são expostos os resultados e problemas encontrados durante a pesquisa e desenvolvimento da estação.

2 Plataforma operacional

Para o algorítmo de controle de posição da antena é utilizado um computador com sistema operacional Linux (distribuição Debian 3.0r0) apenas em modo texto, o que deixa a máquina com uma carga computacional menor. Nela, foi adicionada uma placa conversora analógica-digital (*National 6023E PCI*) responsável pela comunicação entre a máquina e os dispositivos externos (placa SOTEREM, inversor, circuitos, etc.). O conjunto de drives utilizados para reconhecimento da placa e bibliotecas de programação foi o *Comedi*. A instalação e configuração do *Comedi* e do *Comedilib* não serão descritos neste relatório.

Todos os programas usados para aquisição de dados e envio de comandos, foram desenvolvidos em C utilizando-se bibliotecas do projeto GNU (GCC 2.95) e *Comedi* (*Comedilib*). O kernel do sistema atual é o 2.4.25 sem patches para obtenção de dados em tempo real (*real tasks*), o que pode ser necessário, em virtude de testes realizados com algoritmos de controle na antena (PI). Pretende-se futuramente utilizar o RTLinux ou o RTAI (preferencialmente), que permitem ao kernel trabalhar com tarefas em tempo real, aumentando a prioridade dos processos.

A primeira tarefa realizada na máquina foi a instalação do sistema operacional (SO) e a recompilação do kernel, dando suporte apenas ao hardware nativo do PC. Isso deixou o SO mais rápido para trabalhar, pois o kernel padrão da Debian (bf2.4) traz suporte a diversos dipositivos de hardware. O *Comedi* e o *Comedilib* foram adicionados ao sistema logo em seguida, sendo realizado alguns testes para familiarização.

Para uma melhor organização dos *softwares* produzidos para o projeto, um servidor CVS (*Version Control System*) também foi instalado e devidamente configurado.

3 Posicionamento

O sistema de posição da antena é basicamente dividido em três partes: captação, processamento e multiplexação. A primeira é realizada pelos *resolvers* acoplados nos eixos da antena[1], que fornecem sinais analógicos referentes ao angulo de seu rotor. Esses sinais são convertidos para um formato digital (processamento), e em seguida multiplexados para análise no computador. A figura 2 mostra as partes do sistema de posição e sua interação entre eles.

[1] Os *resolvers* não estão presos diretamente nos eixos da antena, pois existe ainda a presença de um redutor coaxial e de um sistema de engrenagens entre eles.

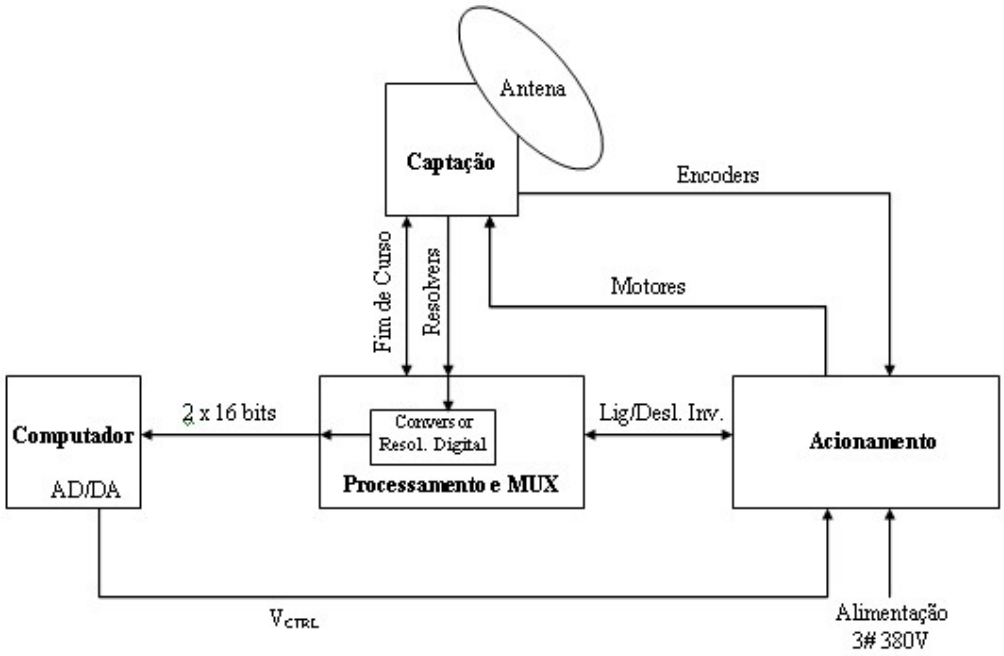

Figura 2: Sistema de posição da antena

3.1 Resolvers

São sensores que medem a posição angular instantânea dos eixos da antena, azimute e elevação. Eles fornecem dois sinais alternados, seno e coseno, quando excitados por uma referência AC, funcionando como uma espécie de transformador, onde o primário encontra-se normalmente no rotor e o secundário no estator. Com esses sinais, é possível o cálculo da posição angular de seu rotor.

