

FACULDADE DE TECNOLOGIA DE SÃO PAULO
ELETRÔNICA INDUSTRIAL

FELIPE MONARI DA SILVA
LUIZ ANTONIO VILIOTI
LUCAS AUGUSTO FERREIRA

SISTEMA MICROCONTROLADO DE MOVIMENTAÇÃO DE ANTENAS DE
RECEPÇÃO DE SINAIS VIA SATÉLITE

SÃO PAULO, FEVEREIRO DE 2018

FELIPE MONARI DA SILVA
LUIZ ANTONIO VILIOTI
LUCAS AUGUSTO FERREIRA

SISTEMA MICROCONTROLADO DE MOVIMENTAÇÃO DE ANTENAS DE
RECEPÇÃO DE SINAIS VIA SATÉLITE

Monografia apresentada ao curso de
Eletrônica Industrial da FATEC-SP, como
requisito parcial para obtenção do título de
tecnólogo, orientado pelo Prof. Dr.
Leonardo Frois.

SÃO PAULO, FEVEREIRO DE 2018

SISTEMA MICROCONTROLADO DE MOVIMENTAÇÃO DE ANTENAS DE
RECEPÇÃO DE SINAIS VIA SATÉLITE

FELIPE MONARI DA SILVA
LUIZ ANTONIO VILIOTI
LUCAS AUGUSTO FERREIRA

Monografia defendida e aprovada, em 15/02/2018, pela banca examinadora:

Professor Doutor Leonardo Frois Hernandez

Orientador e Professor do Curso de Eletrônica Industrial da FATEC-SP

Professor Mestre Ricardo Cardoso Rangel

Professor do Curso de Eletrônica Industrial da FATEC-SP

Mestre Alisson Rodolfo Leite

Tecnico do Instituto de Física da USP

DEDICATÓRIAS

Dedico este trabalho a minha mãe, Carmen, e meu pai, Bernardo, por todo o apoio e paciência, e a equipe de manutenção eletrônica da Sky Brasil, pela oportunidade de crescimento pessoal, profissional e acadêmico.

Felipe Monari da Silva

Dedico este trabalho primeiramente a Deus, por me ter feito alcançar essa meta, e também a meus familiares, em especial minha esposa Fabiana e minha filha Rebeca pela paciência, e a todos os professores do curso.

Luiz Antonio Vilioti

Dedico este trabalho a Deus, em primeiro lugar, por me conduzir e me sustentar no curso; aos meus pais, por me assistirem nesses anos, e a minha namorada Marina, por me impulsionar a terminá-lo.

Lucas Augusto Ferreira

LISTA DE FIGURAS

Figura 1 - Apontamento de uma antena. ⁴	12
Figura 2 - Cadeia de operações de uma empresa de transmissão de televisão DTH.	15
Figura 3 – Um pátio de antenas de recepção.....	16
Figura 4 - LNBS horizontal e vertical conectados a um alimentador.....	17
Figura 5 - Sistemas de apontamento mecânicos.	18
Figura 6 - Antenas A e B.	19
Figura 7 - Sistemas de posicionamento motorizados.	19
Figura 8 - Botões de controle (A), que acionam conjuntos de contadoras e relés (B e C)	20
Figura 9 - Pinagem do PIC18F4550. ⁸	22
Figura 10 – Registrador ADCON0 do PIC18F4550. ⁸	23
Figura 11 – Registrador ADCON1 do PIC18F4550. ⁸	24
Figura 12 – Registrador ADCON2 do PIC18F4550. ⁸	25
Figura 13 – Registrador RCON do PIC18F4550. ⁸	25
Figura 14 - Registrador INTCON do PIC18F4550. ⁸	26
Figura 15 – Registrador INTCON2 do PIC18F4550. ⁸	26
Figura 16 – Registrador INTCON3 do PIC18F4550. ⁸	27
Figura 17 - Simulação do circuito projetado.	30
Figura 18 - Simulação da memória EEPROM.....	31
Figura 19 - Montagem do sistema para testes em bancada.	31
Figura 20 - Layout da placa de circuito impresso no ARES.	32
Figura 21 - Fotolito (fora de escala, com indicação de componentes e não espelhado) das placas do teclado (A) e PIC (B).	32
Figura 22 - Montagem das placas	33
Figura 23 – Estrutura da antena.....	38
Figura 24 - Exterior da caixa do sistema projetado.	39
Figura 25 - Interior da caixa do sistema projetado.	39
Figura 26 - Diagrama de blocos do sistema de controle.	42

Figura 27 - Diagrama de blocos da função main()	44
Figura 28 - Diagrama de blocos da função main (cont.).....	45
Figura 29 - Tela 0	45
Figura 30 - Tela de confirmação de zeramento.....	46
Figura 31 - Tela de efetuação do zeramento.....	46
Figura 32 - Tela 1.....	46
Figura 33 - Tela 5.....	47
Figura 34 - Tela 2.....	47
Figura 35 - Tela 3 (A) e tela 4 (B).....	48
Figura 36 - Circuito do LCD.....	49
Figura 37 - LCD Custom Char Generator.....	49
Figura 38 – Circuito do Teclado Analógico.....	50
Figura 39 - Diagrama de blocos da função de leitura do teclado.	54
Figura 40 - Diagrama de blocos da função de movimentação manual.....	57
Figura 41 - Tela de digitação.....	58
Figura 42 - Tela indicando a movimentação.....	59
Figura 43 - Diagrama de blocos da função de digitação de novo valor.....	59
Figura 44 - Diagrama de blocos da função de seleção de posições pré-definidas....	62
Figura 45 - Tela 6.....	63
Figura 46 – Tela indicando a movimentação.....	63
Figura 47 - Diagrama de blocos da função de gravação.....	64
Figura 48 - Tela 7.....	65
Figura 49 - Tela 8 (A) e tela de digitação (B).....	65
Figura 50 - Tela 9 (A) e tela de digitação (B).....	65
Figura 51 - Tela 10.....	65

LISTA DE TABELAS

Tabela 1 – Tabela descrevendo as possíveis combinações dos bits 3-0 do registrador ADCON1. ⁸	24
Tabela 2 - Configuração de micropassos. ¹¹	34
Tabela 3 - Configuração de corrente. ¹¹	35
Tabela 4 - Valores calculados para o teclado analógico.	51
Tabela 5 - Valores medidos na protoboard para o teclado analógico.	52
Tabela 6 - Relação de botões e resistores.	53
Tabela 7 - Custos do circuito de controle	68

RESUMO

No centro de transmissão de uma empresa de televisão por satélite existe a necessidade de apontar uma antena para diversas posições orbitais com a maior velocidade e praticidade possível. Nesse caso específico, baseado também em conhecimentos profissionais adquiridos em um estágio feito através da FATEC-SP, o sistema de posicionamento da antena em questão é motorizado, porém sem controle remoto para um operador. Neste trabalho será desenvolvido um circuito eletrônico baseado no microcontrolador PIC18F4550 capaz de controlar uma antena motorizada. Foi desenvolvido, além do circuito, uma programação para controlar o sistema através do microcontrolador e uma maquete para demonstração de seu funcionamento. Esse sistema deve permitir ao usuário movimentar uma antena através de teclas indicando direção, digitando um valor para cada posição e selecionando uma posição de *azimuth* e elevação pré-definida, além de gravar novas posições pré-definidas. O sistema de controle foi desenvolvido e é capaz de guiar o usuário através da interface, receber seus comandos, e enviar os comandos equivalentes para os *drivers* de controle. A demonstração do funcionamento com uso de maquete foi implementada, porém não foi possível a realização de testes mais apurados, uma vez que os *drivers* utilizados não forneceram a potência mínima necessária para movimentar apropriadamente os motores. No entanto, a operação do sistema de controle foi considerada satisfatória pois seu funcionamento é independente da parte de movimentação dos motores.

Palavras-chave: Antena, eletrônica, microcontroladores, programação, telecomunicação.

ABSTRACT

In the broadcasting center of a satellite television company there is the need of pointing a receiving antenna to many different orbital positions with as much speed and practicality as possible. In this specific case, also based on professional knowledge acquired in an internship done through FATEC-SP, the antenna position system is motorized, but does not have a form of remote control. In this work an electronic circuit based on the microcontroller PIC18F4550 capable of controlling an antenna such as described was developed. For this extend, it was developed, beside the circuit, a program (source code) to control the system through the microcontroller and a model to demonstrate how the system works. This system allows the user to move the antenna through keys indicating azimuth or elevation directions, by typing a new position value and through selecting a previously recorded position of azimuth and elevation, beside recording a new pre-defined position as desired. The controlling system developed can guide the users through its interface, collect their commands and send them to the correspondent motor driver. It was possible to implement the demonstration using a model, but some specific tests were not performed due to a power failure on the drives, However, the system operation was considered adequate once it is not dependent on the driver neither the motor movement.

Palavras-chave: Antenna, eletronicas, microcontroller, coding, telecommunication.

SUMÁRIO

1. Introdução, objetivos e justificativas.....	11
2. Introdução teórica	14
2.2 A empresa de televisão por satélite e seu <i>modus operandi</i>	14
2.3 Antenas de recepção.....	16
2.4 Microcontrolador PIC18F4550.....	21
2.4.2 Recursos internos do PIC utilizados nesse projeto	22
3. Metodologia, materiais e ferramentas.....	28
3.2 Descrição da empresa-modelo e motivações.....	28
3.3 Metodologia de operação e metodologia de desenvolvimento	29
3.4 Materiais, componentes e ferramentas.....	33
3.4.2 Motor de passo.....	33
3.4.3 Driver TB6600 e fonte.....	34
3.4.4 Display de LCD 20x4.....	35
3.4.5 Softwares.....	35
4. Desenvolvimento	38
4.2 Maquete e acabamento	38
4.3 Interface.....	40
4.3.2 Modos de operação.....	41
4.4 Circuito e programação	42
4.4.2 Variáveis globais	43
4.4.3 Função main()	44
4.4.4 Circuito e programação do LCD	48
4.4.5 Teclado Analógico	50
4.4.6 Lógica circular de teste do teclado	55
4.4.7 Controle dos motores e funções de movimento manual.....	56
4.4.8 Função digitar()	58
4.4.9 Memória EEPROM	60
4.4.10 Função de seleção de posições pré-definidas.....	61
4.4.11 Função de gravação de posições pré-definidas	63
5. Resultados.....	66
6. Análise de custos do circuito de controle desenvolvido	68

7. Conclusão e trabalhos futuros	69
Referências bibliográficas	71
Apêndice A	73
Apêndice B	117
Apêndice C	120
Apêndice D	121
Apêndice E	122
Apêndice F	123

1. Introdução, objetivos e justificativas

De acordo com a ABTA, a Associação Brasileira de Televisão por Assinatura, em abril de 2017 58.0% do mercado de televisão por assinatura era composto de clientes do chamado modelo DTH (*Direct-to-Home*, direto para casa) [1]. Esse modelo consiste na transmissão via satélite do conteúdo televisivo contratado, que é captado por antenas de recepção instaladas nas casas dos clientes.

Apesar do mercado de televisão por assinatura vir apresentando uma queda pequena desde 2015, em comparação com o crescimento constante desde seu início de atividades no país e do crescimento de 100% entre 2010 e 2014, as empresas de televisão Oi e Sky Brasil, ambas de DTH, apresentaram crescimento em sua base de assinantes em 2017, respectivamente de 15,73% e 2,09% [2].

Além de utilizá-los na transmissão, empresas de DTH também recebem o sinal dos canais que integram seus pacotes através de satélites geoestacionários. Para captar esses sinais, elas possuem inúmeras antenas parabólicas de recepção dispostas em pátios, cada uma apontada para um satélite em uma posição específica da órbita terrestre.

Segundo Kraus [3], “uma *antena* de rádio pode ser definida como uma estrutura associada com a região de transição entre uma onda guiada e uma onda no espaço livre, ou vice-versa”. Neste trabalho, a palavra antena será usada para referenciar todo o conjunto da estrutura de sustentação, sistema de apontamento, prato refletor e amplificador de baixo ruído, o equipamento que efetivamente capta o sinal. Esse amplificador é conectado em um cabo coaxial que transmite o sinal para os equipamentos que irão distribuí-lo, tratá-lo ou analisá-lo.

