



# DECODIFICADOR SBCDA/ARGOS EM PC INTEGRADO AO SINDA

Arthur Bezerra Dantas Saraiva (UFRN, Bolsista PIBIC/CNPq)
E-mail: arthur_saraivasp@hotmail.com

José Marcelo Lima Duarte (INPE, Orientador)
E-mail: jmarcelold@gmail.com

Junho 2016

# DECODIFICADOR SBCDA/ARGOS EM PC INTEGRADO AO SINDA

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA

(PIBIC/CNPq/INPE)

Arthur Bezerra Dantas Saraiva (UFRN, Bolsista PIBIC/CNPq)
E-mail: arthur_saraivasp@hotmail.com

José Marcelo Lima Duarte (INPE, Orientador)
E-mail: jmarcelold@gmail.com

Junho 2016

# Agradecimentos

Agradeço a todas as pessoas que me ajudaram neste trabalha, direta ou indiretamente.

Ao Instituto Nacional de Pesquisas Espaciais (INPE) – Pela oportunidade de estudos e utilização de suas instalações.

Ao meu orientador Dr. José Marcelo Lima Duarte pela disponibilidade constate para tirar duvidas e pelo apoio durante o trabalho.

A minha família por sempre me dar apoio e incentivo ao estudo.

Aos professores do curso de Engenharia elétrica da Universidade Federal do Rio Grande do Norte, por todo o conhecimento passado.

Aos Engenheiros Pedro Silva de Aquino e Lúcio dos Santos Jotha pela participação e contribuição para o desenvolvimento do trabalho.

# Resumo

Este trabalho tem como objetivo a especificação do computador de bordo do Subsistema de Controle e Rastreio (SCR) da Estação Multimissão de Natal (EMMN). O computador de bordo será implementado na plataforma arduino. As efemérides, coordenadas de apontamento para a antena, serão geradas através do software gratuito Orbitron, que será executado em um PC, e serão transferidas para o Arduino via um cabo USB. O computador de bordo terá como funções principais: ler as efemérides e o sinal dos sensores de posição da antena; executar o algoritmo de controle de posição e gerar os sinais para os atuadores; executar os telecomandos de ligar e desligar o sistema; prover telemetrias ao computador (azimute, elevação, referência atual azimute e elevação); desligar o sistema em caso de eminência de colisão com o fim de curso da antena; controlar LEDs do painel da gaveta de *pilotage*. O algoritmo de controle está funcional, porém ainda temos alguns problemas com o hardware do sistema, o que ainda impossibilita o ligamento do sistema.

# Abstract

This article aims to onboard computer specification Subsystem Control and Tracking (SCR) Christmas Multimission Station (EMMN). The onboard computer will be implemented on the Arduino platform. The ephemeris, pointing coordinates for the antenna, Orbitron will be generated using the free software that will run on a PC, and will be transferred to the Arduino via a USB cable. The onboard computer will have as main functions: to read the ephemeris and the signal from the antenna position sensors; perform the position control algorithm and generate the signals to the actuators; perform remote controls on and off the system; provide telemetry to the computer (azimuth, elevation, current reference azimuth and elevation); shut down the system in case of collision eminence with the end of the antenna course; control LEDs pilotage drawer panel. The control algorithm is functional, but we still have some problems with the system hardware, which also prevents the system ligament.

## Lista de Figuras

# Sumário

## 1. Contextualização

Foram desenvolvidos no INPE, dois Satélites de Aplicação Científica (SACI - Satelite Avançado de Comunicações Interdisciplinares) o qual não chegaram a operar em órbita devido à falhas: o SACI-1 e o SACI-2. O primeiro chegou a ser lançado em órbita, porém houve uma falta de transmissão de qualquer sinal à base devido a problemas de equipamentos de bordo. Já o segundo satélite, nem se quer chegou a sua órbita e por isso foi necessário abordar o lançamento e acionar a autodestruição do mesmo.

A Estação Multimissão de Natal (EMM-Natal) consiste em um projeto de desenvolvimento tecnológico, para reaproveitar a estação francesa destinada aos Satélites de Aplicação Científica. A EMM-Natal será uma estação para múltiplas missões, onde está sendo projetada para agregar as funcionalidades de centro de controle e missão e estação solo de recepção.

O trabalho em questão é a continuidade do que outros bolsistas do INPE vieram fazendo para deixar a EMM-Natal funcional. O controle atual do sistema é feito através de um software para PC desenvolvido em plataforma Linux, no qual o PC é o centro de controle e processamento. O programa desenvolvido era capaz de realizar rastreio dos satélites com sucesso, porém devido a pouca interface gráfica e relativa dificuldade de operação foi decido desenvolver um novo sistema em plataforma Arduino em conjunto com a utilização do software Orbitron. A ideia é deixar o Arduino como um computador de bordo do sistema, necessitando apenas de um cabo USB para comunicação com o PC para receber as Efemérides.

Este trabalho está dividido em duas partes, na Seção 2 é dado uma visão geral sobre o hardware do sistema (maiores detalhes estão disponíveis em [3]), e na Seção 3 o software de controle do sistema na plataforma Arduino é detalhado. O algoritmo desenvolvido até o momento está disponível no apêndice deste trabalho.

## 2. Introdução ao Orbitron

O Orbitron é um software gratuito desenvolvido por Sebastian Stoff para fins de observação radio amador. É utilizado tanto por profissionais do tempo, usuários de comunicação por satélite, astrônomos e até mesmo astrólogos [4]. Foi escolhido utilizar o Orbitron para o sistema de rastreio devido à relativa facilidade de utilização do software, além de ele possuir uma interface agradável ao usuário, permitindo a percepção visual das passagens dos satélites. A Figura 1 ilustra o software em execução.