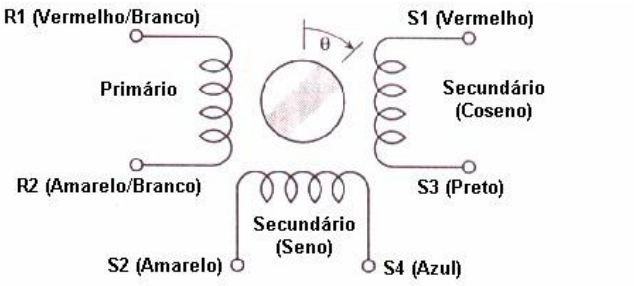

Figura 3: Esquema do resolver

3.2 Placa SOTEREM 2266-11

A placa SOTEREM 2266-11 é responsável pela conversão dos sinais analógicos do seno, cosseno e referência em um formato digital de 10, 12, 14 ou 16 bits[2] referente à posição da antena. Aparentemente, ela também é responsável pela geração do sinal alternado que alimenta os resolvers e os RDC´s. Os componentes básicos desta placa são: o CI RDC19220 e o CI LM324N. O primeiro é responsável pela conversão dos sinais

[2]A placa foi configurada para fornecer a posição em 16 bits

analógicos em digitais e o segundo pela geração da onda AC. A figura 4 apresenta a placa 2266-11 evidenciando suas entradas para alimentação, sinais dos resolvers (AC) e coleta de dados (posição da antena no formato digital).

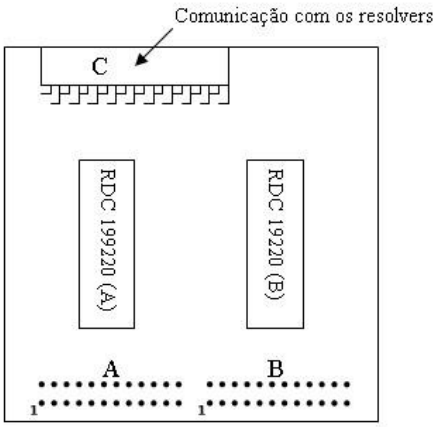

Figura 4: Placa SOTEREM 2266-11

A comunicação da placa 2266-11 se dá basicamente por três portas: A, B e C. A primeira refere-se ao posicionamento da antena no eixo de elevação e a segunda ao eixo de azimute. Ambas (portas A e B) são também utilizadas na alimentação da placa, sendo necessárias tensões de $+-12V$ e $+5V$. A porta C é destinada a comunicação com os resolvers.

Na figura 4, os RDC´s A e B destinam-se respectivamente a conversão das posições de elevação e azimute, disponabilizando seus resultados nas portas A e B citadas anteriormente. A tabela 1 descreve a sáida de dados das portas A e B enquanto a tabela 2 apresenta a alimentação da placa. É interessante ressaltar que a pinagem dessa placa segue a descrição da figura 4, em que os pinos ímpares se encontram na parte de baixo (ver indicação do pino 1) e os pares na parte de cima. Esta identificação é de suma importância para que a placa funcione perfeitamente, evitando assim a danificação total ou parcial de seus componentes.

Descrição	Porta A	Porta B
MSB	21	15
A14	22	16
A13	19	13
A12	20	14
A11	17	11
A10	18	12
A09	15	9
A08	16	10
A07	13	7
A06	14	8
A05	11	5
A04	12	6
A03	9	3
A02	10	4
A01	7	1
LSB	8	2

Tabela 1: Saída de dados da placa 2266-11

Pino/Porta	Tensão
1A	-12V
2A	+12V
3A 4A 21B 22B	GND
5A 6A 23B 24B	+5V

Tabela 2: Alimentação da placa 2266-11

Para a identificação do resolver, também será adotada a nomenclatura descrita anteriomente (A - elevação e B - azimute). A tabela 3 disponabiliza a pinagem da porta C, diferenciando os resolvers de cada eixo.