De modo geral, o apontamento dessas antenas é feito por um sistema mecânico que precisa ser ajustado manualmente por um operador e deve ser realizado de modo que a captação do sinal seja a melhor possível. Para tal, é utilizado também um equipamento chamado analisador de espectro, que permite visualizar a intensidade do sinal em relação à frequência, como os fabricados pela Hewlett-Packard, além do auxílio de uma bússola e transferidor.

O processo de apontamento costuma ser realizado por duas pessoas: uma movimentando a antena no eixo azimutal ou de elevação ao ar livre e a outra acompanhando a intensidade do sinal com o analisador de espectro de dentro de um

prédio. Estes dois eixos, azimutal e de elevação são respectivamente a rotação num eixo perpendicular ao plano da Terra e o medido a partir do plano/solo da Terra, como ilustrado na Figura 1, em inglês *azimuth* (eixo azimutal) e *elevation* (eixo de elevação).

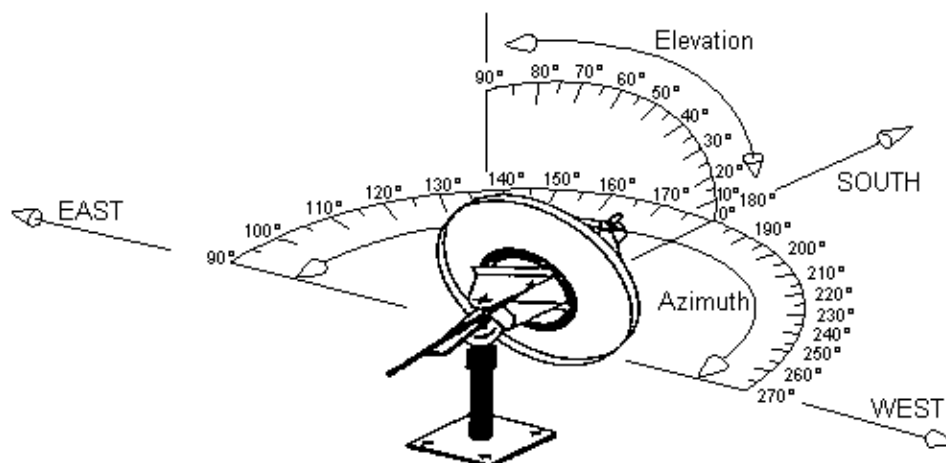


Figura 1 - Apontamento de uma antena [4].

Esta operação não é realizada com frequência no dia-a-dia, mas apenas na instalação das antenas, quando satélites são desativados ou quando canais específicos mudam de satélite. Entretanto, no caso de defeitos nos equipamentos que ficam na antena ou no caso de manutenções periódicas nas antenas em operação é necessário ter uma antena reserva que será apontada para o satélite em questão. Esta antena reserva deve permitir apontamento para qualquer satélite utilizado, e no caso de emergências, com a maior velocidade possível, uma vez que a indisponibilidade do sinal pode gerar perdas para a empresa. Por isso, em geral, essa antena reserva possui um sistema de apontamento motorizado.

Assim, o objetivo do presente trabalho é desenvolver um sistema que permita controlar à distância através de cabos uma antena motorizada de recepção de sinais de satélite, justificado uma vez que isso facilitará consideravelmente a operação das mesmas e agilizará o processo em caso de emergência, além de permitir a operação por apenas um operador dentro do prédio, o que elimina preocupações com as condições climáticas.

São objetivos específicos deste trabalho:

- a) Desenvolver um circuito eletrônico para envio de sinais de controle dos motores;
- b) Implementar, via microcontrolador, metodologia de operação dos controles, interfaces de controle das posições, modos de movimentação e outros;
- c) Projetar uma estrutura de demonstração do sistema com o suporte de uma antena de médio porte e pontos de fixação de motores e eixos;
- d) Avaliar o desempenho do conjunto quando parado ou em movimentação.

Este trabalho está dividido nos seguintes capítulos:

- 1) Introdução Teórica, descrevendo o funcionamento de um centro de transmissão de televisão por satélite e o funcionamento básico da operação de suas antenas de recepção.
- 2) Metodologia, materiais e ferramentas, elaborando os métodos utilizados na elaboração desse trabalho, além dos materiais e ferramentas utilizados.
- 3) Desenvolvimento, onde são explicados os mecanismos desenvolvidos.
- 4) Resultados.
- 5) Análise dos custos do circuito de controle desenvolvido.
- 6) Conclusões e trabalhos futuros.

2. Introdução teórica

Nesse capítulo são apresentados conceitos básicos de televisão por satélite e do principal componente do projeto, o microcontrolador PIC18F4550.

2.2 A empresa de televisão por satélite e seu *modus operandi*

Dentre as modalidades do negócio de televisão por assinatura existe aquela em que a transmissão é realizada por satélite, utilizando sinais de radiofrequência. Este é um modelo de negócio no qual o cliente capta o sinal de determinada empresa através de uma antena parabólica instalada em sua casa e tem acesso aos diversos canais transmitidos naquele sinal abrindo-o em um receptor. A assinatura desse serviço inclui a habilitação necessária para visualizar esses canais mediante ao pagamento de mensalidades. O conjunto de antena-receptor em geral é fornecido ao cliente pela empresa por comodato.

As empresas que atuam nesse segmento possuem centros de transmissão onde são realizados os processos necessários para transmitir o sinal para o cliente. Um centro de transmissão comum realiza basicamente os seguintes processos para realizar sua função: captação de sinal, tratamento de vídeo e transmissão. Estes processos são representados esquematicamente na Figura 2 e detalhados a seguir.

a) Captação de sinal: o sinal que a empresa disponibiliza para os clientes contém determinados canais que são gerados em outro lugar por produtoras de conteúdo. Para estes serem retransmitidos por uma empresa de televisão por assinatura é necessário antes captar o sinal de cada um. Esses canais podem ser transmitidos pelas suas geradoras através de alguns meios, como fibra óptica, transmissão terrestre, ou por transmissão via satélite. No caso dessa última forma a empresa de televisão recebe os canais utilizando antenas de recepção. Essas antenas captam o sinal do espaço e o transmitem via cabo até um receptor, que o demodula e decodifica os sinais.

b) Tratamento de vídeo: pode incluir uma série de outros processos, como a inserção de conteúdos da empresa (logo, menus, etc.), compressão de vídeo (para que mais canais possam ser enviados em um mesmo sinal de capacidade de banda limitada), e multiplexação, que une os áudios e vídeos de diversos canais em um mesmo sinal.

c) Transmissão: os sinais de vídeos tratados em geral são sinais digitais, que precisam ser modulados em sinais de radiofrequência para serem transmitidos. Dependendo dos equipamentos utilizados na modulação e da faixa de frequência utilizada pela empresa, ainda é necessário transladar esse sinal para outras frequências. O sinal modulado então tem sua potência amplificada e é transmitido para um outro satélite através da antena de transmissão. O satélite é responsável por retransmitir o sinal da empresa de televisão para todo território em que estão os clientes.

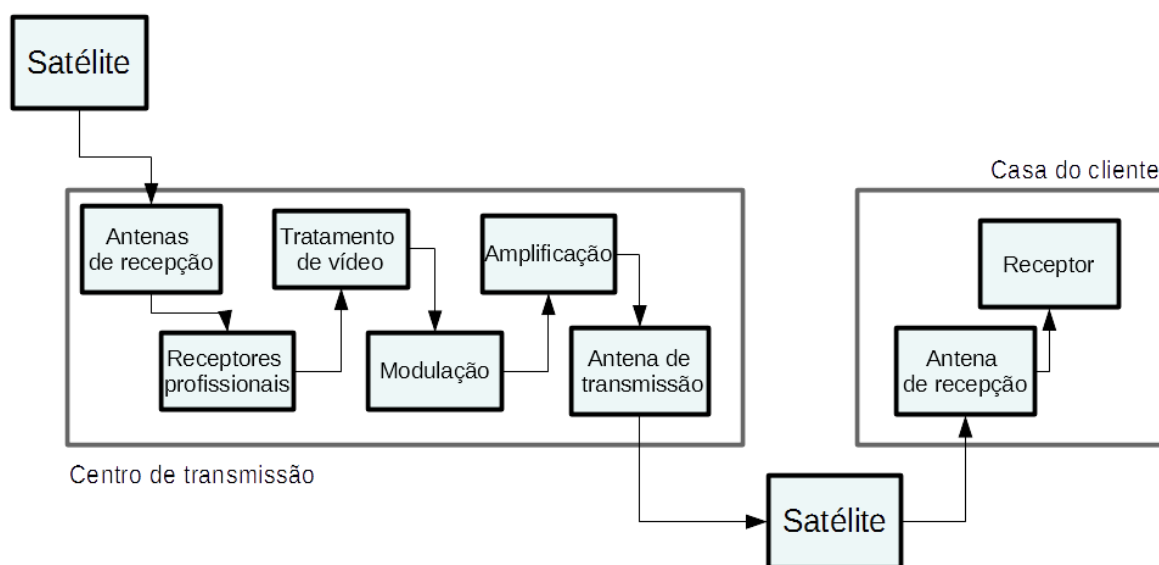


Figura 2 - Cadeia de operações de uma empresa de transmissão de televisão DTH.

2.3 Antenas de recepção

A fase de captação de sinal envolve diversos canais, que são transmitidos pelas produtoras através de diversos satélites localizados em posições diferentes da órbita terrestre. É necessário ter uma antena de recepção apontada para cada satélite envolvido. No caso em que esse trabalho é baseado são utilizadas 25 antenas parabólicas de recepção dispostas em um pátio, como mostrado na Figura 3.



Figura 3 – Um pátio de antenas de recepção.

Essas antenas trabalham recebendo sinais que trafegam na chamada banda C, que segundo o IEEE compreende as faixas de frequência entre 4GHz e 8GHz [5]. Esses sinais podem se propagar com polarização circular ou ortogonal, caso no qual podem ser transmitidos dois sinais distintos ao mesmo tempo: um vertical e um horizontal.

A parábola das antenas tem a função de refletir e concentrar o sinal para que ele seja efetivamente captado por um conversor de baixo ruído, o LNB (*low noise block-downconverter*). Esse equipamento é conectado a um receptor através de um cabo coaxial e suspenso sobre o centro da parábola com um conjunto chamado de

“alimentador”, como mostrado na Figura 4. O LNB tem a função de amplificar a potência do sinal e transladar sua frequência gerando o menor ruído possível para que os receptores possam decodificá-lo.



Figura 4 - LNBs horizontal e vertical conectados a um alimentador.

Em geral, essas antenas podem ter 4,5 metros de diâmetro, e todo o conjunto de antena e pedestal pode pesar cerca de 2500 kg [6].

Essas antenas possuem um sistema de apontamento mecânico, que utiliza arranjos de porcas e roscas ajustados manualmente, como mostrado na Figura 5. Esses sistemas foram feitos para ficarem fixos por longos períodos de tempo, uma vez que os satélites para os quais as antenas estão apontadas são geoestacionários, se mantendo fixos em um ponto da órbita da terra, além de terem longa vida útil. Pela natureza mecânica, são demorados para apontar e requerem esforço físico, além do acompanhamento do nível de sinal com a utilização de um analisador de espectro.

Para apontar uma antena é necessário movimentar sua parábola nos eixos vertical e horizontal tendo-se em mente a posição orbital do satélite, o que em geral requer o auxílio de uma bússola e transferidor. No caso de sinais de propagação ortogonal ainda é necessário ajustar a polarização do LNB para que estes estejam alinhados com os sinais vertical e horizontal.



Figura 5 - Sistemas de apontamento mecânicos.

Se tratando do negócio de televisão por assinatura, garantir a entrega de sinal para os clientes é essencial e qualquer problema nas antenas de recepção pode gerar perdas para a empresa. Por isto, as empresas devem dispor de antenas reservas prontas para receber o sinal de qualquer um dos satélites utilizados. Além disso, a realização de manutenções preventivas periódicas e que requerem que a antena saia de funcionamento também torna necessário o uso de antenas reserva.