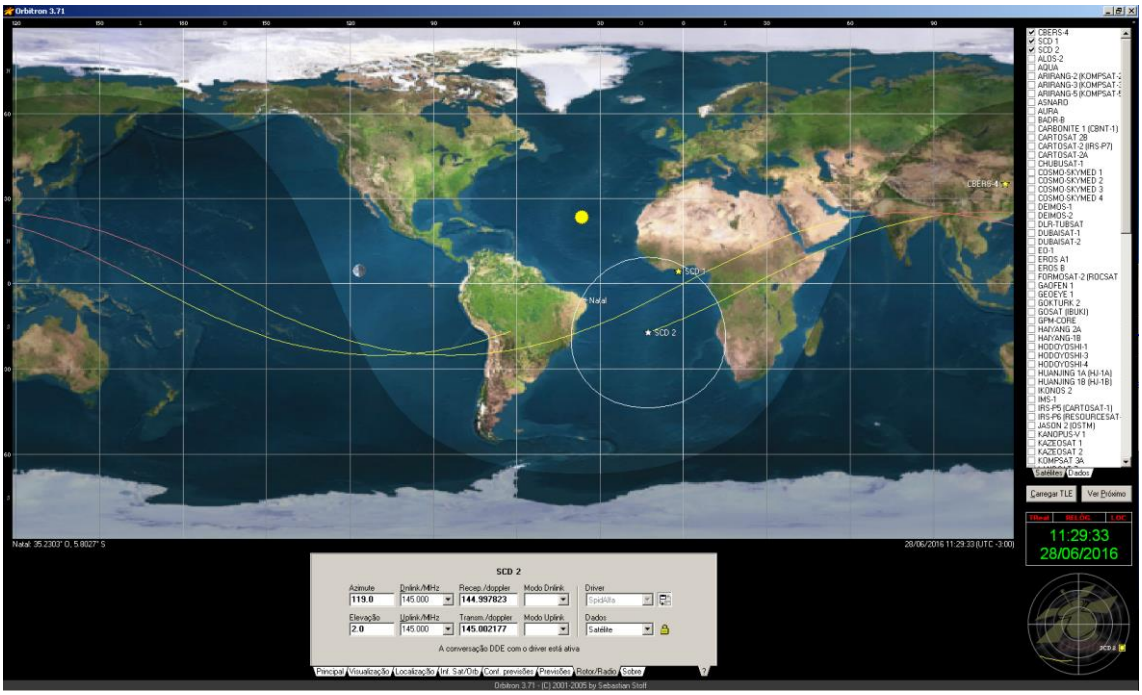


**Figura 1 - Painel frontal do Orbitron.**

Como pode-se observar na imagem acima, a interface gráfica do Orbitron é bastante agradável para visualização das passagens dos satélites. Neste software, para começar o rastreio define-se a posição do receptor na terra, na aba “Localização”, e a partir daí o software gera as efemérides do satélite selecionado. A Figura 2 mostra o painel principal do programa.


**Figura 2 - Painel principal do Orbitron.**

Neste painel é possível carregar um arquivo de TLEs (Two Line Elements), que é um formato para a descrição dos dados orbitais de satélites, e após isso selecionar quais satélites se deseja ser rastrear. Os satélites disponíveis para rastreio ficam listados ao lado direito do mapa mundi. Para selecionar o satélite a ser rastreado deve-se o marcar nessa lista. Logo após, o satélite irá aparecer no mapa. Com isso a trajetória do satélite fica visível através de uma linha amarela – nos pontos em que o satélite estiver recebendo luz solar – ou vermelha, nos pontos em que não estiver recebendo luz solar. Neste mesmo painel é possível fazer simulações para um satélite selecionado, tanto para passagens futuras quanto para passagens que já ocorreram definindo o modo entre "Tempo Real" ou "Simulação". Caso seja escolhida simulação, será disponibilizado ao lado as opções do passo de tempo para avanço (caso passagem futura) ou regresso (caso passagem passada).


**Figura 3 - Painel Rotor/Radio do Orbitron.**

A Figura 3 ilustra o painel Rotor/Radio. Escolhido o satélite a ser rastreado, neste painel o Orbitron gera a cada segundo informações de: Azimute e elevação do satélite em relação à referência escolhida na terra; Frequencia de Uplink e Downlink do sinal considerando o efeito Doppler. Porém, o Orbitron não disponibiliza essas informações para gerar um arquivo de efemérides ou para envia-la serialmente. Para isso é utilizado o driver SpidAlfa. Este driver está disponível em [4] e tem a função de receber as informações do Orbitron e envia-la serialmente para a porta serial escolhida. Essa informação será recebida pelo Arduino e será utilizada como posição de referência para a malha de controle do sistema.

Maiores informações sobre o funcionamento do Orbitron estão disponíveis na opção “Ajuda” no painel principal.

## 2.1. Driver SpidAlfa

Este driver tem como função principal receber as informações do orbitron e envia-las para o Arduino. A Figura 4 ilustra o painel principal do driver em funcionamento.

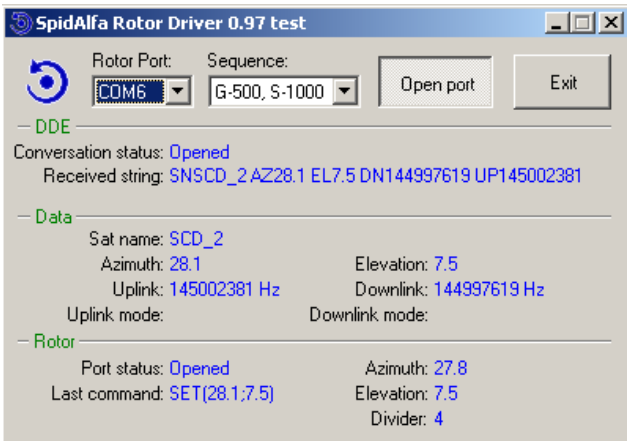


**Figura 4 - Painel SpidAlfa.**

As informações desse driver são divididas em três partes principais: DDE, Data e Rotor. Na sessão DDE, é visualizado o status da conversação com o Orbitron, e o

*string* de dados recebido dele. O *status* “Opened” será visualizado nessa seção quando o driver for aberto. Isso significa que a comunicação entre o Orbitron e o SpidAlfa foi estabelecida, e então é exibido o string de informações recebidas na subseção “Received string”, como mostrado na Figura 4. Na seção Data as informações contidas no “Received string” são apresentadas separadamente. As informações visíveis são: nome do satélite, Azimute, Elevação, e taxa de Downlink e Uplink.

Na ultima seção é mostrado o *status* de comunicação do driver com o rotor, que para este sistema de rastreio será o Arduino. Quando a comunicação estiver estabelecida será exibido “Opened” em “Port status”. O driver possui três comandos principais: SET, STATUS e STOP. O comando SET envia as informações de referência pela porta serial. A informação é enviada no formado de um string de 13 caracteres, contendo o azimute e elevação do satélite selecionado. No comando STATUS, o driver deve receber do Arduino o status atual de elevação e azimute do rotor. O comando STOP será executado quando a opção “Open port” for desmarcada pelo usuário, fazendo com que a comunicação entre o driver e o arduino seja desfeita. Devido a isso será exibido “Closed” em “Port Status”.

Maiores detalhes sobre o funcionamento do driver estão disponíveis no protocolo de comunicação**,** disponível em [5].

## 3. Introdução ao Arduino

O Arduino é uma placa composta de um microcontrolador e vários circuitos de entrada/saída de dados. Possui uma linguagem de programação baseada em C/C++ e pode ser programado utilizando sua IDE, sendo necessário apenas um cabo USB para comunicação com o computador. Ele foi escolhido para ser o computador de bordo do sistema de rastreio devido a sua alta funcionalidade e sua relativa facilidade de programação. Sua IDE já vem com várias funções para comunicação digital, analógica e serial prontas para uso.

Para este projeto foi escolhido utilizar o Arduino Mega, devido a sua boa capacidade de processamento e devido a ele possuir um número suficiente de portas digitais para a realização deste projeto. Este Arduino possui: microcontrolador ATmega2560; 54 portas digitais, das quais 15 podem ser usadas como PWM; 16 entradas analógicas. A tensão de saída máxima de suas portas é de 5V.

A Figura 5 ilustra a placa do Arduino Mega.

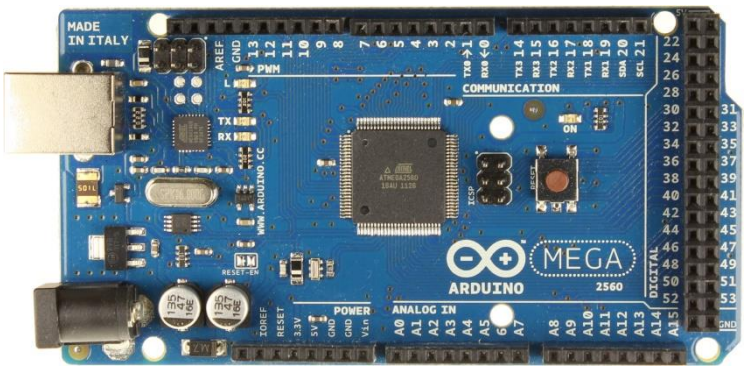


**Figura 5 – Arduino Mega.**

## 4. Introdução ao sistema

Todos os componentes do sistema de rastreio foram herdados da antiga estação de rastreio SACI [3]. O sistema é composto por quatro módulos: O módulo de potência, o módulo posicionador, o módulo de comando e o módulo de controle (Arduino e o software Orbitron). Cada um desses módulos será explicado brevemente nas seções seguintes. A Figura 6 Ilustra como é feita a comunicação entre os módulos do sistema.