Função	Resolver A	Resolver B
+S	05	03
-S	19	23
+C	06	04
-C	39	24
+REF	40	20
-REF	25	21

Tabela 3: Entrada de dados da placa 2266-11

No sistema SACI a placa da SOTEREM encontra-se conectada a uma segunda placa, GESPAC modelo GESPIA-2AW, e somente a partir do seu manual é que foi possível a identificação dos pinos de alimentação para a 2266-11.

3.3 Procedimento para leitura

Para que seja possível a leitura dos dados referente a posição da antena, os bits de controle 25A e 19B da placa 2266-11 devem ser ativados[3]. Quando estão em nível lógico alto, os CI's (RDC 19220) encontram-se realizando o processo de conversão dos sinais, deixando a saída de dados com alta impedância. Em nível lógico baixo, o processo de conversão é inibido, e o resultado da última conversão é disponabilizado (os pinos de saída ficam com baixa impedância).

Para testes iniciais com a antena, apenas os 10 bits mais significativos foram coletados de cada posição, através de um circuito multiplexador (solução temporária) montado com componentes discretos. Essa solução foi adotada para viabilizar os estudos de posicionamento e controle inicial da antena pelo computador. O esquema do circuito para multiplexação utilizado está descrito na figura 5, cuja sequência de seleção dos MUXs é realizada por flip-flops J-K[4]. Com a chegada de uma nova placa, com mais canais digitais, os 16 bits de cada posição (azimute e elevação) serão ligados em paralelo e depois lidos em dois blocos. Inicialmente o pino 25A será habilitado e o 19B desabilitado, deixando o primeiro com a saída de dados em baixa impedância, e o segundo em alta. No instante seguinte, os estados serão invertidos e uma nova leitura executada.

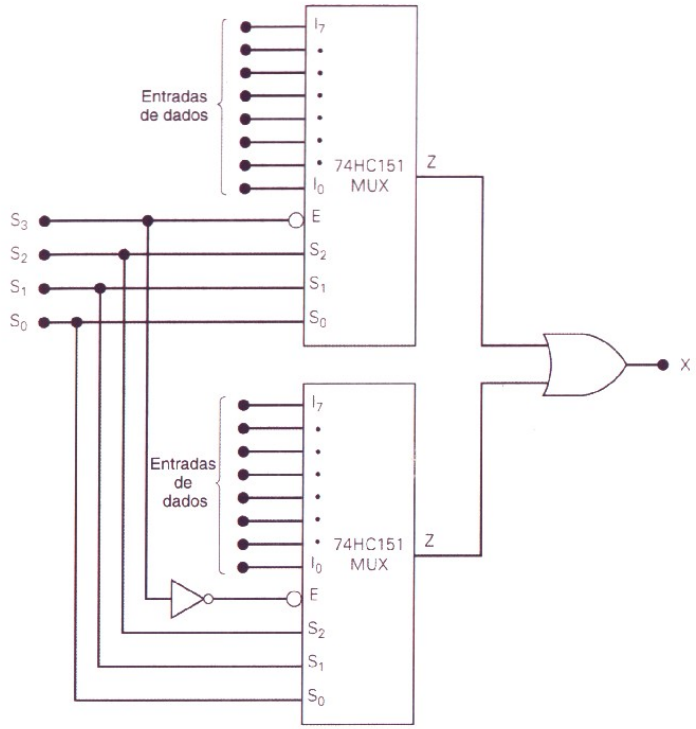

Figura 5: Circuito multiplexador de 16 bits

[3]Esses pinos são ativados em nível lógico baixo

[4]O clock do sistema é fornecido pelo computador através da placa AD

3.4 Calibração da posição da antena

Um fator importante para o controle da planta corresponde a escolha adequada do sistema de posição para a antena. Fisicamente, ela pode girar de 0 a 360 graus no eixo azimute, e de -3 a 182 graus na elevação. Essa folga de alguns graus para cima e para baixo na elevação é utilizada pelos sensores de fim de curso para acionar o sistema de proteção e desligar a parte de potência, caso a antena não pare.

Quando os resolvers foram instalados no eixo da antena, o seu zero não foi colocado no zero desejado para o sistema, o que requer uma posterior adaptação na leitura, em virtude da nova referência. A partir daí, uma equivalência entre escalas foi realizada, levando-se em consideração os valores extremos para cada uma. Além disso, existe uma diferença na variação de posição do resolver elevação e na posição física da antena. Enquanto a mesma pode girar um pouco mais que 180 graus, o eixo do resolver gira aproximadamente 313, provendo uma maior definição na leitura realizada. Inicialmente, apenas a equivalência para a elevação foi realizada, pois somente sua posição está sendo coletada adequadamente. De acordo com a figura 6, podemos deduzir a equação 1, em que X representa o valor lido pelo resolver e Y o valor da nova posição.