No caso da empresa estudada são utilizadas duas antenas reserva, das marcas Andrew e Scientific Atlanta, que serão chamadas respectivamente de A e B, para as vinte e cinco antenas em funcionamento. A antena A é utilizada para sinais de propagação ortogonal e a B para sinais de propagação circular. Essas antenas podem ser vistas na Figura 6.



Figura 6 - Antenas A e B .

No caso de emergências, para que a interrupção da transmissão do sinal seja a menor possível é necessário que o apontamento da antena reserva seja feito com rapidez, por isso, em vez do sistema mecânico, as antenas reserva são equipadas com motores controlados que movem suas parábolas nos eixos vertical e horizontal, como visto na Figura 7.



Figura 7 - Sistemas de posicionamento motorizados.

O posicionamento da antena A é controlado por um equipamento chamado APC-100, um controlador de antenas programável, localizado dentro do prédio do centro de transmissão. Com esse equipamento, é possível para o usuário controlar manualmente a parábola nos eixos vertical e horizontal e girar o conjunto do alimentador em cerca de 150°, através de um teclado e display de LCD. Também é possível armazenar via comunicação serial posições pré-definidas que contenham informações como o nome do satélite e posicionamento, segundo os quais o APC-100 pode realizar o apontamento automático. Em ambos os casos é necessário o acompanhamento com um analisador de espectro para verificar se o sinal está sendo recebido de forma correta.

O APC-100 reconhece e informa a movimentação e posição da antena Andrew através de transformadores elétricos (*resolvers*) em seus motores. Esse equipamento pode ser adquirido por US\$ 2600 [7].

O posicionamento da antena B é feito por botões ligados a contadoras e relés em sua base e dentro de um painel, que controlam o acionamento dos motores, conforme mostrado na Figura 8.

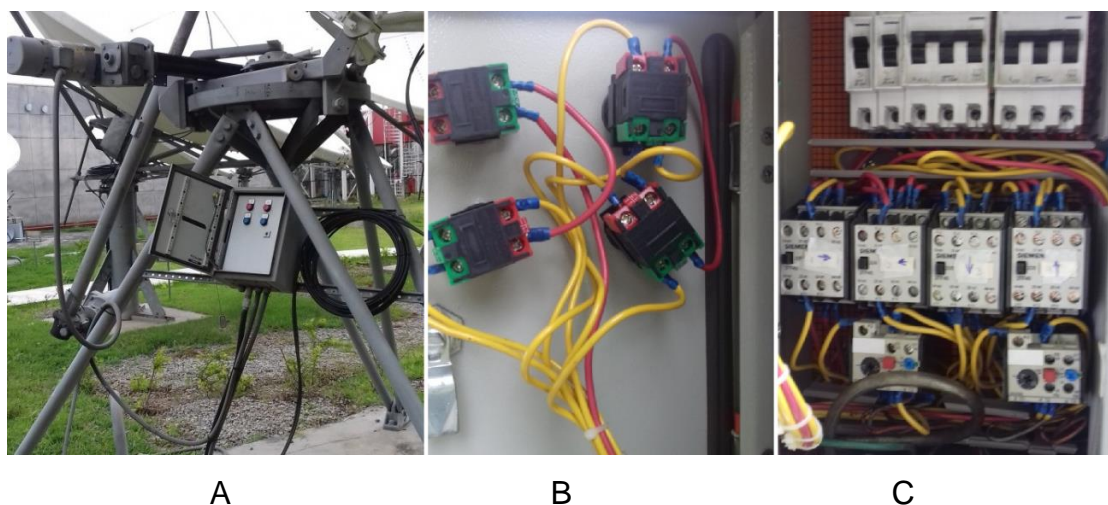


Figura 8 - Botões de controle (A), que acionam conjuntos de contadoras e relés (B e C)

Os sinais captados por essas antenas chegam dentro do prédio do centro de transmissão através de cabos RG-11, cerca de 60 metros, e são roteados por uma matriz para diversos equipamentos, incluindo um analisador de espectro para monitorar o sinal.

2.4 Microcontrolador PIC18F4550 [8]

Um microcontrolador é a união de diversos dispositivos em um único chip, como por exemplo: microprocessador, oscilador, memória RAM, memórias de programa e etc. Diferente dos microprocessadores, possuem capacidade de memória considerada baixa se comparados com produtos mais complexos, como por exemplo um computador. Porém, para projetos na área acadêmica e de automação simples, sua capacidade é mais que suficiente. Seus componentes internos, facilidade e flexibilidade para a ligação a periféricos foram uma revolução na indústria. Placas que antes eram grandes e robustas, puderam diminuir o seu tamanho e aumentar a eficiência do produto eletrônico. Microcontroladores estão presentes em quase todos os produtos eletrônicos na atualidade.

O Microcontrolador usado neste projeto é o PIC18F4550. O PIC é uma linha de microcontroladores fabricados pela MICROCHIP. Existem basicamente quatro famílias de PICs diferenciadas pelo tamanho da palavra de memória de programa: 12, 14, 16 e 32 bits, e possuem uma arquitetura interna do tipo Harvard. A diferença entre essa arquitetura e as tradicionais, do tipo Von-Neumann, é que ela possui um barramento para o programa e outro para os dados, diferente da arquitetura tradicional em que um barramento é tanto de dados como de endereço. A letra F (PIC18**F**4550) identifica que o microcontrolador em questão utiliza a tecnologia FLASH, ou seja, pode ser regravado. Outra característica importante da arquitetura PIC reside na semelhança e compatibilidade entre os diversos microcontroladores membros de sua família. Isto facilita grandemente a migração de microcontrolador para outro, bastando mudar, em alguns casos, apenas alguns comandos no programa, pois partes dos registradores internos não diferem muito entre si.

O PIC utilizado possui 40 pinos, como mostrado na Figura 9, alguns com funcionalidades específicas, como alimentação, e outros que podem ser configurados como entradas ou saídas de dados, agrupados em 5 grupos chamados de *PORTs*: RA, RB, RC, RD e RE.

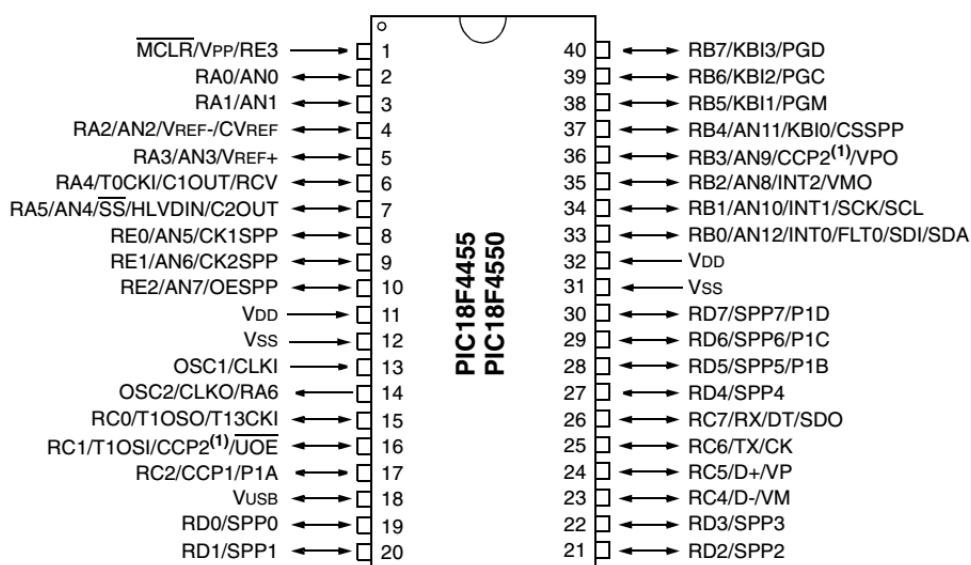


Figura 9 - Pinagem do PIC18F4550 [8].

O PIC18F4550 vem com diversas funcionalidades internas controladas por registradores pré-definidos e descritos em seu *datasheet*⁸, entre essas funcionalidades estão a memória EEPROM, interrupções, temporizadores, comparadores e conversores. Os principais elementos do PIC18F4550 utilizados nesse trabalho foram sua memória EEPROM, módulo conversor de analógico para digital e interrupção.

2.4.2 Recursos internos do PIC utilizados nesse projeto [8]

a) Memória EEPROM

Além da memória para dados e programa, o PIC18F4550 possui uma memória EEPROM (*electrically erasable programmable read-only memory*). A EEPROM é uma memória não-volátil separada da memória de programa e de dados, e é usada para armazenar dados a longo prazo.

No caso do PIC18F4550, a memória EEPROM possui 256 bytes de espaço, ou seja, 256 endereços de memória capazes de armazenar informações de 8 bits.

Segundo seu *datasheet*, o PIC18F4550 possui registradores que comandam a gravação e leitura de dados na memória EEPROM, porém, no desenvolvimento deste

trabalho utilizou-se funções disponíveis na biblioteca do compilador utilizado, que facilitam seu uso.

b) Módulo de conversão analógico-digital

O PIC18F4550 possui um módulo conversor de analógico para digital de 10 bits de resolução com 13 canais de entrada. Esse módulo é controlado por cinco registradores: ADRESH, ADRESL, ADCON0, ADCON1 e ADCON2.

Como o resultado da conversão possui 10 bits um registrador de 1 byte não é suficiente para armazená-lo, por isso os bits mais significativos do resultado são registrados em ADRESH e os menos significativos em ADRESL.

A Figura 10 mostra os bits do registrador ADCON0, que seleciona o canal a ser utilizado e o estado do módulo conversor:

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							
						bit 0	

Figura 10 – Registrador ADCON0 do PIC18F4550 [8].

Os bits 7 e 6 não são utilizados. A combinação dos bits 5-2 (CHS3-CHS0) indica qual dos 13 canais é utilizado para realizar a conversão em determinado instante. Por exemplo: esses bits serem setados como 0000 seleciona o canal 0 (pino AN0) como entrada do valor que será convertido, e 1100 seleciona o canal 12 (pino AN12).

O bit 0 (ADON) indica se o módulo conversor está ativo (1) ou não (0). Quando o conversor está ativo o bit 1 (GO/DONE) indica se a conversão está sendo realizada (1) ou não (0).

A Figura 11 apresenta o registrador ADCON1, que seleciona as entradas de tensão de referência e quais canais estarão configurados como entrada analógica:

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0 ⁽¹⁾	R/W ⁽¹⁾	R/W ⁽¹⁾	R/W ⁽¹⁾
—	—	VCFG0	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Figura 11 – Registrador ADCON1 do PIC18F4550 [8].

Os bits 7 e 6 não são utilizados. O bit 5 (VCFG0) seleciona a tensão de referência de fundo de escala, quando é setado como 1, essa tensão é a aplicada ao pino AN2 (Vref-), e quando 0 é o terra conectado ao PIC (VSS). O bit 4 (VCFG0) seleciona a tensão de referência do topo da escala, quando 1 essa é a tensão aplicada ao pino AN3 (Vref+) e quando 0 é a tensão de alimentação do PIC (VDD).

Os bits 3-0 (PCFG3-PCFG0) selecionam quais canais estarão configurados como entrada analógica ou como pinos entrada-saída digital comuns, de acordo com a Tabela 1.

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

Tabela 1 – Tabela descrevendo as possíveis combinações dos bits 3-0 do registrador ADCON1 [8].

A Figura 12 apresenta o registrador ADCON2, responsável pelo formato de saída do resultado, tempo de aquisição e *clock* de conversão:

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

Figura 12 – Registrador ADCON2 do PIC18F4550 [8].

O bit 7 (ADFM) seleciona o formato de registro do valor de 10 bits nos 16 bits dos registradores ADRESH e ADRESL. Quando 1, o resultado é justificado à direita (ADRESH: 000000XX e ADRESL:XXXXXXXX), e quando 0, à esquerda (ADRESL: XXXXXXXX e ADRESL: XX000000).

O bit 6 não é utilizado. Os bits 5-3 (ACQT2-ACQT0) definem o tempo de aquisição, 111 = 20 TAD e 000 = 0 TAD, por exemplo. Os bits 2-0 (ADCS2-ADCS0) definem o *clock* de conversão, 111 seleciona o *clock* derivado do oscilador RC do conversor A/D e 010 utiliza FOSC/32.

c) Interrupção

O PIC18F4550 possui múltiplas fontes de interrupção e a função de estruturá-las em diferentes níveis de prioridade. Há dez registradores utilizados para controlar o funcionamento das interrupções do PIC18F4550: RCON, INTCON, INTCON2, INTCON3, PIR1, PIR2, PIE1, PIE2, IPR1, IPR2.⁸

Cada fonte de interrupção possui basicamente três bits para controlar seu funcionamento: um bit de indicação (*flag*), um bit de ativação (*enable*) e um bit de prioridade.