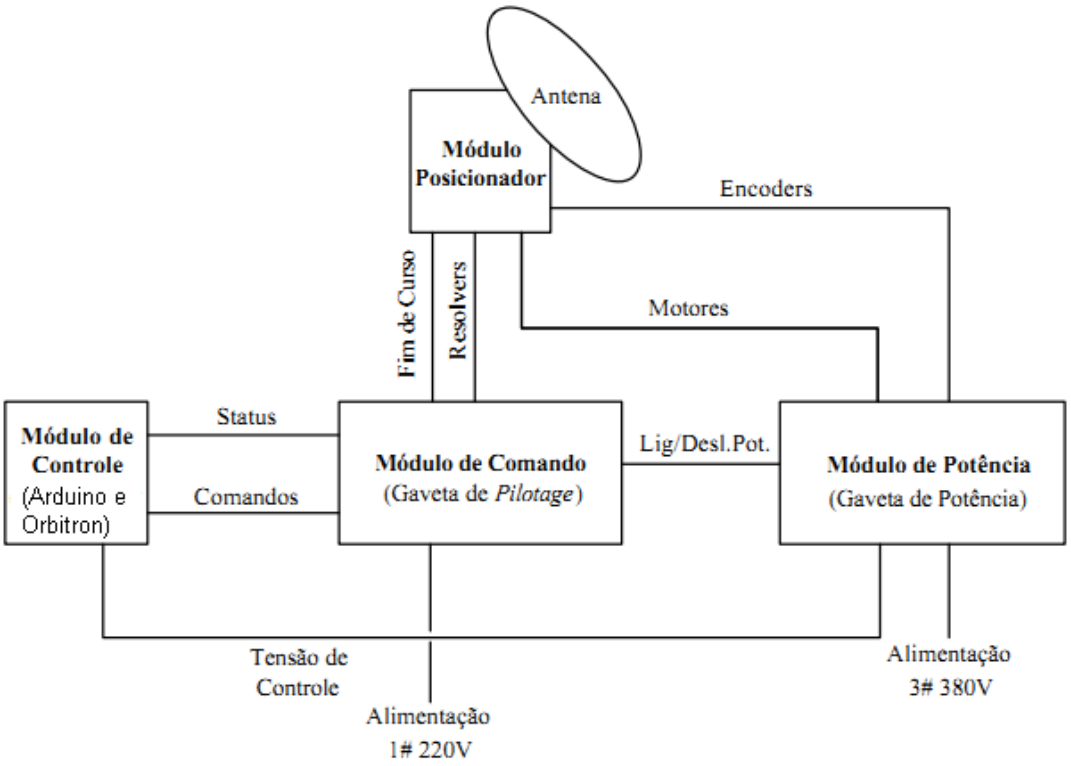


**Figura 6 - Estrutura de hardware da estação. Fonte:[3](Adaptado).**

## 4.1. A teoria de controle

O sistema deve posicionar a antena corretamente em seus eixos azimute e elevação a cada intervalo de amostragem. Para isso, como a referência para este sistema pode ser aproximada por uma rampa, foi utilizado um controlador do tipo Proporcional Integrativo (PI) foi utilizado. A Figura 7 ilustra o diagrama de blocos do sistema.

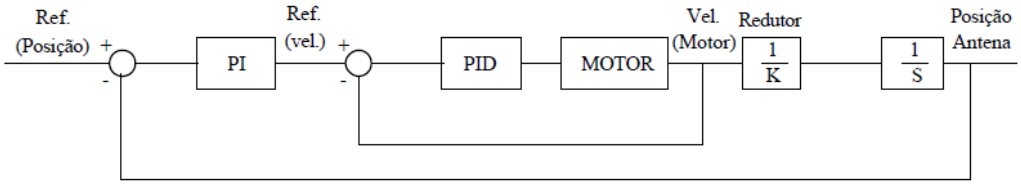


**Figura 7 - Diagrama de blocos do sistema. Fonte:[3]**

Como podemos observar na imagem acima, além da malha de controle posição, existe a malha de controle da velocidade dos motores de cada eixo. Este controle é realizado pelos próprios inversores do módulo de potência através de um controlador PID (Proporcional Integrativo e Derivativo). Como o controle de velocidade é realizado de forma bem mais rápida que o controle de posição, para fins de simplificação o controle de velocidade pode ser desconsiderado.

Um controlador digital do tipo PI, funcionando no módulo de controle, comparaa posição atual da antena com a posição desejada (referência). A diferença entre as duas grandezas é o erro de posição, que é enviado ao controlador PI e este gera o sinal de controle para os inversores. O sinal de controle é um sinal analógico que pode variar de -10 V a +10 V e corresponde a velocidade de rotação dos motores. O período de amostragem e as constantes do controlador PI digital foram mantidas as mesmas do projeto anterior e correspondem a:

$$\mathbf{kpELE = kpAZ = 0,6}$$

$$\mathbf{kiELE = kiAZ = 0,05}$$

## 4.2. Módulo posicionador

O módulo posicionador permite que a antena se mova de forma independente em cada eixo, azimute e elevação. Cada eixo é movido por conjunto motor mais redutor coaxial. Os motores são do tipo autossíncronos com encoders oriundos de fábrica. Os encoders são equipamentos que medem velocidade angular instantânea e são utilizados na malha de controle de velocidade. Para cada eixo, existe um dispositivo chamado resolver que mede a posição angular instantânea da antena. Essa posição é um sinal analógico e será convertido em um sinal digital codificado de 16 bits no módulo de comando. Porém para este sistema serão utilizados apenas os 12 bits mais significativos, pois com isso já se consegue uma precisão considerada satisfatória para o sistema.

Como proteção, o sistema possui: dispositivos de fim de curso em cada eixo, que ao serem acionados desativam o sistema; proteção via software, que consiste na verificação constante do status do sistema para se algum evento indesejado ocorrer desligar o sistema; e proteção mecânica, que são borrachas nos fins de cursos que amortecem o impacto caso o fim de curso seja atingido. A Figura 8 ilustra a antena da estação.

**Figura 8 - Antena da estação. Fonte: [2]**

## 4.3. Módulo de potência

O módulo de potência, ou gaveta de potência, é onde ficam os dois inversores, um para cada eixo, modelo UMV 4301 da *Leroy Somer*. Estes inversores são responsáveis pelo controle de velocidade dos motores da antena do sistema. Para isso recebem um sinal de controle proveniente do Arduino, que após conversão varia entre -10 V e +10 V, onde 10 V significa velocidade nominal (3000 rpm), -10 V velocidade nominal em sentido contrario e 0 V significa parado. O sistema de controle interno do inversor garante a convergência da velocidade do motor com a velocidade de referência. A Figura 9 ilustra a gaveta de potência e seus inversores.