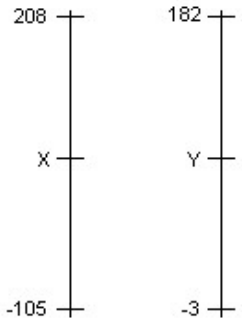

Figura 6: Equivalência entre escalas na elevação

$$\begin{aligned} \frac{X+150}{208+150} &= \frac{Y+3}{182+3} \\ 185 \cdot (X+105) &= 313 \cdot (Y+3) \\ \frac{185}{313} \cdot (X+105) &= Y+3 \\ 0.6 \cdot (X+105) &= Y+3 \\ Y &= 0.6X + 60 \end{aligned} \tag{1}$$

4 Acionamento

Os motores utilizados no projeto SACI, e atualmente no CITOSINA, são do tipo autosíncronos (ímas permanentes) trifásicos, acionados através de um inversor/*drive* UMV 4301. O parâmetro de entrada para controle dos motores corresponde a uma tensão DC (V_{ctrl}) na faixa de -10V a 10V, com limite máximo de velocidade igual a 3000 rpm, tanto no sentido horário, quanto no sentido anti-horário.

Inicialmente, ao ligar o inversor pela primeira vez em conjunto com o motor, alguns procedimentos básicos devem ser realizados, como por exemplo o "faseamento" (*autotune*) motor-inversor. O procedimento de *autotune* permite ao inversor, o conhecimento de parâmetros fundamentais do motor para execução do controle de velocidade interno (PID). A cada troca de motor ou inversor, este procedimento deve ser realizado para perfeito funcionamento de ambos. Caso contrário, o erro ENC PH9 será mostrado no visor do drive, acusando problemas no resolver ou encoder.

O inversor necessita ainda de mais algumas informações (parâmetros), que devem ser fornecidas pelo o usuário (quantidade de pólos do motor, frequência de alimentação, tipo do motor, etc.) através de seu teclado. O item seguinte abordará a configuração inicial do inversor e seus detalhes de ligação com o motor.

4.1 Configuração do inversor

Existem quatro camadas de configuração para o inversor:

- Configuração do padrão
- Modo de operação
- Modo de interface com o usuário
- Configuração da macro

4.1.1 Configuração do padrão

Esta configuração refere-se ao padrão (americano ou europeu) de fornecimento de energia AC para o inversor. Normalmente, ele já vem pré-configurado quando sai da fábrica, de acordo com o continente em que será vendido. A diferença entre esses padrões se dá na frequência de alimentação: $60Hz$ para o padrão americano e $50Hz$ para o Europeu. Para mudar o padrão de alimentação para o americano (nosso caso), execute os passos da seção 4.1.5.

4.1.2 Modo de operação

Após a escolha da configuração do padrão, o próximo passo é informar ao inversor o tipo de motor que ele vai trabalhar (síncrono, indução, etc.). Os modos de operação são os seguintes: *open loop*, *closed-loop vector* e *closed-loop servo*. Os dois primeiros referem-se a motores de indução e o último a motores autosíncronos. O modo de operação do CITOSINA é *closed-loop servo*, representado pela sigla **SV** ou **CL**.

4.1.3 Configuração da macro

As *macros* tem a finalidade de ajustar vários parâmetros do inversor de uma vez, para cada tipo de aplicação desejada, reduzindo a quantidade de modificações necessárias para o *drive*. Existem 9 *macros* diferentes, numeradas de 0 a 8, de acordo com o objetivo do projeto (ver tabela 4). Para a aplicação em questão (controle da antena), será utilizada a *macro 0* que é de propósito geral. A seção 4.1.5 explica como configurar o inversor para trabalhar nesse modo.

Tipo	Aplicação
Macro 0	General purpose
Macro 1	Easy mode
Macro 2	Motorized potentiometer
Macro 3	Preset frequencies/speeds
Macro 4	Torque control
Macro 5	PID control
Macro 6	Axis-limit control
Macro 7	Brake control
Macro 8	Digital lock shaft orientation

Tabela 4: Tipos de macro

4.1.4 Modo de interface com o usuário

A interface com o usuário pode ser feita através dos terminais do *drive* ou através do teclado (*keypad*) localizado na sua parte frontal. As informações de configuração do inversor estão divididas em menus, que vão de 0 a 19. Por exemplo, o parâmetro 0.42 faz parte do menu 0 e armazena o número de pólos do motor (item 42).