A Figura 13 apresenta o registrador RCON:

REGISTER 4-1: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1 ⁽¹⁾	U-0	R/W-1	R-1	R-1	R/W-0 ⁽²⁾	R/W-0
IPEN	SBOREN	—	RI	TO	PD	POR	BOR
bit 7							bit 0

Figura 13 – Registrador RCON do PIC18F4550 [8].

Este registrador é responsável pelas funções de reset do PIC, e em relação à função de interrupção apenas seu bit 7 é relevante: o bit IPEN. Em estado 1 esse bit habilita a função de prioridade para as interrupções, e em estado 0 a desabilita.

A Figura 14 mostra o registrador INTCON, que contém vários bits de indicação, ativação e prioridade:

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF ⁽¹⁾
bit 7							bit 0

Figura 14 - Registrador INTCON do PIC18F4550 [8].

Quando IPEN = 0, o bit 7 habilita (1) ou desabilita (0) as interrupções globalmente, e quando IPEN = 1, ele habilita ou não apenas as interrupções de alta prioridade.

Quando IPEN = 0, o bit 6 habilita (1) ou desabilita (0) as interrupções por periféricos, e quando IPEN = 1, ele habilita ou não apenas as interrupções de baixa prioridade.

O bit 5 habilita (1) ou não (0) a interrupção devido ao acionamento do TMR0, que é uma das funções de temporizador do PIC18F4550. O bit 4 habilita (1) ou não (0) a interrupção externa no pino RB0 do PIC18F4550, chamada de interrupção INT0. O bit 3 habilita (1) ou não (0) a função de interrupção por mudança de estado nos pinos RB do PIC.

Os bits 2-0 são os bits de indicação que uma interrupção ocorreu, respectivamente, devido ao TMR0, ao INT0 ou à mudança de estado nos PORTs RB.

Na Figura 15 são apresentados os bits do registrador INTCON2.

REGISTER 9-2: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

Figura 15 – Registrador INTCON2 do PIC18F4550 [8].

Bit 3 e bit 1 não são utilizados. O bit 7 habilita (0) uma funcionalidade dos pinos RB, que são os *pull-up* internos, circuitos que facilitam a utilização dos pinos como entrada de sinal. Os bits 6-4 definem a interrupção nos pinos RB0-RB2 como para acontecer na borda de subida ou descida dos pulsos. E os bits 2 e 0 definem a prioridade das interrupções do TMR0 e dos pinos RB.

E por último, o registrador INTCON3 na Figura 16.

REGISTER 9-3: INTCON3: INTERRUPT CONTROL REGISTER 3

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
bit 7						bit 0	

Figura 16 – Registrador INTCON3 do PIC18F4550 [8].

Os bits 2 e 5 não são utilizados. Os bits 7 e 6 definem a prioridade das interrupções externas respectivamente nos pinos RB2 e RB1 (1 para alta e 0 para baixa) e os bits 4 e 3 habilitam (1) ou desabilitam essas interrupções. Os bits 1 e 0 são os indicadores de que essas interrupções ocorreram (1) ou não (0).

Os registradores PIR e PIE são, respectivamente, relativos aos indicadores e habilitadores das interrupções de periféricos. São utilizados em conjunto com módulos do PIC como comparadores e temporizados e não são configurados nesse trabalho.

3. Metodologia, materiais e ferramentas

Neste capítulo serão discutidos os métodos utilizados na elaboração do trabalho, bem como os componentes e ferramentas.

3.2 Descrição da empresa-modelo e motivações

Como visto, no centro de transmissão estudado para o desenvolvimento deste trabalho, há 25 antenas de recepção em funcionamento e que utilizam apontamento mecânico, para apenas duas antenas reserva com apontamento motorizado, as antenas A e B. O apontamento motorizado permite maior agilidade e praticidade, que é imprescindível no caso de emergências e muito útil nos casos em que seu acionamento é planejado, porém apenas o modelo da antena A permite o controle remoto.

Na situação atual, para realizar o apontamento da antena B é necessário que um operador fique em sua base, fora do prédio do centro de transmissões, utilizando uma bússola e transferidor para verificar a posição, e um segundo operador fique dentro do prédio utilizando um analisador de espectro. Esse analisador é alimentado pelo conversor de baixo ruído da antena para monitorar a chegada do sinal e indicar para o primeiro em que posição o sinal está melhor através de rádios. Dentro do prédio há equipamentos que alimentam o conversor e distribuem o sinal para o analisador e outros equipamentos.

O apontamento poderia ser realizado por apenas um operador. Mas para isso, simplificarmente, ele precisaria subir na antena, retirar os cabos que vão para dentro do prédio, colocar cabos novos que cheguem à base da antena e utilizar um analisador de espectro portátil, um divisor de radio-frequência e uma fonte de tensão para alimentar o conversor de baixo ruído que integra a antena. Isso, além do trabalho excessivo e demorado, não poderia ser realizado em tempo chuvoso. Implementar uma controladora similar a existente na antena A facilitaria consideravelmente a operação, eliminando a preocupação com a condição climática e a necessidade do uso de bússola e transferidor.

Além disso, no caso de problemas com os equipamentos em uma antena em operação, a antena de reserva deverá ser rapidamente movimentada para a posição da antena que apresentou defeito, reduzindo ao máximo o tempo de corte do sinal. Uma vez que o sistema de movimentação da antena é motorizado, esta pode ser direcionada à distância, permitindo, portanto que o mesmo operador pode tanto movimentar a antena quanto analisar o sinal sendo recebido, aumentando a eficiência na recuperação da transmissão, além de eliminar o tempo de deslocamento até a antena e da verificação da posição.

Este sistema deverá ter como características:

- a) Possibilitar o controle nos eixos vertical e horizontal, informando a posição com precisão de pelo menos 1º e em tempo real.
- b) Gravar posições e nomes de satélites e realizar o apontamento automático a partir deles.
- c) Ter interface amigável ao operador.
- d) Ter baixo custo.

3.3 Metodologia de operação e metodologia de desenvolvimento

Se tratando de um equipamento a ser utilizado por um operador, o início deste projeto foi a elaboração de um fluxograma indicando como o equipamento seria utilizado, a partir do qual foram elaborados os fluxogramas mais detalhados das funções que o equipamento realizaria.

O circuito foi montado na ferramenta de simulação, como por exemplo o Proteus ® (Figura 17) e a programação foi feita por etapas, primeiro fazendo os programas que utilizam cada parte do circuito separadamente e depois o programa que utiliza todas essas funções para que o sistema de controle funcione.

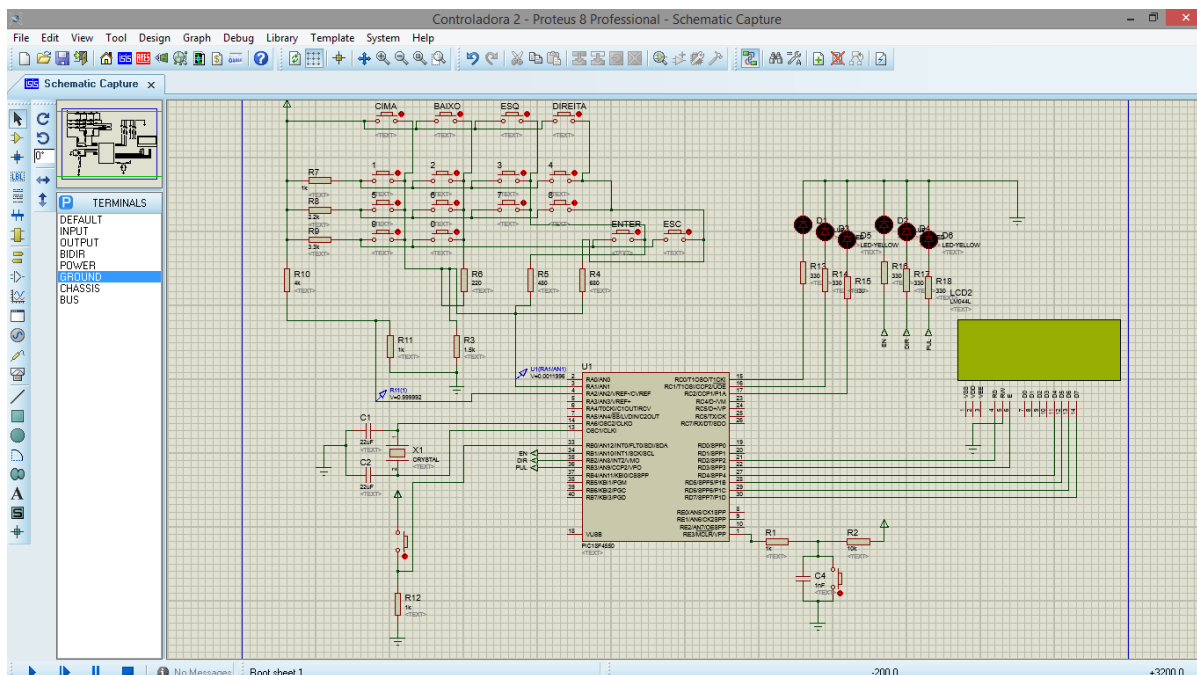


Figura 17 - Simulação do circuito projetado.

Quando possível, as etapas individuais da programação foram inseridas no programa principal como sub-rotinas, funções declaradas antes do início do programa e que podem ser chamadas quando necessário.

Em especial nos momentos em que ocorre a gravação da memória EEPROM a ferramenta *Debugger* do simulador foi utilizada, ela permite que se acompanhe linha a linha a simulação do programa carregado ao mesmo tempo em que se monitora a memória EEPROM simulada do microcontrolador, como mostrado na Figura 18. Para utilizar essa função é necessário carregar no microcontrolador da simulação o arquivo de extensão *.COFF* (*Common Object File Format*) gerado pelo compilador.

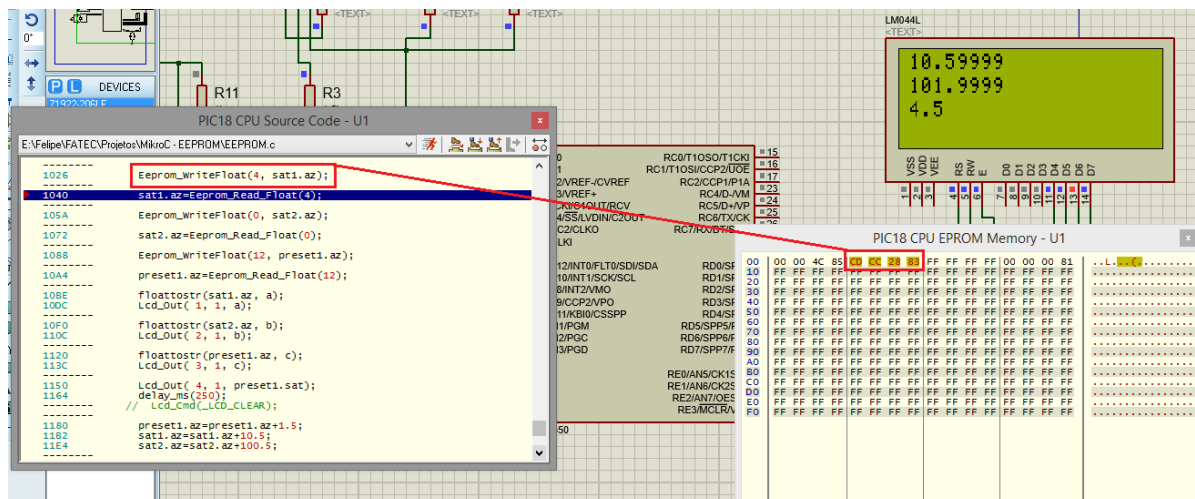


Figura 18 - Simulação da memória EEPROM.