**Figura 9 - Gaveta de potência. Fonte [2].**

## 4.4. Módulo de comando

O módulo de comando ou gaveta de comando (*tiroir pilotage*) é onde estão todos os circuitos de comando e fontes de alimentação para alimentação dos inversores e segurança do sistema. Em sua parte frontal também estão LEDs para visualização de eventos, como: fim de curso atingido, rastreio, potência ativa, entre outros. Nela também existe um autotransformador (230V/48V) que alimenta o sistema de aquecimento (herdado da antiga estação francesa) para não ocorrer congelamento dos motores do módulo posicionador. Apesar de aqui no Brasil não ser necessário devido ao clima quente naturalmente, o sistema de aquecimento foi mantido apenas por compatibilidade com o antigo sistema [3].

A comunicação entre o módulo de comando e o módulo de controle é feita via conectores que saem de sua parte traseira identificados por uma letra e um número.

O sinal do resolver referente a posição atual da antena chega até o módulo de comando como sinal analógico. Nele existe a placa "resolver para digital", a qual faz a conversão do sinal dos resolvers para um sinal digital de 16 bits. Dessa forma, seriam necessários 32 canais digitais para poder receber as posições de azimute e elevação do sistema. Com a finalidade de diminuir a quantidade de canais necessários, foi desenvolvida a placa INPE-01, que conecta os canais referentes a azimute e elevação em paralelo, permitindo a seleção de qual posição se deseja ler através dos conectores 19B e 25A da placa SOTEREM 2266-1, como um circuito multiplexador. Quando essas entradas se encontram e nível lógico alto, o processo de conversão da placa Resolver para Digital continua em ação. Porém, quando um desses pinos estiver em nível lógico baixo o processo de conversão é parado e é possível ser feita a leitura.

Alguns dos principais circuitos de comando serão mostrados a seguir. A Figura 10 abaixo apresenta o esquema geral de alimentação do circuito de comando.

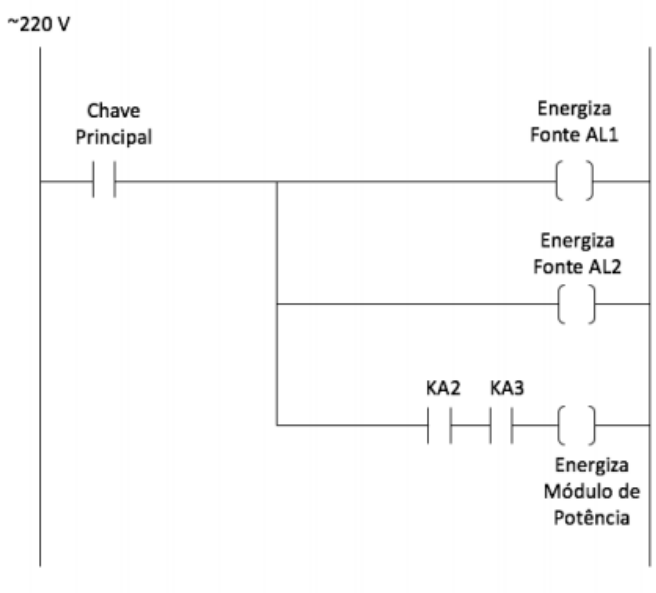


**Figura 10 - Esquema geral de alimentação do circuito de comando. Fonte: [2].**

Pela figura acima, podemos ver que ao ser ligada a chave principal, as fontes de alimentação AL1 (12 V) e AL2 (24 V) são alimentadas. Contudo, para o módulo de potência ser energizado as bobinas KA2 e KA3 precisam ser alimentadas. Na Figura 11 abaixo é mostrado o esquema simplificado do circuito alimentado pela fonte AL1.

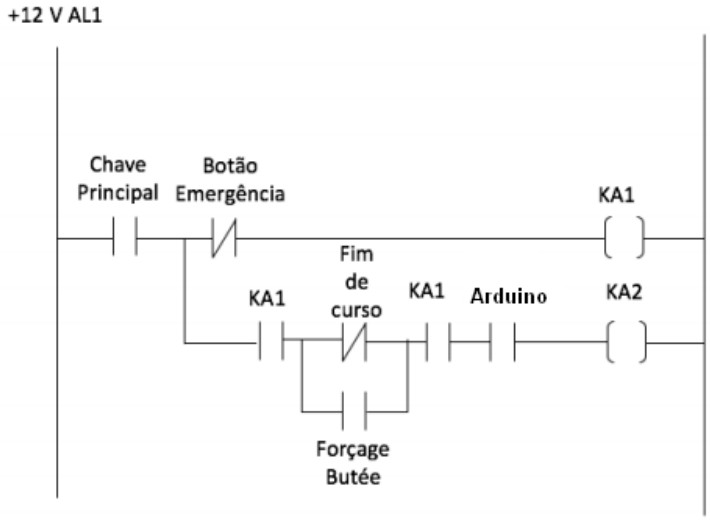


**Figura 11 - Circuito alimentado pela fonte AL1. Fonte: [2] (Adaptado).**

Ao ser energizada, a fonte AL1 energiza a bobina KA1. Com isso os contatores de KA1 normalmente aberto são fechados no circuito de alimentação de KA2. Percebemos então que para energizar KA2 é necessário que o Arduino feche este circuito, que será ativo quando alguma passagem estiver disponível.

Ainda nesse circuito vemos o contato normalmente fechado dos fins de curso da antena em paralelo com o contato do botão "Forçage Butée". A função desse botão é poder remover a antena do fim de curso utilizando o *joystick* do módulo de comando caso ela atinja o fim de curso.

A Figura 12 abaixo mostra o esquema simplificado do circuito alimentado pela fonte AL2.

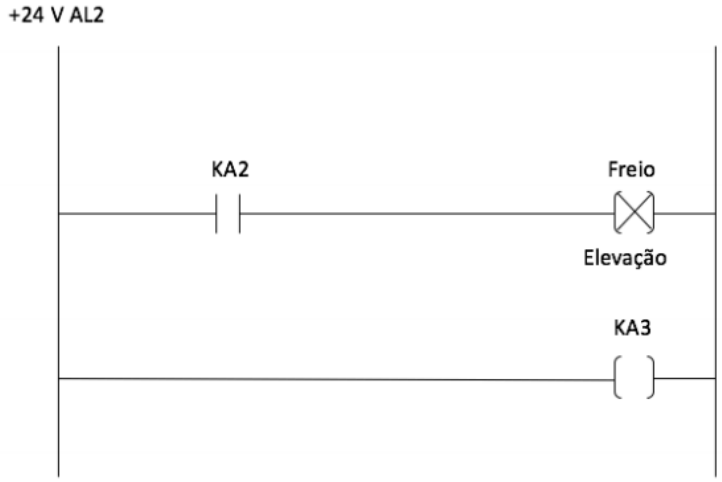


**Figura 12 - Circuito alimentado pela fonte AL2. Fonte:[2].**

Ao analisar a figura acima percebe-se que ao ser energizada a fonte AL2 energiza a bobina KA3, que nesse caso fecha seu contato que alimenta o módulo de potência, na Figura X. Com isso, concluímos que o acionamento do sistema dependerá apenas do comando do Arduino, que energizará KA2. Nesta figura, vemos também que a bobina KA2 ao ser energizada desativa o freio que o motor de elevação possui. A função principal desse freio é impedir que a antena atinja o sensor de fim de curso ao ser desligada.