4.1.5 Sequência de configuração

Os passos iniciais para configuração do inversor devem ser os seguintes:

1. **Modificação do modo de operação e frequência:** Acesse o menu 0.00 e digite a opção 1254 (padrão americano). Em seguida escolha o parâmetro 0.48 e selecione o modo **SerVO**). A modificação do modo de operação será possível apenas, após a inserção do valor 1254 no parâmetro 0.00.

2. **Seleção da entrada de controle:** a seleção da referência deve ser realizada pelo parâmetro 0.05. Com a escolha da opção 1, informamos ao *drive* que a tensão de controle será aplicada nos seus terminais 5 e 6.

3. **Programação dos parâmetros do motor:** As informações do motor (número de pólos, velocidade, etc.) devem ser inseridas nos seus respectivos parâmetros do inversor que variam de 0.42 a 0.47. A tabela 5 apresenta os valores que devem ser inseridos para cada item.

Parâmetro	Nome	Valor	Unidade
0.47	Motor rated frequency	0	Hz
0.46	Motor rated current	2.5	A
0.45	Motor rated full load RPM	0	RPM
0.44	Motor rated voltage	0	V
0.43	Motor power factor	1	-
0.42	Motor number of poles	6	-
0.41	PWM switching frequency	3	Khz

Tabela 5: Parâmetros do motor

4. **Autocalibração ou Faseamento (*autotune*)**: Para este procedimento o motor deve estar sem carga e o contato EXTERNAL TRIP/DRIVE ENABLE fechado. Acesse o parâmetro 0.40 do inversor e mude seu valor para 1. O motor girará lentamente 360 graus, parando logo em seguida. O 0.40 voltará para 0 automaticamente. Acesse o parâmetro 0.00 e insira o valor 1000 para salvar as alterações.

4.2 Freio do motor elevação

O motor de elevação apresenta um freio responsável por manter a antena fixa numa determinada posição. Quando seus terminais são alimentados (24V e 0.67A), o eixo do motor passa a girar livremente, permitindo seu controle pelo inversor. Um circuito auxiliar (ver figura 7), controlado pelo computador, permite o comando desse freio através da condição do transistor T_1. Quando este encontra-se saturado ($V_i = 5V$), a bobina do freio é alimentada, liberando o eixo, e quando cortado ($V_i = 0V$) trava-o. Caso o inversor receba um comando para girar o motor nessa situção de eixo travado, uma proteção do UMV4301 será ativada, cortando a alimentação do motor. Logo em seguida, um código de erro será apresentado em seu visor. Para fazer cortar ou saturar T_1, foi utilizado uma das saídas digitais da placa AD, responsável pelo nível lógico ALTO (5V) ou BAIXO (0V) em sua entrada.

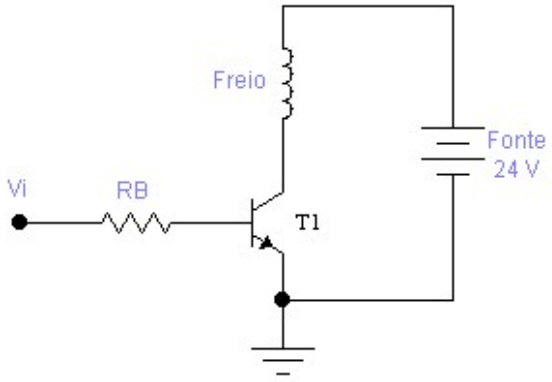

Figura 7: Circuito auxiliar para o freio do motor de elevação

Em virtude da corrente nominal do freio ser 0.67A, escolheu-se o transistor TIP31 para o cicuito auxiliar 7, com h_{FE} de aproximadamente 78 (valor medido). Com base nessas informações, a resistência R_B foi calculada como descrito abaixo.
Para a condição de saturação:

$$I_B > \frac{I_C}{h_{FE}}$$

$$I_B > \frac{0.67}{78}$$

$$I_B > 9.85mA$$

Valor adotado $\longrightarrow I_B = 11mA$

$$V_i - R_B \cdot 11 \cdot 10^{-3} - 0.7 = 0$$

$$R_B = \frac{5 - 0.7}{11 \cdot 10^{-3}} = 390\Omega$$

5 Algoritmos de controle

O controlador inicial proposto para o sistema de posição foi o PID, em virtude de sua simplicidade, ampla faixa de operação e bons resultados práticos. Ele é um dos controladores mais usados na área industrial, com um grande índice de aceitação por parte dos operadores. Apesar de existirem outros controladores semelhantes, como o avanço-atraso, o PID domina o cenário, pois necessita de menos parâmetros a se determinar durante o seu projeto. Porém, para funcionar adequadamente, é necessário o conhecimento total da planta, o que não reflete a situação do *Citosina*. Todos os parânetros da planta são desconhecidos, sendo necessário um processo de modelagem para o sistema motor+inversor. A equação 2 apresenta a planta do controlador PID, onde K_p, K_d e K_i referem-se aos ganhos da parte proporcional, derivativa e integrativa, respectivamente.

$$G_c(s) = K_p + K_d s + \frac{K_i}{s} \tag{2}$$

Mesmo não conhecendo a planta, um controlador PI foi introduzido com parâmetros, K_p e K_i, encontrados de forma intuitiva. As premissas utilizadas nesse projeto baseiam-se em observações práticas, coletadas na implementação de vários outros sistemas de controle. As regras para escolha dos valores foram as seguintes:

Para K_p
Grande → Resposta mais rápida (pode gerar grandes oscilações).
Pequeno → Resposta mais lenta (as oscilações são menores ou sem oscilações).

Para K_i
Grande → Resposta mais rápida (pode gerar oscilações significativas). O erro em regime permanente (E.R) zera mais rapidamente.
Pequeno → Resposta mais lenta (com pequenas oscilações ou sem oscilações). E.R=0 mais lentamente.

Para K_d
Grande → Resposta mais lenta e possivelmente sem oscilações.
Pequeno → Resposta mais rápida e possivelmente com oscilações.

Atualmente, a inclusão da parte derivativa na lei de controle está sendo estudada. A proposta final é realizar um estudo profundo da planta e obter o controlador PID devidamente adequado ao sistema. Uma segunda esfera de interesse corresponde a implementação de um controle adaptativo por modelo de referência (MRAC) ou por estrutura variável (VS-MRAC). Este tipo de controlador é utilizado quando os parâmetros da planta são desconhecidos ou conhecidos com incerteza. A robustez do sistema é outro ponto importante. Nesses algoritmos, MRAC e VS-MRAC, a dinâmica não modelada será considerada bem como eventos que tornem o sistema instável (perda de robustez).

Pelos testes realizados com o controlador PI, para uma referência constante, obteve-se bons resultados. O erro em regime foi para zero e o desempenho no transitório foi satisfatório, porém um pouco oscilatório, em dentrimento da falta da parte deivativa. O

algoritmo implementado no sistema *Citosina* encontra-se listado no apêndice A. Apenas a posição de elevação foi testada, pois o motor do azimute estava em manutenção.

6 Conclusão e considerações finais

O controle de posição da antena foi realizado apenas para a posição de elevação, pois alguns problemas surgiram na malha do azimute que impossibilitaram os testes. Vale salientar também que o algoritmo de controle implementado não englobou a rotina de rastreio de satélites, mas apenas a convergência da saída da planta para uma referência constante. Dessa forma é possível posicionar a antena numa posição desejada, dentro do sistema de referência escolhido.

Toda a cabeação e estrutura física da estação SACI foi aproveitada, apesar da falta de uma documentação adequada. Devido a esta barreira, o trabalho de pesquisa tornou-se extremamente prático, dependendo de testes em bancada para a solução de alguns problemas. O principal deles foi sanado através de um contato com o representante dos motores e inversores (*Leroy Somer*) no Brasil. Sem isso, a parte de acionamento estaria bem atrasada em relação as demais.

A malha de controle de posição está quase concluída, dependendo apenas dos trabalhos de manutenção para o eixo de azimute. O sistema de posicionamento e acionamento foram analisados por completo, e implementados parcialmente na estação *Citosina*. O desempenho do sistema operacional Linux foi excelente, apesar da não inclusão de *patches* no *kernel* para trabalhar com tarefas em tempo real.

As próximas atividades do projeto *Citosina* estão descritas abaixo:

1. Solução de problemas com o motor azimute e a sua leitura de posição (verificar a possibilidade de alguns dos componentes não está funcionando corretamente).