Paralelamente à simulação e elaboração dos programas foi realizada a montagem em matriz de contatos do circuito e, posteriormente, os testes em bancada utilizando os componentes físicos, conforme pode ser visto na Figura 19.

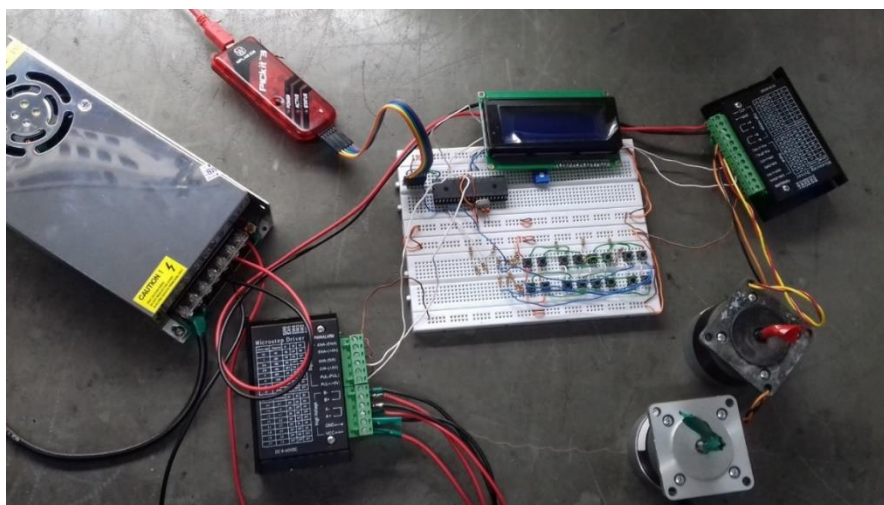


Figura 19 - Montagem do sistema para testes em bancada.

Nessa fase foi utilizado o gravador PIC PICkit 3 e sua função de gravação ICSP (*in-circuit serial programming*), que não requer que o PIC seja retirado do circuito para ser gravado. Também foi gerado o layout da placa de circuito impresso, como visto na Figura 20.

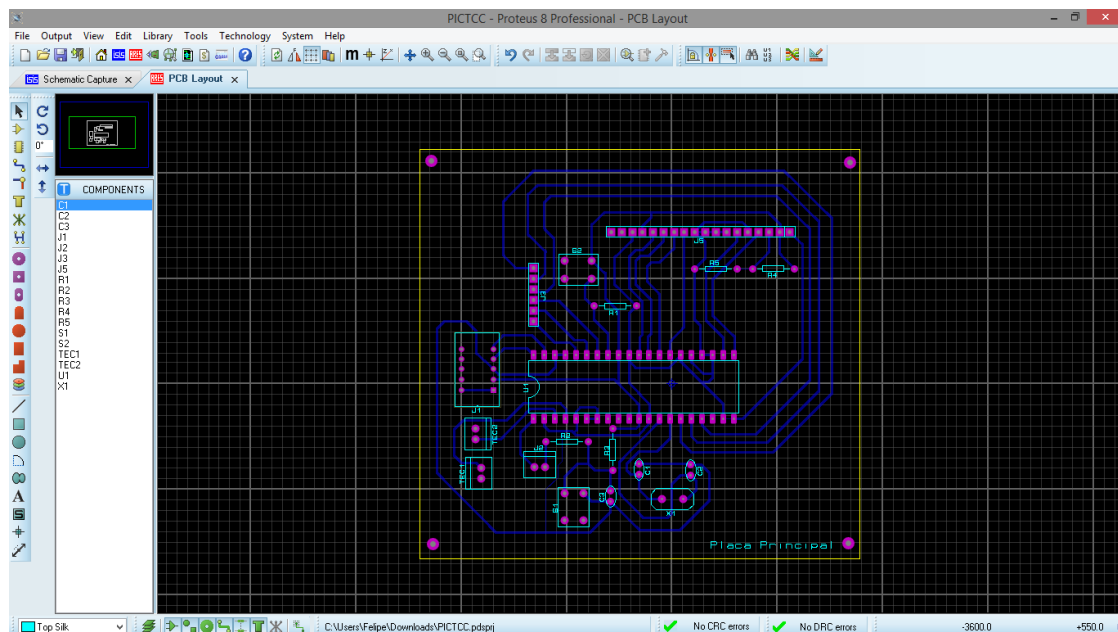


Figura 20 - Layout da placa de circuito impresso feito em software, como por exemplo o ARES ®.

O fotolito pronto para impressão da placa do teclado pode ser encontrado no Apêndice D, e o da placa principal pode ser encontrado no Apêndice F. Ambos podem ser vistos na Figura 21 e a montagem prática das placas na Figura 22.

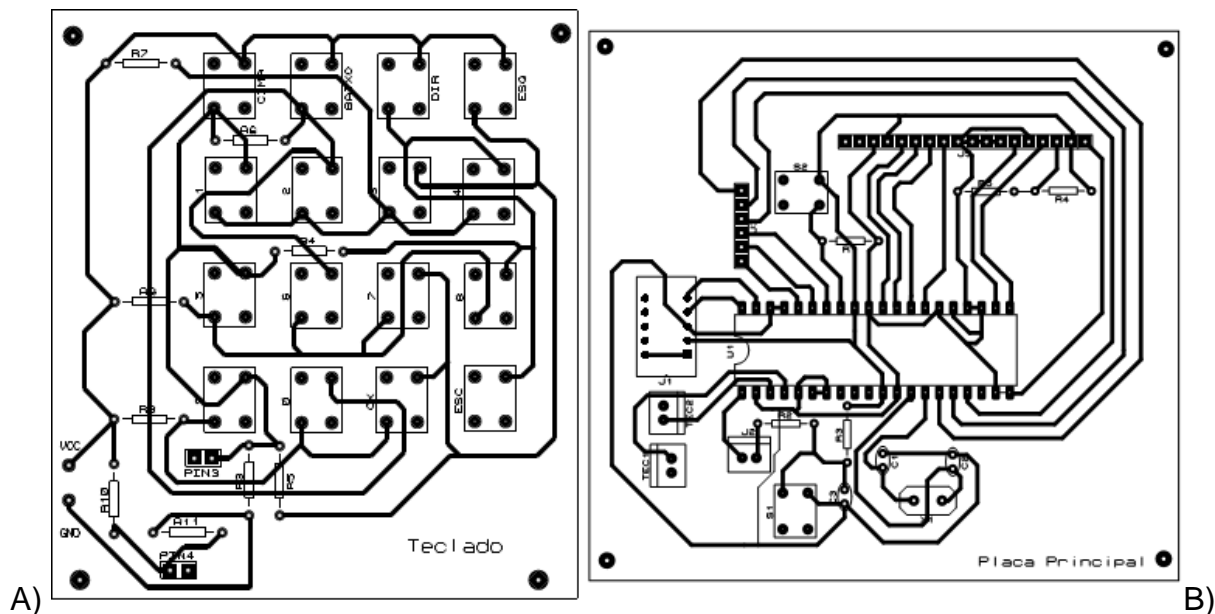


Figura 21 - Fotolito (fora de escala, com indicação de componentes e não espelhado) das placas do teclado (A) e PIC (B).

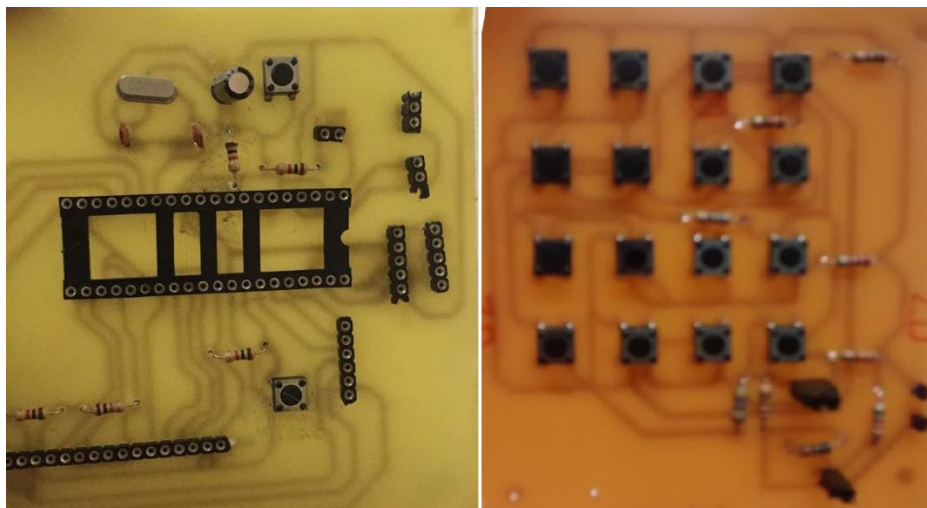


Figura 22 - Montagem das placas

3.4 Materiais, componentes e ferramentas

Nesta seção serão apresentados os materiais, componentes e ferramentas utilizados.

3.4.2 Motor de passo [9]

Os Motores de Passo são dispositivos eletromecânicos que convertem pulsos elétricos em movimento, rotacionando o eixo do motor em pequenos incrementos angulares, denominados “passos”, quando pulsos elétricos são aplicados em uma sequência específica em seus terminais. A rotação desses motores é diretamente relacionada aos pulsos elétricos recebidos, assim como a direção em que o motor gira. Um motor de passo pode ser uma boa escolha em aplicações em que é necessário controlar fatores como: ângulo de rotação, velocidade, posição e sincronismo. O ponto forte de um motor de passo, no entanto não é a sua força, nem capacidade de desenvolver altas velocidades, e sim a possibilidade de controlar seus movimentos de forma precisa.

O motor de passo foi escolhido para a demonstração do circuito desenvolvido neste trabalho por permitir um controle preciso e simples do ângulo de movimentação através de pulsos e programado no microcontrolador.

O motor aplicado nesse projeto é um motor de passo AK23/15F6FN1.8 de passo de $1,8^\circ$ e sendo utilizado em ligação bipolar, ou seja, no esquema mostrado na Figura 21 não são utilizados os terminais branco e preto. Segundo o *datasheet* do motor [10], a bobina do mesmo possui resistência de 2Ω e, em ligação bipolar, ele deveria consumir 8.82W ($2.1\text{ A} \times 4.2\text{V}$). A resistência medida experimentalmente na bobina do motor foi de 2.2Ω .

3.4.3 Driver TB6600 e fonte

Para controlar os motores de passo foi utilizado um driver da marca DRRobot baseado no CI TB6600. Esse driver funciona com três entradas digitais: EN, DIR e STEP. A entrada DIR controla a direção em que o motor deverá girar, a entrada STEP é o passo e a entrada EN controla a alimentação das bobinas do motor. [11]

Esse driver possui a função de micro-passo, que consiste em controlar a corrente e tempo de pulso fornecido ao motor para que ele se movimente em frações de um passo. Essa função pode ser controlada através da combinação de 3 chaves (S1, S2 e S3) de acordo com a Tabela 2.

Micro-passo	Pulsos/revolução	S1	S2	S3
-	-	1	1	1
1	200	1	1	0
2/A	400	1	0	1
2/B	400	0	1	1
4	800	1	0	0
8	1600	0	1	0
16	3200	0	0	1
32	6400	0	0	0

Tabela 2 - Configuração de micropassos [11].

A diminuição do grau do passo causa perda do torque de movimentação do motor.

Sua alimentação é de 9 V a 42 V e corrente de saída de até 3,5A, que pode ser controlada atrás de 3 chaves (S4, S5 e S6) de acordo com a Tabela 3.

Corrente (A)	S4	S5	S6
0,5	1	1	1
1,0	1	0	1
1,5	1	1	0
2,0	1	0	0
2,5	0	1	1
2,8	0	0	1
3,0	0	1	0
3,5	0	0	0

Tabela 3 - Configuração de corrente [11].

3.4.4 Display de LCD 20x4

O display de LCD utilizado no projeto foi o 2004A, com capacidade para 20 caracteres dispostos em 4 linhas, entrada de dados de 4 ou 8 bits e LED de fundo, o chamado *backlight* [12].

3.4.5 Softwares

Nesta seção são apresentados os softwares utilizados para simulação e programação.