## 4.5. Módulo de controle

Este ultimo módulo é composto pelo computador e pelo Arduino. O computador irá enviar serialmente as informações de referência para o Arduino a cada 1 segundo através do software Orbitron. O Arduino ira receber essa referência e gerar o sinal de controle para os inversores do módulo de potência, permitindo assim o sucesso no rastreio. Ele deve evitar o fim de curso da antena e verificar constantemente se algum evento indesejado ocorreu, como: acionamento de fim de curso, defeito nos inversores, botão de emergência acionado, entre outros. Caso algum evento indesejado ocorra o sistema será desligado até a normalização da situação. Além disso, ele fará o controle dos leds do painel de controle. A tabela abaixo mostra a configuração dos canais digitais utilizados do Arduino.

**Tabela 1 – Configuração de canais digitais do Arduino.**

| PINO | Descrição | Configuração |
|---|---|---|
| 22 | Bit 00 (LSB) | Entrada Digital |
| 23 | Bit 01 | Entrada Digital |
| 24 | Bit 02 | Entrada Digital |
| 25 | Bit 03 | Entrada Digital |
| 26 | Bit 04 | Entrada Digital |
| 27 | Bit 05 | Entrada Digital |
| 28 | Bit 06 | Entrada Digital |
| 29 | Bit 07 | Entrada Digital |
| 30 | Bit 08 | Entrada Digital |
| 31 | Bit 09 | Entrada Digital |
| 32 | Bit 10 | Entrada Digital |
| 33 | Bit 11 (MSB) | Entrada Digital |
| 34 | LED Butées | Saída Digital |
| 35 | LED Poursuite | Saída Digital |
| 36 | LED Arrêt d’urgence | Saída Digital |

| | | |
|---|---|---|
| 37 | LED Puisance | Saída Digital |
| 38 | LED Défaut | Saída Digital |
| 39 | LED Manuel | Saída Digital |
| 40 | LED Survie | Saída Digital |
| 41 | Defeito azimute | Entrada Digital |
| 42 | Defeito elevação | Entrada Digital |
| 43 | Status KA1 | Entrada Digital |
| 44 | Status KA2 | Entrada Digital |
| 45 | Status KA4 | Entrada Digital |
| 46 | Libera leitura azimute | Saída Digital |
| 47 | Libera leitura elevação | Saída Digital |
| 48 | Liga gaveta de potência | Saída Digital |

## 5. Algoritmo de controle e rastreio em Arduino

Para funcionar, o Arduino deve ser carregado com o algoritmo de controle e conectado a um PC com o software Orbitron e o driver SpidAlfa em funcionamento. O diagrama de blocos abaixo na Figura 13 ilustra um esquema geral do algoritmo de controle.

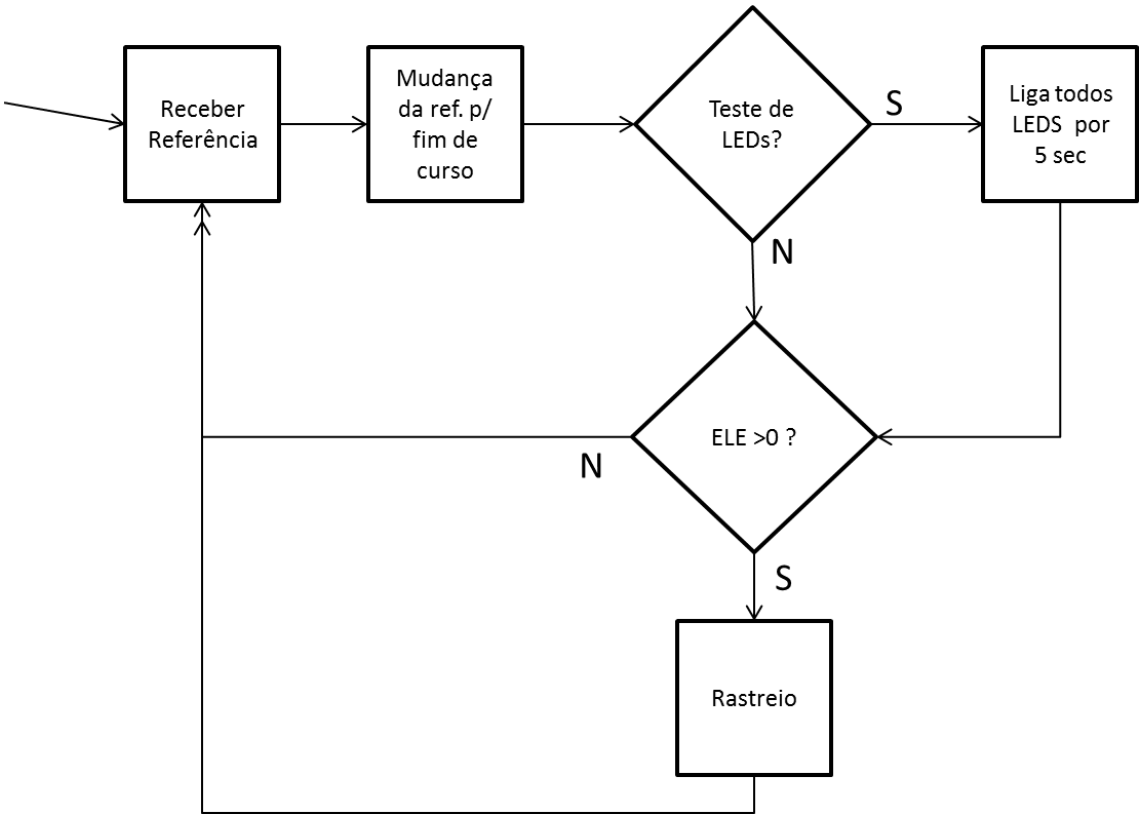


**Figura 13 – Diagrama geral do sistema de controle.**

O algoritmo de rastreio do sistema começa na função loop( ) do arduino, onde ele recebe a referência de azimute e elevação do Orbitron. A cada 1 segundo esse par de referências é enviado para o Arduino pela porta serial. Após a recepção, é feita a mudança desses ângulos para 0º no fim de curso da antena, pois isso facilita a parte do algoritmo de controle. O Arduino permanecerá recebendo as referências do Orbitron sem realizar o rastreio até que a posição de elevação seja maior que zero. Esse critério de inicio de rastreio foi escolhido devido ao driver SpidAlfa não transmitir referências

de ângulos menores que zero. Logo, quando a referência de elevação de um satélite selecionado for maior que zero, significa que o satélite entrou na zona de visibilidade da antena, e com isso podemos iniciar o rastreio. Durante a fase inicial de recepção de referências – e apenas nela - é possível solicitar o teste de LEDs. Quando solicitado, todos os LEDs serão acesos por um período de 5 segundos, e logo após desligados, retornando ao processo de recepção e checagem das passagens. O diagrama de blocos abaixo na Figura 14 detalha o bloco de rastreio do algoritmo a baixo.