2. Bateria de testes com a posição Azimute.

3. Estudo do RTAI (Real-Time Application Interface) para trabalhar em tempo real no linux. Apesar dos resultados atuais serem satisfatórios apenas com o *Comedi* (drives e bibliotecas da placa A/D), será necessária uma otimização do sistema para torná-lo mais estável. Hoje sem o uso do RTAI, o linux não garante exclusividade total ao software de controle da antena, o que prejudica o seu desempenho.

4. Introdução de novas técnicas de controle da planta (PID, controle adaptativo direto ou indireto, etc.)

5. Modelagem da planta ou o uso de técnicas de controle que estimem seus parâmetros.

6. Construção de um algoritmo de controle para rastreio de satélites.

Outras atividades menores também serão executadas, porém possuem prioridade menor, e podem ser facilmente adequadas as necessidades. Esta lista corresponde a sequência de ações futuras na sua ordem de execução, que pode ser mudada em virtude de sugestões e de fatos inesperados.

A Listagem do algoritmo PI

Foi desenvolvido um programa de computador, na linguagem C, cuja listagem é apresentada neste apêndice.

```
#include <stdio.h>   /* para o printf */
#include <comedilib.h>
#include <sys/time.h>
#include <math.h>
#include <stdlib.h>
#include <pthread.h>
//Objetivos:
//Este programa tem por finalidade a execucao do algortimo de controle

int subdevA = 1; // Saida analogica
int subdevD = 2;//Saidas e entradas digitais
int chanF = 0;//Canal DIO0 52 DGND 18
int chanCLK = 2;//Canal de CLOCK DIO2 49 DGND 15
int chanLIB = 3;//Canal para liberar a leitura DIO3 47 DGND 13
int chanCLR = 4;//Canal do CLEAR DIO4 19 DGND 53 LSB
int chanL1 = 5;//Canal de leitura da posicao DIO5 51 DGND 9
int chanL2 = 6;//Canal do leitura da posicao DIO6 16 DGND 50
int chan = 0;
int range = 0;
int aref = AREF_GROUND;
double AZ,ELE,ELEA,AZA;
void pos();
void escritaELE(double volts);
void escritaAZ(double volts);
double outv=0.0; //sinal de controle
double ref=0.0; //Sinal de referencia
double kp=0.05; //Constante do controlador
double ki=25;
double I=0.0;
double erro=0.0;
comedi_t *it;


void *control(void *unused)
{
   int sat=0;
   double Isat=0.3, volts=0.0, erroa=0.0, Ta=0.1;

   while(1)
   {
    usleep(100000);
    pos();
    //printf("ELE=%lf AZ=%lf",ELE,AZ);
```

```
    erro=ref-ELE;
    if(sat==0){
       I=I+(Ta/ki)*erroa*(-1);
    }
    if(I>Isat) I=Isat;
    if(I<-Isat)I=-Isat;
    outv=kp*erro*(-1)+I;
    if(outv>10.0){
       outv=10.0;
       sat=1;
    }
    else if(outv<(-10.0)){
       outv=-10.0;
       sat=1;
    }
    else sat=0;
    escritaELE(outv);
    erroa=erro;
   }
   return NULL;
}
void *print(void *unused1)
{
  while(1){
     usleep(1000000);
     printf("ELE=%lf AZ=%lf outv=%lf erro=%lf I=%lf\n",ELEA,AZ,outv,erro,I);
 }
}

main()
{
  pthread_t thread_id, thread_id1;

  //abertura do device
  it=comedi_open("/dev/comedi0");

  //Configuracao dos canais para a funcao pos()
  comedi_dio_config(it,subdevD,chanCLK,COMEDI_OUTPUT);
  comedi_dio_config(it,subdevD,chanCLR,COMEDI_OUTPUT); //clear
  comedi_dio_config(it,subdevD,chanL1,COMEDI_INPUT); //Leitura de 16 bits
  comedi_dio_config(it,subdevD,chanL2,COMEDI_INPUT);//Leitura de 4 bits
  comedi_dio_config(it,subdevD,chanLIB,COMEDI_OUTPUT);

  //Configuracao do canal para a funcao escritaELE
  comedi_dio_config(it,subdevD,chanF,COMEDI_OUTPUT);
  printf("Qual a posicao?");
  scanf("%lf", &ref);
  pthread_create(&thread_id,NULL,&control,NULL);
```

```
  pthread_create(&thread_id1,NULL,&print,NULL);

  while(1);
  //printf("ELE=%lf AZ=%lf\n",ELE,AZ);
  if(comedi_close(it)) perror("Erro ao fechar");
  return 0;
}


void pos()
{
  lsampl_t data;
  unsigned int *bit, bits[20];
  int i;

  //Escrita nos canais
  comedi_dio_write(it,subdevD,chanLIB,0); //libera bits para leirura

  //Limpa os flip-flops
  comedi_dio_write(it,subdevD,chanCLR,1);
  usleep(1);
  comedi_dio_write(it,subdevD,chanCLR,0);
  usleep(1);
  comedi_dio_write(it,subdevD,chanCLR,1);
  AZ=0;
  ELE=0;

  //Le os canais do azimute
  bit=malloc(sizeof(unsigned int));
  for(i=0;i<16;i++)
  {
     comedi_dio_read(it,subdevD,chanL1,bit);
     bits[i]=*bit;
     if (i<10) ELE=ELE+bits[i]*pow(2,i);
     else AZ=AZ+bits[i]*pow(2,i-10);

     if(i<4)
     {
        comedi_dio_read(it,subdevD,chanL2,bit);
        bits[i+16]=*bit;
AZ=AZ+bits[i+16]*pow(2,i+6);
     }
     comedi_dio_write(it,subdevD,chanCLK,0);
     usleep(1);
     comedi_dio_write(it,subdevD,chanCLK,1);
     usleep(1);
     comedi_dio_write(it,subdevD,chanCLK,0);
     usleep(1);
```

```
  }
  free(bit);
  comedi_dio_write(it,subdevD,chanLIB,1); //volta ao estado inicial

  AZ=AZ*360/(pow(2,10));
  ELE=ELE*360/(pow(2,10));
  if(ELE>240) ELE=ELE-360;
  ELE=0.6*ELE+60;
  ELEA=ELE;
  AZA=AZ;
}


void escritaELE(double volts)
{
  lsampl_t data, maxdata;
  comedi_range *range1;

  maxdata=comedi_get_maxdata(it, subdevA, chan);
  range1=comedi_get_range(it,subdevA,chan,range);
  data=comedi_from_phys(volts, range1, maxdata);
  if(volts==0.0){
  comedi_dio_write(it, subdevD,chanF,0);//coloca o freio
  }
  else{
  comedi_dio_write(it,subdevD,chanF,1);//retira o freio
  }
  comedi_data_write(it,subdevA,chan,range,aref,data);
}

void escritaAZ(double volts)
{
  lsampl_t data, maxdata;
  comedi_range *range1;

  maxdata=comedi_get_maxdata(it, subdevA, chan);
  range1=comedi_get_range(it,subdevA,chan,range);
  data=comedi_from_phys(volts, range1, maxdata);
  comedi_data_write(it,subdevA,chan,range,aref,data);
}
```