3.4.5.2 Compilador

A programação deste trabalho foi escrita em linguagem C embarcada e para gerar o arquivo em linha de comando foi utilizado um compilador específico para PIC, como por exemplo o MikroC ®.

Primeiramente, é preciso entender que a linguagem dos processadores, incluindo os microcontroladores, é o código de máquina. Se o programador necessitasse desenvolver seu programa nessa linguagem, sua tarefa seria extremamente árdua. Observando a dificuldade empregada, desenvolveu-se uma linguagem chamada de Assembly, que nada mais é do que a representação das ações da linguagem de código de máquinas em mneumônicos, ou seja, abreviações de termos usuais das operações efetuadas pelo comando em código de máquina.

Porém, ainda que tenha facilitado exponencialmente a programação, a linguagem Assembly é considerada de baixo nível, ou seja, não possui nenhum comando, instrução ou função além daqueles definidos no conjunto de processador utilizado. Isso implica em um trabalho extra do programador para desenvolver rotinas e operações que não fazem parte do conjunto de instruções do processador, produzindo, por conseguinte, programas muito extensos e complexos com um fluxo muitas vezes difícil de ser seguido. Com a proposta de ser intermediadora entre o código de máquina e o Assembly, foi criada a linguagem C. Visto que se trata de uma programação estruturada, onde o programa é dividido em funções específicas para realização de determinadas tarefas, programas de maior complexidade puderam ser elaborados.

Um compilador nada mais é do que um tradutor da linguagem escrita pelo programador, para o código de máquina. Essa tradução pode ser direta, ou indireta, ou seja, é feita a tradução para a linguagem Assembly para posteriormente ser feita a tradução para o código de máquina.

Além de compilar as instruções descritas em C embarcado, o compilador dispõe de uma biblioteca de funções pré-definidas para a utilização na programação de PICs, funções que envolvem conversão de variáveis, utilização de periféricos comuns (como displays de LCD), e gravação em memórias.

Para gravar a programação no microcontrolador foi utilizado o gravador PICKit 3, que pode ser utilizado em dispositivos das linhas PIC16F, PIC18F, dsPIC33f, PIC24 e PIC 32, e possui conexão USB e gravação ICSP (*in-circuit serial programming*).

3.4.5.3 Simulador

Foi utilizado um software de simulação e desenvolvimento de circuitos eletrônicos, que também permite o *design* de placas de circuito impresso, como por exemplo o Proteus®. Em especial, esse software é capaz de simular o funcionamento de uma grande variedade de microcontroladores PIC utilizando um arquivo .HEX gerado em uma compilação, além de simular a memória EEPROM interna do PIC.

4. Desenvolvimento

4.2 Maquete e acabamento

Além da montagem física do circuito, foi confeccionada uma maquete de antena para demonstração do funcionamento do conjunto, com base, suporte e ligações para os motores. O prato refletor da antena foi feito de cartolina e a haste de sustentação com um cano de PVC. A estrutura pode ser vista na Figura 23.



Figura 23 – Estrutura da antena.

Um esquema de montagem e ilustração dos componentes da maquete pode ser encontrado no Apêndice C. Como pode ser visto no referido apêndice e na Figura 24, os eixos dos motores realizam a movimentação da antena diretamente.

Também foi confeccionada uma caixa de madeira que comporta os drivers, fonte de 24V/10A, fonte de 5V para o circuito lógico, e as placas confeccionadas dos circuitos. Esta caixa pode ser vista nas Figuras 24 (exterior) e 25 (interior).



Figura 24 - Exterior da caixa do sistema projetado.

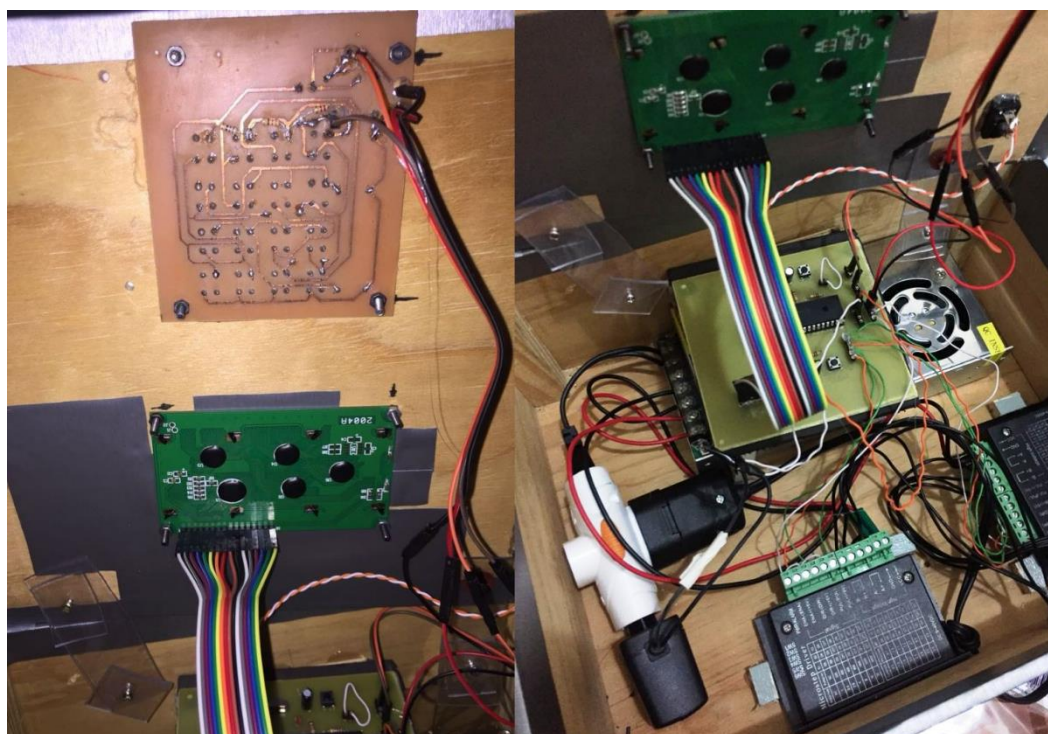


Figura 25 - Interior da caixa do sistema projetado.

4.3 Interface

Para permitir a movimentação à distância de uma antena por um operador foi necessário desenvolver uma interface através da qual este operador envia as instruções para o circuito que irá controlar o movimento da antena.

O funcionamento dessa interface segue a seguinte sequência:

- a) **Tela de apresentação**, com o nome do projeto e de seus autores.
- b) **Menu inicial**, onde é possível escolher entre a função de posição pré-definida (item c.) e a função de movimentação manual (item d.).
- c) **Menu da função de posição pré-definida**, onde é possível escolher entre a função de seleção (item e.) e a função de gravação (item f.).
- d) **Menu da função de movimentação manual**, onde é possível escolher entre a Tela de movimentação manual do eixo azimutal (item g.) e a Tela de movimentação manual do eixo de elevação (item h.).
- e) **Função de seleção de posição pré-definida**, ao selecionar uma posição pré-definida é mostrada uma tela para o usuário acompanhar a movimentação nos dois eixos.
- f) **Função de gravação de nova posição**, ao selecionar a posição da memória em que será gravada a nova posição, são exibidas telas guiando o usuário para digitar os novos valores.
- g) **Tela de movimentação manual no eixo azimutal**, em que o usuário pode movimentar a antena através das setas de direção, ou digitar um novo valor (item i.).
- h) **Tela de movimentação manual no eixo de elevação**, funcionando de forma análoga ao mostrado no item g). Ou digitar um novo valor (item j.).
- i) **Tela para digitar um novo valor de azimuth**, que guia o usuário para digitar o novo valor que, se selecionado, acompanha a movimentação no eixo.
- j) **Tela para digitar um novo valor de elevação**, análogo ao item i.

Em cada uma dessas telas é possível o usuário retornar à anterior quando desejar.

O usuário também tem a opção de liberar os eixos dos motores e realizar um ajuste manual de posição, uma espécie de calibração.

4.3.2 Modos de operação

Basicamente, o sistema possui três formas de controlar os motores da antena: através de posições pré-definidas, através de digitar um novo valor para a posição de um dos eixos ou através das teclas direcionais.

A função de posição pré-definida envolve duas etapas: gravar uma posição e selecionar uma posição. Essas posições gravadas englobam um valor para o eixo azimutal e um valor para a elevação, e estão atreladas a um satélite (satélite 1, satélite 2, etc.). A ideia é que o operador possa gravar várias posições referentes aos vários satélites utilizados em sua empresa e selecionar o apropriado na hora que quiser, para que a antena se movimenta nos dois eixos automaticamente. No programa e interface da controladora, a “posição pré-definida” é referida como *preset*.

Digitar um novo valor para um dos eixos é uma função que pode ser utilizada antes de gravar uma nova posição. Sabendo as coordenadas de um novo satélite, o usuário digita uma das posições, movimenta a antena, digita a outra, movimenta a antena e pode verificar se elas estão corretas, para então gravar a nova posição na memória do sistema.

Movimentar o satélite através das teclas direcionais é o modo mais demorado, já que o usuário necessita manter o botão apropriado apertado até que o motor se movimenta o bastante. No entanto, essa função é útil para realizar um ajuste fino, excursionando os limites de uma posição conhecida e confirmando a posição de melhor captação de sinal.

4.4 Circuito e programação

O circuito do sistema de controle funciona basicamente como ilustrado no diagrama de blocos da Figura 26:

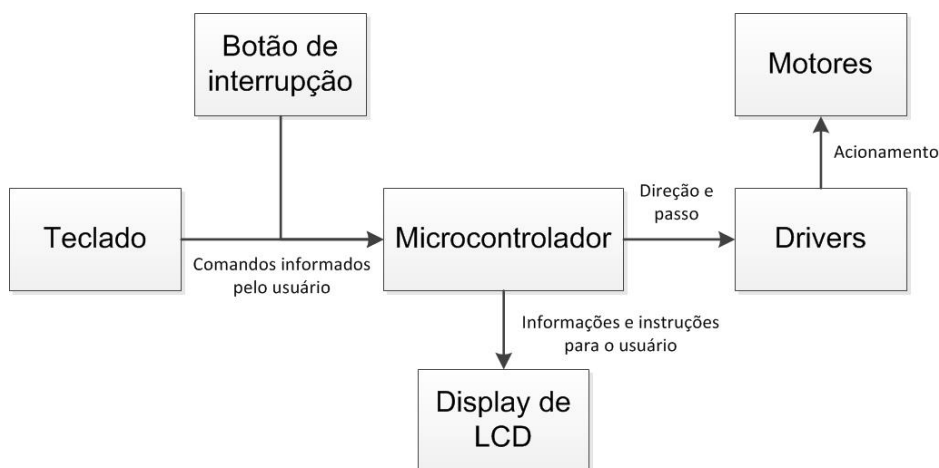


Figura 26 - Diagrama de blocos do sistema de controle.

O usuário entra com comandos através dos botões disponíveis no teclado ou o botão de interrupção, para zerar o posicionamento. O microcontrolador exibe no display de LCD as informações de posicionamento e as opções que o usuário pode selecionar, e, seguindo os comandos recebidos, ativa os drivers que por sua vez acionam os motores.

Um esquema mais detalhado do circuito pode ser encontrado no Apêndice F. A programação na íntegra pode ser encontrada no Apêndice A e uma explicação mais detalhada dos registradores configurados pode ser encontrada no Apêndice B.

4.4.2 Variáveis globais

As variáveis globais utilizadas são:

- a) sat: indica o satélite em questão nas telas de seleção e gravação de posições pré-configuradas. O valor 0 indica a posição atual.
- b) x: posição no eixo horizontal.
- c) y: posição no eixo vertical.
- d) x1: exibe o valor de x no LCD.
- e) y1: exibe o valor de y no LCD.
- f) tecla: inicializada como 0, recebe o valor de teclado() para indicar qual tecla foi pressionada.
- g) m: indicador que indica se o movimento é azimutal (1) ou de elevação (0) para a função digitar().
- h) intrrpt: inicializada como 0, faz com que a função de zeramento seja acionada.