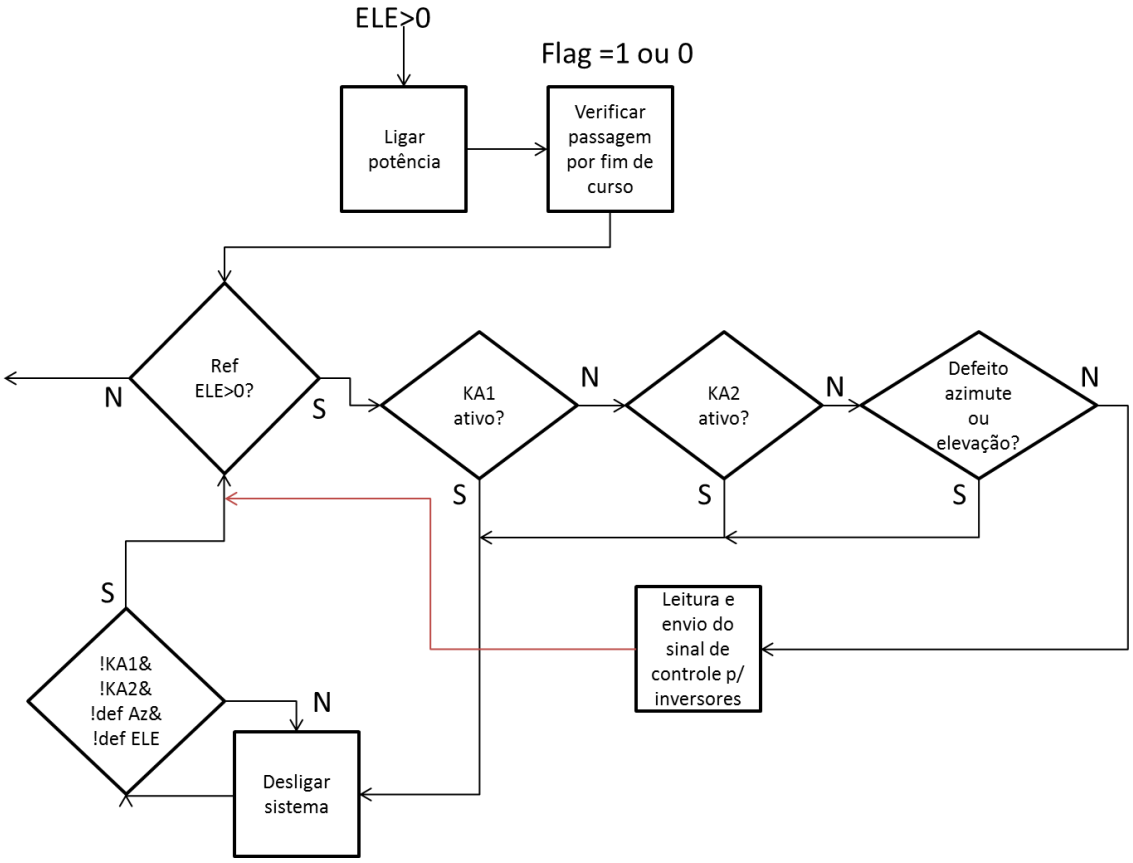


**Figura 14 - Diagrama de blocos do bloco de rastreio.**

Após a condição para inicio de rastreio ser satisfeita, a potência do sistema é ligada, e a partir daí o sistema irá verificar se a passagem do satélite ira passar pelo fim de curso da antena. O critério escolhido para saber se tal evento ira ocorrer é feito pela análise de duas posições de azimute consecutivas. Se a diferença entre o segundo valor e o primeiro valor de azimute for positivo, significa que a referência de azimute está

aumentando, e logo poderá passar pelo fim de curso. Logo, caso isso ocorra, é gerada uma flag que informa que o fim de curso será atingido. Para evita-lo, utilizamos o segundo quadrante da elevação, e o terceiro e quarto quadrante do eixo azimute. As equações utilizadas para isso estão mostradas abaixo.

$$refELE = 180 - refELE;$$
$$refAZ = refAZ + 180;$$
$$\text{Se } refAZ > 360 \rightarrow refAZ = refAZ - 360$$

A fim de trabalhar apenas com ângulos variando de 0º a 360º, o sistema verifica se o valor de azimute ultrapassou 360º. Caso tenha ultrapassado é feita subtração de 360º desse valor de referência. Após o ajuste o algoritmo verifica se algum evento indesejado ocorreu. Os eventos são analisados pelo status das bobinas KA1, KA2 e pelos pinos de status de defeito dos inversores de elevação e azimute. KA1 será acionado caso o botão de emergência seja ativado. KA2 será ativado quando: KA1 for ativado, fim de curso ativado ou o controle mandar desligar o sistema (ainda não implementado). Os pinos de defeito do azimute e elevação de cada inversor estarão ativos quando seus respectivos inversores estiverem com defeito. Qualquer um desses eventos levara o sistema para o estado “Desligar”, onde é desligada a potência e o sistema ficará “preso” neste bloco até que todas as condições acima estejam normalizadas.

Caso nenhum dos eventos indesejados ocorram, o sistema segue recebendo uma nova referência e novamente fazendo o ajuste para o fim de curso da antena. O diagrama a baixo na Figura 15 ilustra o bloco de leitura e envio do sinal de controle.