Referências

[1] Tocci, Ronald J., Widmer, Neal S., Sistemas Digitais *Princípios e Aplicações*, LTC, 2000.

[2] Dorf, Richard C., Bishop, Robert H., *Sistemas de controle modernos*, LTC, oitava edição, 2001.

[3] Ogata, K., *Sistemas de controle moderno*, Prentice-Hall do Brasil, quarta edição, 1998.

[4] Araújo, Aldayr D. de, *Notas de aula da disciplina "Sistemas de Controle II"*, UFRN, 2005.

[5] Boylestad, Robert L., Nashelsky, L., *Dispositivos eletrônicos*, Pearson/Pratice Hall, 2004.

[6] Nise, Norman S., *Engenharia de Sistemas de Controle*, terceira edição, LTC, 2002.

[7] Satry, S., *Adaptive control: stability, convergence and robustness*, Prentice Hall, 1989.

[8] Ioannou, Petros A., Sun, J., *Robust adaptive control*, 1996, Prentice Hall.

[9] Schleef, D., Hess, Frank M., Bruyninckx, H., *The Control and Measurement Device Interface handbook*, 2002. Obtido em: http://www.comedi.org/doc/index.html.

[10] Leroy Somer SMV UM Moteur autosynchrones catalogue technique.

[11] Leroy Somer UMV 4301 Variateur de vitesse pour moteur asynchrones avec et sans retour et pour moteurs autosynchrones installation et maintenance.

[12] Leroy Somer UMV 4301 Variateur de vitesse pour moteur asynchrones avec et sans retour et pour moteurs autosynchrones - Paramétrage et synoptiques.

[13] GESPAC - Dual Parallel Interface Modulo (2x16 I/O): GESPIA 2A & 2AW.

[14] RDC-19220 & RD-19230 - Series Converters - Applications Manual MN-19220XX.

[15] SYNCHRO/RESOLVER CONVERSION HANDBOOK.