4.4.3 Função main()

A função principal do programa basicamente direciona o usuário através das telas e funções que efetivamente movimentam a antena e alteram as variáveis. Seguindo os seguintes fluxogramas, apresentados nas Figuras 27 e 28:

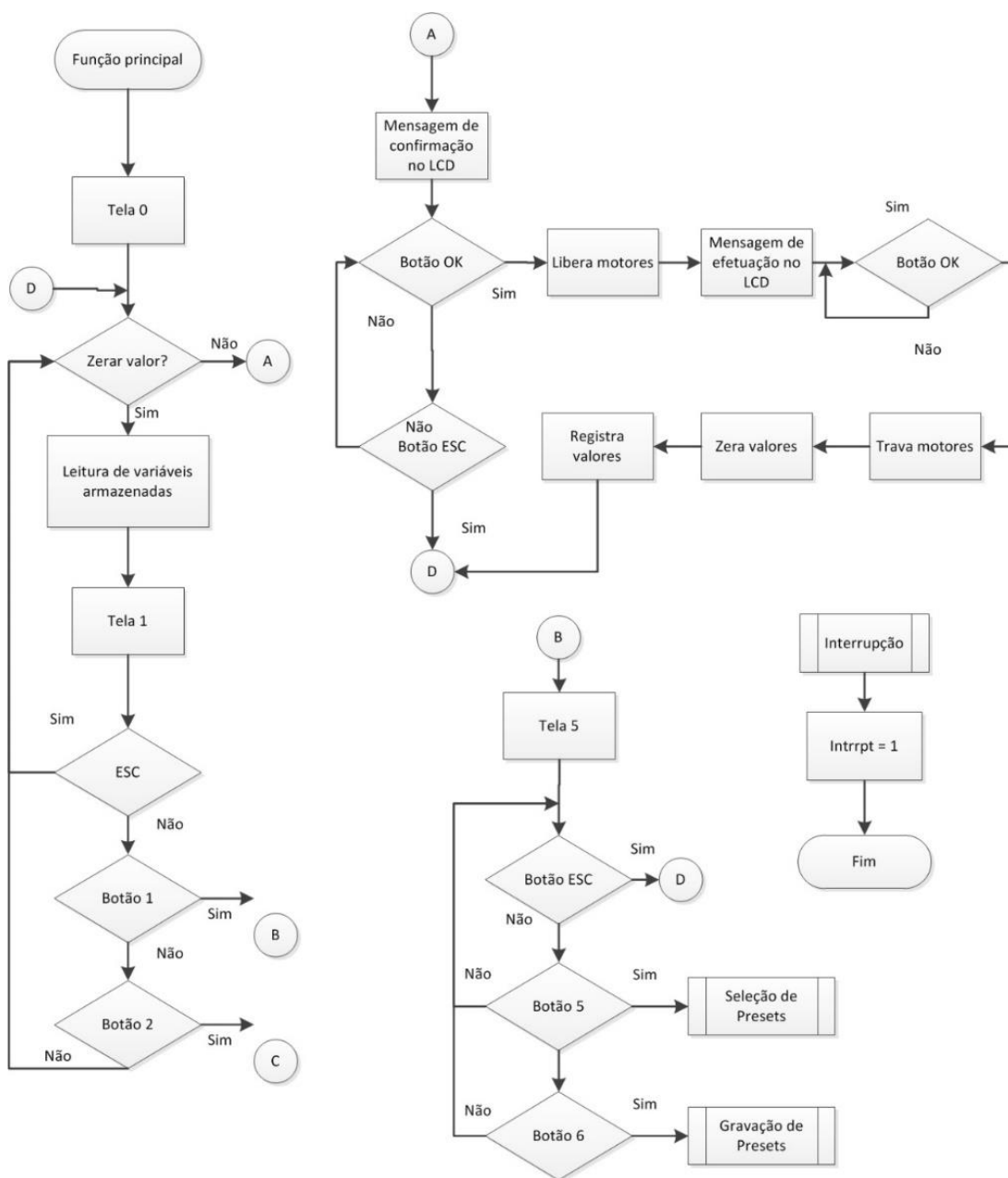


Figura 27 - Diagrama de blocos da função main()

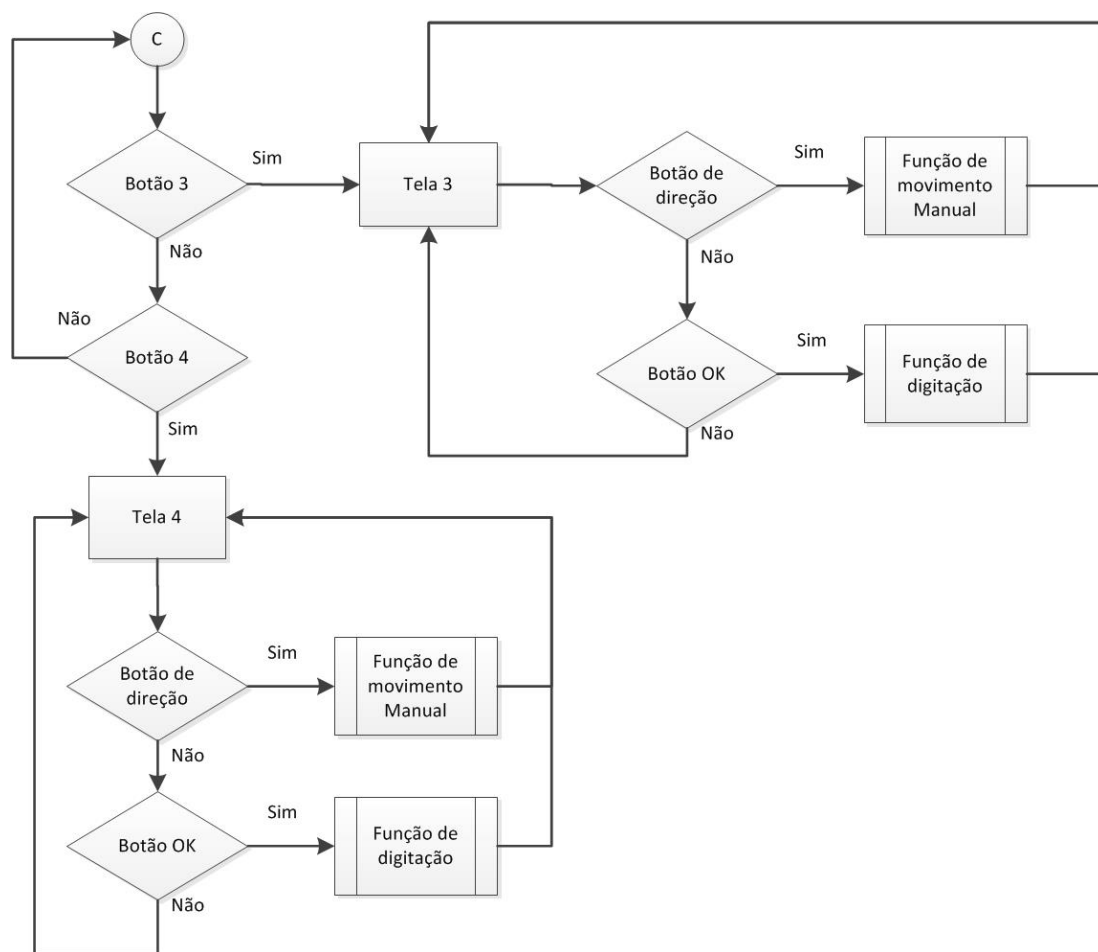


Figura 28 - Diagrama de blocos da função main (cont.)

A função começa com as configurações necessárias do microcontrolador: declaração de variáveis e definição dos pinos como entrada ou saída, e em seguida inicializa o LCD, limpando sua tela e desabilitando o cursor. A primeira tela exibida é a “tela 0” (Figura 29), com o nome do projeto e autores. Em seguida o programa entra em um loop infinito e pode entrar em um de dois “while”: `intrrpt = 0` ou `intrrpt = 1`. No primeiro caso, a interrupção INT0 não foi acionada, então o programa procede para sua funcionalidade normal, já no segundo caso, a interrupção ocorreu, então o programa realiza a rotina de zeramento (função A no fluxograma).

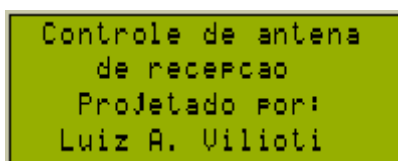
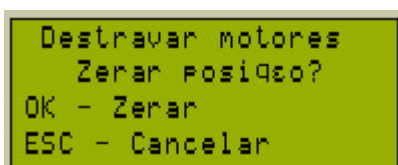


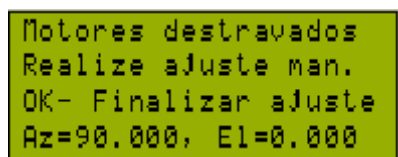
Figura 29 - Tela 0

Na rotina de zeramento o usuário recebe uma mensagem de confirmação no LCD (Figura 30), se ele confirmar, os motores são liberados para movimentação e os valores de azimuth e elevação são alterados respectivamente para 90° e 0°. É esperado que o usuário movimento os motores manualmente para essas posições e então pressione OK na controladora, na tela seguinte (Figura 31), para zerar a variável intrrpt e retornar ao programa normal. Esse processo é ilustrado pelas Figuras 25 e 26.



```
Destravar motores  
Zerar Posicao?  
OK - Zerar  
ESC - Cancelar
```

Figura 30 - Tela de confirmação de zeramento.

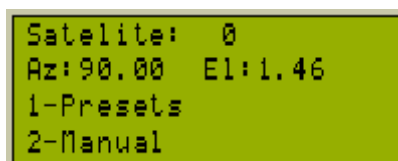


```
Motores destravados  
Realize ajuste man.  
OK- Finalizar ajuste  
Az=90.000, El=0.000
```

Figura 31 - Tela de efetuação do zeramento.

Quando intrrpt = 0, o programa primeiro lê as variáveis armazenadas nas posições 0 (*azimuth*) e 4 (*elevação*) da memória EEPROM, ajusta-as para a exibição no LCD, e exibe a tela 1 (Figura 32). Em seguida o programa fica preso em uma função “while” até que o usuário pressiona algum dos seguintes botões do teclado: 1, 2 ou ESC.

Nesse ponto do programa e em outros com a mesma estrutura, para acionar a interrupção é necessário, além de pressionar o botão de zeramento, pressionar a tecla ESC para sair da lógica circular (*looping*) de teste o teclado.



```
Satelite: 0  
Az:90.00 El:1.46  
1-Presets  
2-Manual
```

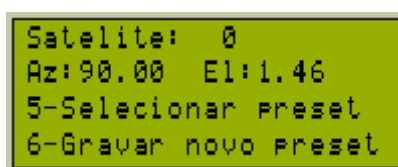
Figura 32 - Tela 1.

Quando alguma das três teclas é pressionada, o programa realiza testes para verificar qual delas foi acionada.

Se foi a tecla ESC, o programa sai do *looping*, realiza os testes seguintes, imprime a tela novamente e volta para o *looping* do teclado. Essa situação é útil, como foi explicado, para o caso da interrupção ter sido acionada.

Se foi a tecla 1, o programa vai para o menu de posições pré-configuradas, a seção “B” no fluxograma.

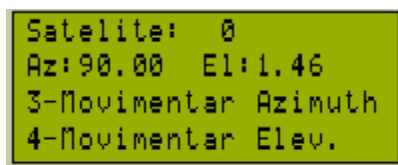
Nesse caso o programa exibe a tela 5 (Figura 33) e entra no *looping* do teclado testando as teclas ESC, 5 e 6. Caso alguma delas seja pressionada, o programa realiza os testes e, se for a tecla 5, vai para a função de seleção de posição pré-configurada, caso seja a tecla 6, vai para a função de gravar uma nova posição. Essas funções serão explicadas nas seções seguintes.



```
Satelite: 0
Az:90.00 El:1.46
5-Selecionar preset
6-Gravar novo preset
```

Figura 33 - Tela 5.

Se tecla 2, o programa vai para a parte descrita no fluxograma 2, função “C”. A tela exibida é a tela 2, apresentada na Figura 34, e as escolhas do usuário são a de movimentar o *azimuth* (tecla 3) ou a elevação (tecla 4) manualmente. Dependendo de qual opção for selecionada, a variável de indicação *m* é alterada, para depois ser utilizada na função de digitar.



```
Satelite: 0
Az:90.00 El:1.46
3-Movimentar Azimuth
4-Movimentar Elev.
```

Figura 34 - Tela 2.