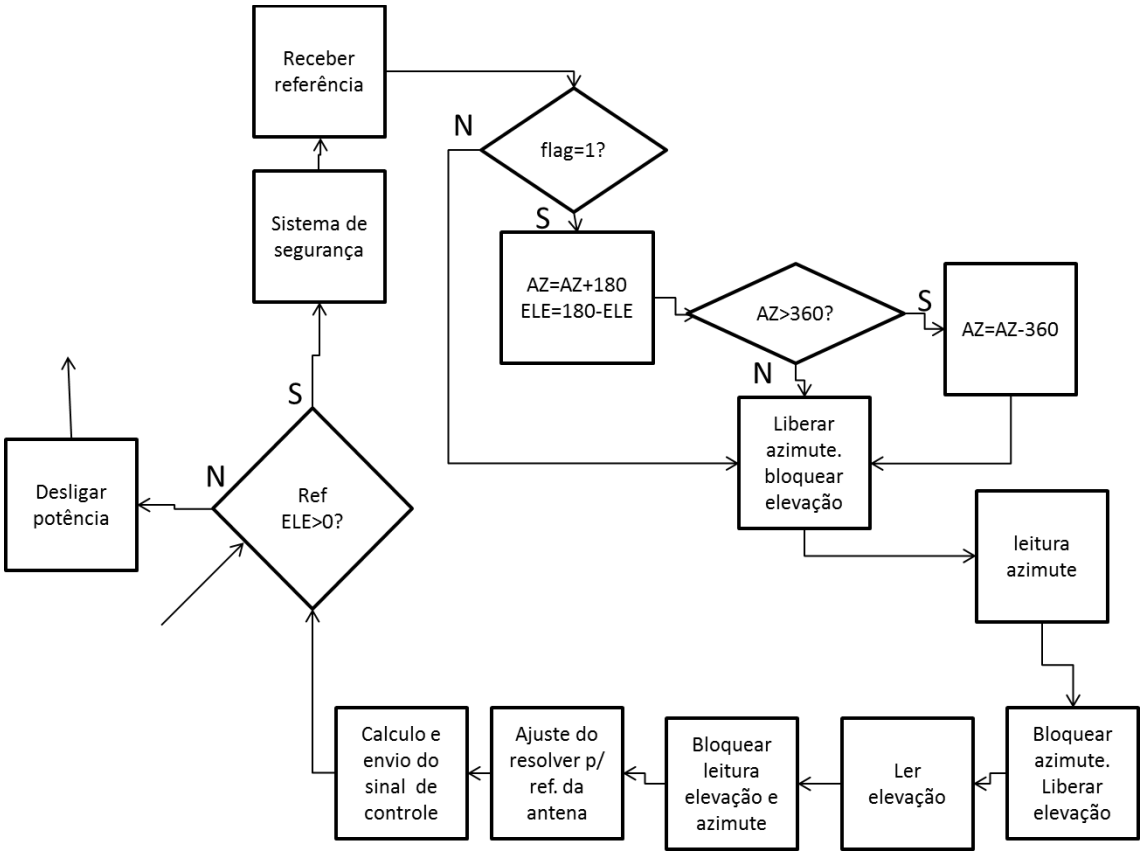


**Figura 15 - Diagrama de blocos do bloco "Leitura e envio do sinal de controle p/ inversores".**

Em seguida é feita a leitura das posições atuais da antena para cada eixo, primeiro liberando e lendo a leitura do azimute e bloqueando a leitura da elevação. Depois é feito o bloqueio do azimute e liberado e lido a leitura da elevação. Ao final, ambas as leituras são bloqueadas e é feita a decodificação das leituras dos resolvers. Para uma leitura de 12 bits, as equações a baixo realizam a decodificação.

$$\text{angResAZ} = \text{resAZ} * \frac{360}{2^{12}}$$

$$\text{angResELE} = \begin{cases} \text{resELE} * \frac{360}{2^{12}} & \text{se resELE} * \frac{360}{2^{12}} < 230 \\ \text{resELE} * \frac{360}{2^{12}} - 360, & \text{se não} \end{cases}$$

O ajuste no ângulo do resolver de elevação é necessário devido a uma descontinuidade que existe no grau 0, que passaria de 0º para 360º. A Figura 16 abaixo ilustra a comparação dos valores extremos de azimute e elevação para os resolvers e para a antena.

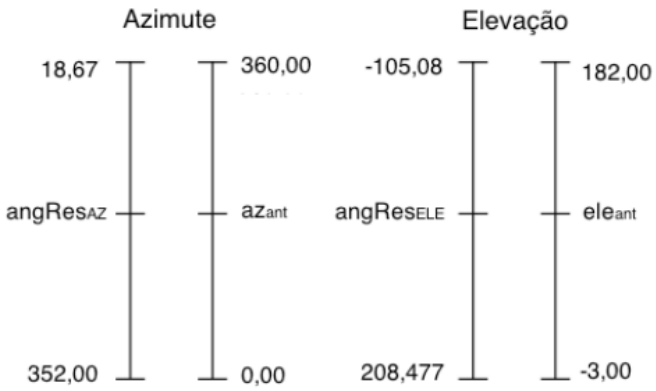


**Figura 16 - Conversão entre escalas. Fonte:[2]**

Após a decodificação do valor lido no resolver é feita a conversão desses valores para a referência da antena no fim de curso. As equações de conversão estão mostradas a baixo e foram obtidas em utilizando conversão entre escalas com os valores extremos para cada escala, como na figura acima.

$$AZant = -1.08*angResAZ+380,16$$
$$ELEant = -0.59*angResELE+120$$

Com a posição atual lida no resolver e a posição de referência fornecida pelo Orbitron, é calculado o erro da antena, e a partir o algoritmo de controle calcula-se o sinal de controle para os inversores. Após esse calculo e envio do sinal, o sistema vai checar se a elevação ainda é maior que zero, e em seguida se algum evento indesejado ocorreu. Caso não tenha ocorrido, irá lê mais uma posição de referência e refazer todo o algoritmo de rastreio até que a referência de elevação seja menor que zero. Quando isso acontecer a potência do sistema será desligado e o sistema ficará em estado de espera

pela próxima passagem (elevação maior que zero) desse mesmo satélite, ou outro selecionado manualmente no Orbitron.

Por questão de simplificação, não foi colocado o controle dos LEDs da gaveta de comando nos diagramas esquemáticos acima. Porém, o controle de LEDs foi implementado em código. Este código de rastreio está disponível no Apêndice.

## 6. Conclusões e Recomendações

O trabalho realizado até aqui tem como objetivo deixar a estação funcional, e a partir dai procurar aperfeiçoamentos para o sistema de controle. O algoritmo desenvolvido ainda não foi testado na prática devido a questões de hardware. Está sendo feita manutenção em alguns componentes da estação devido ao longo tempo sem uso. Além disso, ainda estamos esperando alguns componentes que solicitamos ao INPE chegar, como a placa que enviará o sinal de controle do Arduino para os inversores e o próprio Arduino Mega da estação.

O desenvolvimento do sistema de controle com o Arduino pode ser considerado satisfatório. Em relativo pouco tempo conseguimos preparar o algoritmo de rastreio. As funções de comunicação do Arduino são bastante simples de usar, o que reduziu bastante o trabalho.

Para implementações futuras pretendemos implementar um software que, dentre os satélites selecionados, rastreie automaticamente o satélite que estiver disponível e, caso ocorra mais de uma passagem ao mesmo tempo, permita escolher o critério de escolha para o rastreio. Além disso, tornar o sistema mais comunicativo com o usuário, permitindo por exemplo a exibição do status do sistema em um display digital, dar ao usuário opção de controle manual, entre outras opções.

**Referências Bibliográficas**

[1] Language Reference Arduino: https://www.arduino.cc/en/Reference/HomePage. Acesso em: 30/06/2016.

[2] GONÇALVEZ, T. R. Sistema de Controle para Rastreio de Satélite Utilizando Plataforma Labview, Universidade Federal do Rio Grande do Norte, Natal. 2015.

[3] QUEIROZ, K. I. P. M. Descrição Funcional do Subsistema de Controle e Rastreio da Estação Multimissão de Natal (EMMN), Instituto Nacional de Pesquisas Espaciais, Natal. 2006.

[4] Orbitron - Satellite Tracking System: http://www.stoff.pl/. Acesso em: 30/06/2016.

[5] Protocolo de comunicação do SpidAlfa: http://ryeng.name/blog/3. Acesso em: 30/06/2016

# Apêndice

```
//Posições de referência
float refAZ,refELE;
//posições atuais
float AZ,ELE,AZres,ELEres;
//Leitura da posição
int bits[12];
//Status bobinas e inversores
int KA1=0,KA2=0,KA4=0,defAZ=0,defELE=0;

//variaveis de auxilio
int i,j,k;

//Variaveis do algoritmo de controle
//Azimute
float erroAZ,erroaAZ=0,outvAZ=0,IsatAZ=0.15,Ta=1,kpAZ=0.6,kiAZ=0.05,IELE=0;
int satAZ=0;
//Elevação
float erroELE,erroaELE=0,IsatELE=0.15,outvELE=0,kpELE=0.6,kiELE=0.05,IAZ=0;
int satELE=0;

void setup()
{
//Iniciar porta serial
Serial.begin(9600);

//Configuração de pinos de leitura de posição atual Azimute e Elevação (Conector S8 -
Gaveta de pilotage)
pinMode(22,INPUT); //LSB
pinMode(23,INPUT);
pinMode(24,INPUT);
pinMode(25,INPUT);
pinMode(26,INPUT);
pinMode(27,INPUT);
pinMode(28,INPUT);
pinMode(29,INPUT);
pinMode(30,INPUT);
pinMode(31,INPUT);
pinMode(32,INPUT);
pinMode(33,INPUT); //MSB

//Configuração dos pinos dos LEDs (Conector S7 - Gaveta de pilotage)
pinMode(34,OUTPUT); //Butéés - Fim de curso
pinMode(35,OUTPUT); //Poursuite - Rastreio
pinMode(36,OUTPUT); //Arrêt d'urgence - Parada de emergência
```

```
pinMode(37,OUTPUT); //Puisance - Gaveta de potência
pinMode(38,OUTPUT); //Defáut - Defeito
pinMode(39,OUTPUT); //Manuel - Modo manual
pinMode(40,OUTPUT); //Survie -

//Defeito azimute e elevação (Conector S7 - Gaveta de pilotage)
pinMode(41,INPUT); //Azimute
pinMode(42,INPUT); //Elevação

//Status bobinas KA1, KA2 e KA4 (Conector S7 - Gaveta de pilotage)
pinMode(43,INPUT); //KA1 -> Botão de emergência acionado (Status: Pino)
pinMode(44,INPUT); //KA2 -> Fim de curso, Módulo de controle e KA1
pinMode(45,INPUT); //KA4 -> Teste de LEDs

//Liberar leitura azimute e elevação (Conector S8 - Gaveta de pilotage)
pinMode(46,OUTPUT); //azimute
pinMode(47,OUTPUT); //elevação

//Ligar gaveta de potencia (Conector S7 - Gaveta de pilotage)
pinMode(48,OUTPUT);

}



void loop()
{
//Receber referência

//A leitura da referência está implementada, porém
//não foi colocada neste trabalho




// Mudança de referência para 0º no fim de curso
refAZ=refAZ-18;
refELE= refELE+3;

if(refAZ < 0){
  refAZ=refAZ+360;
  }

//Teste de LEDs

if(digitalRead(45)==HIGH){ //Teste de LEDs solicitado
```

```
 digitalWrite(34,HIGH); //Butéés
 digitalWrite(35,HIGH); //Poursuite
 digitalWrite(36,HIGH); //Arrêt d'urgence
 digitalWrite(37,HIGH); //Puisance
 digitalWrite(38,HIGH); //Defáut
 digitalWrite(39,HIGH); //Manuel
 digitalWrite(40,HIGH); //Survie

 delay(10000);

 digitalWrite(34,LOW); //Butéés
 digitalWrite(35,LOW); //Poursuite
 digitalWrite(36,LOW); //Arrêt d'urgence
 digitalWrite(37,LOW); //Puisance
 digitalWrite(38,LOW); //Defáut
 digitalWrite(39,LOW); //Manuel
 digitalWrite(40,LOW); //Survie

  }

//Condição para ligar o sistema

if(refELE>0){
//Ligar sistema e posicionamento
digitalWrite(48,HIGH); //liga modulo de potencia
digitalWrite(37,HIGH); //Liga LED potência
delay(2000);


//RASTREIO ----------------------------------------------------------------------------------
-

while(refELE>0){
digitalWrite(35,HIGH); //Liga LED Rastreio

//Status KA1 (botão de emergencia)
if(digitalRead(43)==HIGH){
  KA1=1;
  digitalWrite(36,HIGH); //Liga LED parada de emergência
  goto Desligar;
    }

//Status KA2 (KA1,fim de curso ou controle)
if(digitalRead(44)==HIGH){
  KA2=1;
```

```
  digitalWrite(34,HIGH); //Liga LED fim de curso
  goto Desligar;
    }

//Defeito Azimute
if(digitalRead(41)==HIGH){
  defAZ=1;
  digitalWrite(38,HIGH); //Liga LED defeito
  goto Desligar;
     }
//Defeito Elevação
 if(digitalRead(42)==HIGH){
  defELE=1;
    digitalWrite(38,HIGH); //Liga LED defeito
  goto Desligar;
    }

//leitura da posição de referência
//A leitura da referência está implementada, porém
//não foi colocada neste trabalho


//leitura da posição atual e conversão para referência da antena (fim de curso)
//leitura do azimute
AZres=0;
ELEres=0;

digitalWrite(46,LOW); //libera leitura Azimute
digitalWrite(47,HIGH); //bloqueia leitura elevação

j=22;
for(i=0;i<12;i++){ //loop para leitura de posição azimute


    if(digitalRead(j)==HIGH){
      bits[i]=1;
           }
    else{
      bits[i]=0;
      }
      AZres=AZres+bits[i]*pow(2,i);
      j=j+1;
  }

digitalWrite(46,HIGH); //bloqueia leitura azimute
digitalWrite(47,LOW); //libera leitura elevação
```

```
j=22;
for(i=0;i<12;i++){ //loop para leitura de posição elevação

   if(digitalRead(j)==HIGH){
     bits[i]=1;
     }
   else{
     bits[i]=0;
     }
     ELEres=ELEres+bits[i]*pow(2,i);
     j=j+1;
  }

digitalWrite(46,HIGH); //bloqueia leitura azimute
digitalWrite(47,HIGH); //bloqueia leitura elevação

AZres=AZres*360/4096;
ELEres=ELEres*360/4096;
if(ELEres>240){ ELEres=ELEres-360;}

//calculo da posição atual da antena a partir da posição dos resolvers
AZ=-1.08*AZres+380.16;
ELE=-0.59*ELEres+120;

//algoritmo de controle p/azimute

erroAZ=refAZ-AZ;
   if(satAZ==0){
     IAZ=IAZ+(Ta*kiAZ)*erroaAZ;
   }
   //Tecnica anti-reset windup
   if(IAZ>IsatAZ)IAZ=IsatAZ;
   if(IAZ<-IsatAZ)IAZ=-IsatAZ;

    outvAZ=kpAZ*erroAZ+IAZ;
   //Limitacao do sinal de controle
   if(outvAZ>10.0){
     outvAZ=10.0;
     satAZ=1; //saturacao da parte integrativa
   }
   else if(outvAZ<-10.0){
     outvAZ=-10.0;
     satAZ=1; //saturacão da parte integrativa
   }
   else satAZ=0;
```

```
outvAZ=outvAZ*(-1); //FALTA MANDAR SINAL DE SAIDA PARA INVERSOR
erroaAZ=erroAZ;

//algoritmo de controle p/ elevação

erroELE=refELE-ELE;
if(satELE==0){
IELE=IELE+(Ta*kpELE)*erroaELE;
}
//Tecnica anti-reset windup
   if(IELE>IsatELE) IELE=IsatELE;
   if(IELE<-IsatELE)IELE=-IsatELE;

outvELE=kpELE*erroELE+IELE;
//limitacao do sinal de controle p/ elevação
   if(outvELE>10.0){
     outvELE=10.0;
     satELE=1; //saturacao da parte integrativa
   }
   else if(outvELE<(-10.0)){
     outvELE=-10.0;
     satELE=1; //saturacao da parte integrativa
   }
   else satELE=0;
            //FALTA MANDAR SINAL DE SAIDA PARA INVERSOR
erroaELE=erroELE;


}

//FIM de rastreio - Desligar potência
digitalWrite(48,LOW); //Desliga modulo de potencia
digitalWrite(37,LOW); //Desliga LED potência

  }

Desligar:
while(KA1==1 || KA2==1 || defAZ==1 || defELE==1){
  digitalWrite(48,LOW); //Desliga modulo de potencia
  outvELE=0;
  outvAZ=0;


  digitalWrite(35,LOW); //Desliga LED rastreio
  digitalWrite(37,LOW); //Desliga LED potência
```

```
    }
}
```