Essas duas opções têm a mesma estrutura: exibe a tela apropriada (tela 3 ou tela 4, mostradas na Figura 35), entra em um *looping* de teclado testando se o usuário

pressiona ESC, para sair da função, OK, para ir para a função de digitação de novo valor (explicada nas próximas seções), ou alguma das teclas de seta (para os lados, no caso do *azimuth*, ou para cima e para baixo, no caso de elevação).

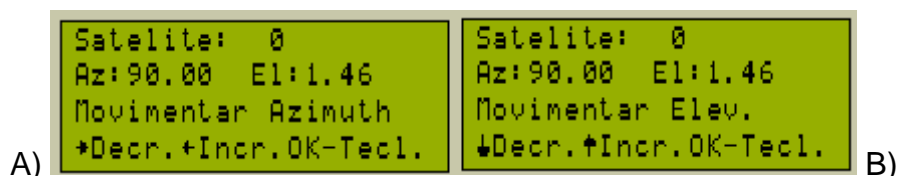


Figura 35 - Tela 3 (A) e tela 4 (B).

Caso o usuário pressione alguma das teclas de seta, o programa vai para a função de movimentação manual de *azimuth* ou elevação, e o valor da tecla pressionada que informa a essa função a direção em que será movimentado o respectivo motor.

Uma vez executada a movimentação, o novo valor é armazenado na memória

4.4.4 Circuito e programação do LCD

O circuito do LCD, ilustrado na Figura 36, é o que permite ao usuário interagir com o equipamento, fornecendo as informações de que botão apertar e os valores de elevação e azimuth. Esse circuito utiliza a ligação de quatro bits de comunicação com o PIC e é acionado via funções da biblioteca de LCD do compilador. Para utilizar essa biblioteca é necessário primeiro associar quais pinos do PIC estão ligados a quais pinos do LCD na programação definindo as variáveis LCD_RS, LCD_EN, LCD_D7, LCD_D6, LCD_D5 e LCD_D4. Isso foi feito na seção “Pinagem do LCD” da programação.

No circuito também foi utilizado um divisor de tensão no pino VEE do LCD responsável pelo brilho dos caracteres e ainda a ligação nos pinos A e K para deixar o led de fundo ligado.

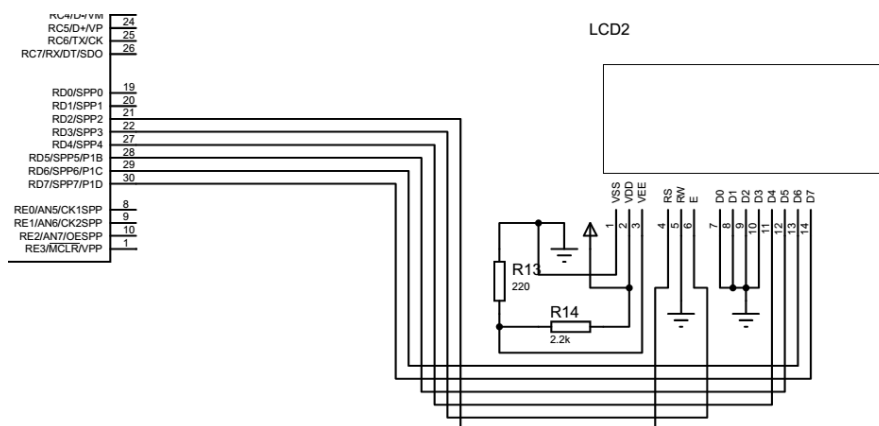


Figura 36 - Circuito do LCD.

Foram utilizados também quatro caracteres especiais que precisam ser construídos na ferramenta de edição de caracteres para LCD do compilador, a LCD Custom Char Generator. Nessa ferramenta é possível criar um caractere para uma unidade de espaço do LCD e gerar automaticamente sua função automaticamente e matriz a ser declarada. Esse processo é ilustrado pela Figura 37.

Ao criar mais de um caractere especial é necessário alterar os valores dos campos Lcd_cmd(72) e Lcd_Chr(pos_row, pos_char, 1) , além de alterar no nome “CustomChar”.

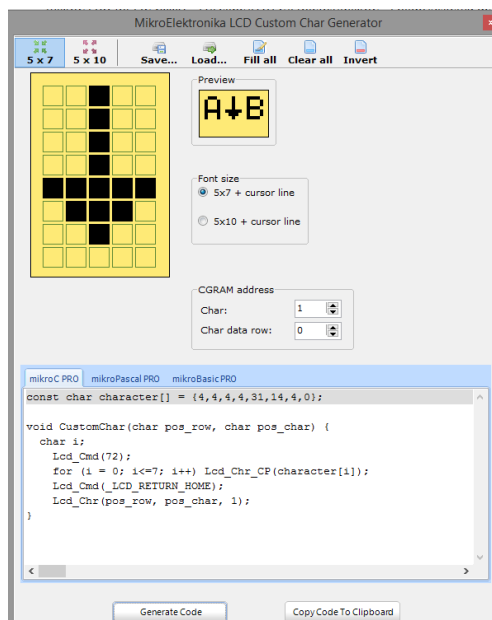


Figura 37 - LCD Custom Char Generator.

4.4.5 Teclado Analógico

A interação do operador com o circuito é realizada através de um teclado com 16 teclas (ESC, OK, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, seta para esquerda, seta para direita, seta para baixo e seta para cima). Para economizar portas de entrada/saída do PIC foi utilizado um modelo de teclado analógico que consiste em uma combinação de botões que altera a resistência de um divisor de tensão ligado a uma entrada analógica do PIC. Esse circuito pode ser visto na Figura 38.

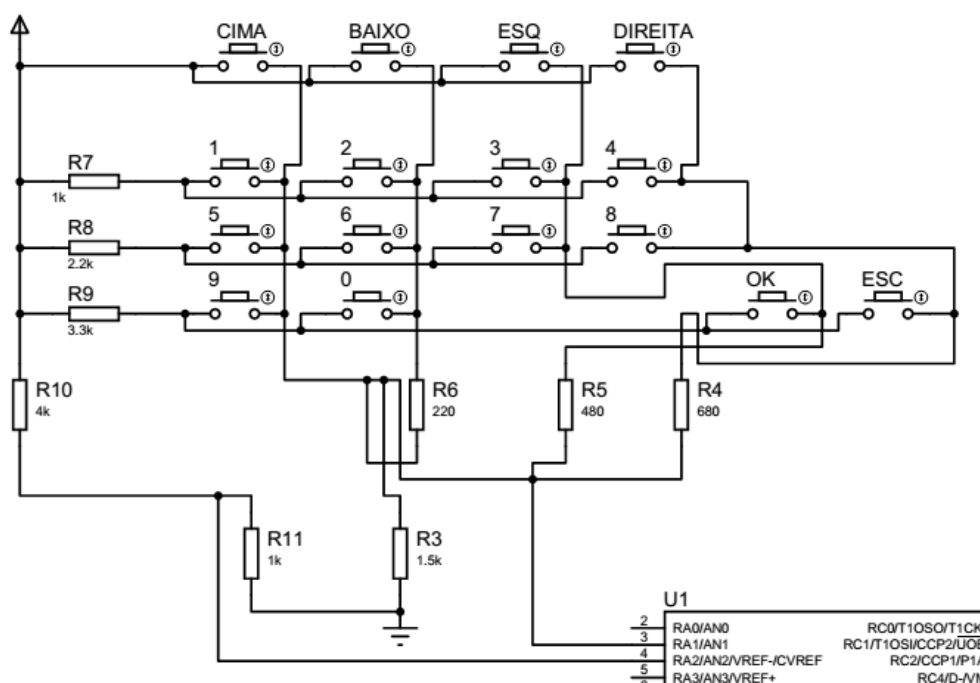


Figura 38 – Circuito do Teclado Analógico.

As tensões obtidas a partir desse teclado variam de 1,37V a 5V, utilizando valores calculados. Para melhorar a precisão do conversor em vez de utilizar o 0V como referência foi aplicada uma tensão de 1V na entrada do PIC Vref- (pino 4, o canal analógico 2).

Cada tecla pressionada informa um valor de tensão para o conversor analógico-digital do PIC, esse valor de tensão é convertido para um valor digital em uma escala de 1024. Para considerar possíveis variações de tensão causadas pela velocidade com que o botão é pressionado ou pelo material utilizado na matriz de contato e placa

de circuito impresso, foram definidos intervalos que cobrem esse valor digital. Essas relações podem ser vistas nas Tabelas 4 e 5, que informam os valores calculados e medidos na matriz de contato.

Tecla	Função	Tensão Calculada (V)	Valor digital	Maior que	Menor que
16	Cima	5	1023	941,5	
15	Baixo	4,360465116	860	786,5	941,5
14	Esquerda	3,787878788	713	669	786,5
13	Direita	3,440366972	625	568,5	669
12	1	3	512	481	568,5
11	2	2,757352941	450	419	481
10	3	2,516778523	388	368	419
09	4	2,358490566	348	305,5	368
08	5	2,027027027	263	248,5	305,5
07	6	1,913265306	234	219	248,5
06	7	1,794258373	204	193,5	219
05	8	1,712328767	183	163,5	193,5
04	9	1,5625	144	135,5	163,5
03	0	1,494023904	127	117,5	135,5
02	Enter	1,420454545	108	101,5	117,5
01	ESC	1,368613139	95	47,5	101,5
	Vref-	1	0		

Tabela 4 - Valores calculados para o teclado analógico.

Tecla	Função	Tensão medida (V)	Valor digital	Maior que	Menor que
16	Cima	4,48	1023	938,5	
15	Baixo	3,92	854	787	938,5
14	Esquerda	3,48	720	667	787
13	Direita	3,13	614	550	667
12	1	2,71	486	452,5	550
11	2	2,49	419	390,5	452,5
10	3	2,30	362	337,5	390,5
09	4	2,14	313	281	337,5
08	5	1,93	249	232,5	281
07	6	1,82	216	201	232,5
06	7	1,72	186	170,5	201
05	8	1,62	155	137	170,5
04	9	1,50	119	108,5	137
03	0	1,43	98	88,5	108,5
02	Enter	1,37	79	70	88,5
01	ESC	1,31	61	30,5	70
	Vref-	1,11	0		

Tabela 5 - Valores medidos na protoboard para o teclado analógico.

Esses valores tabelados foram encontrados utilizando as seguintes fórmulas:

Resolução:

$$res = \frac{V_{cima} - V_{ref-}}{1023} = \frac{5-1}{1023} = 0,003910068$$

Tensão calculada (V):

$$V_{calc} = \frac{V_{cima} * R_3}{((R1_{botão} + R2_{botão}) + R_3)}$$

Onde $R1_{bot\tilde{a}o}$, é igual a 0Ω , R7, R8 ou R9, e $R2_{bot\tilde{a}o}$ é igual a 0Ω , R4, R5 ou R6, dependendo do botão pressionado, conforme a tabela a seguir:

		$R2_{bot\tilde{a}o}$			
		-	R6	R5	R4
$R1_{bot\tilde{a}o}$		0	220	480	680
-	0	CIMA	BAIXO	ESQ	DIR
R7	1k	1	2	3	4
R8	2.2k	5	6	7	8
R9	3.3k	9	0	OK	ESC

Tabela 6 - Relação de botões e resistores.

Valor digital:

$$d = \frac{V_{calc} - V_{ref}}{res}$$

Arredondado para cima.

O intervalo de valor digital utilizado para o PIC reconhecer um botão é calculado tirando a média do valor nominal do botão com o valor seguinte ou anterior. Por exemplo:

$$d_4 \leq d_{4a}, \quad d_{4a} = \frac{d_5 + d_4}{2} = \frac{348 + 263}{2} = 305,5$$

$$d_4 \geq d_{4b}, \quad d_{4b} = \frac{d_4 + d_3}{2} = \frac{348 + 388}{2} = 368$$

Na programação, a leitura desse teclado é realizada pela função teclado(), que retorna um valor char variando de 1 (representando a tecla ESC) à 16 (tecla CIMA), ou o valor 0 quando não há tecla alguma sendo pressionada

A função teclado() funciona de acordo com o fluxograma apresentado na Figura 39:

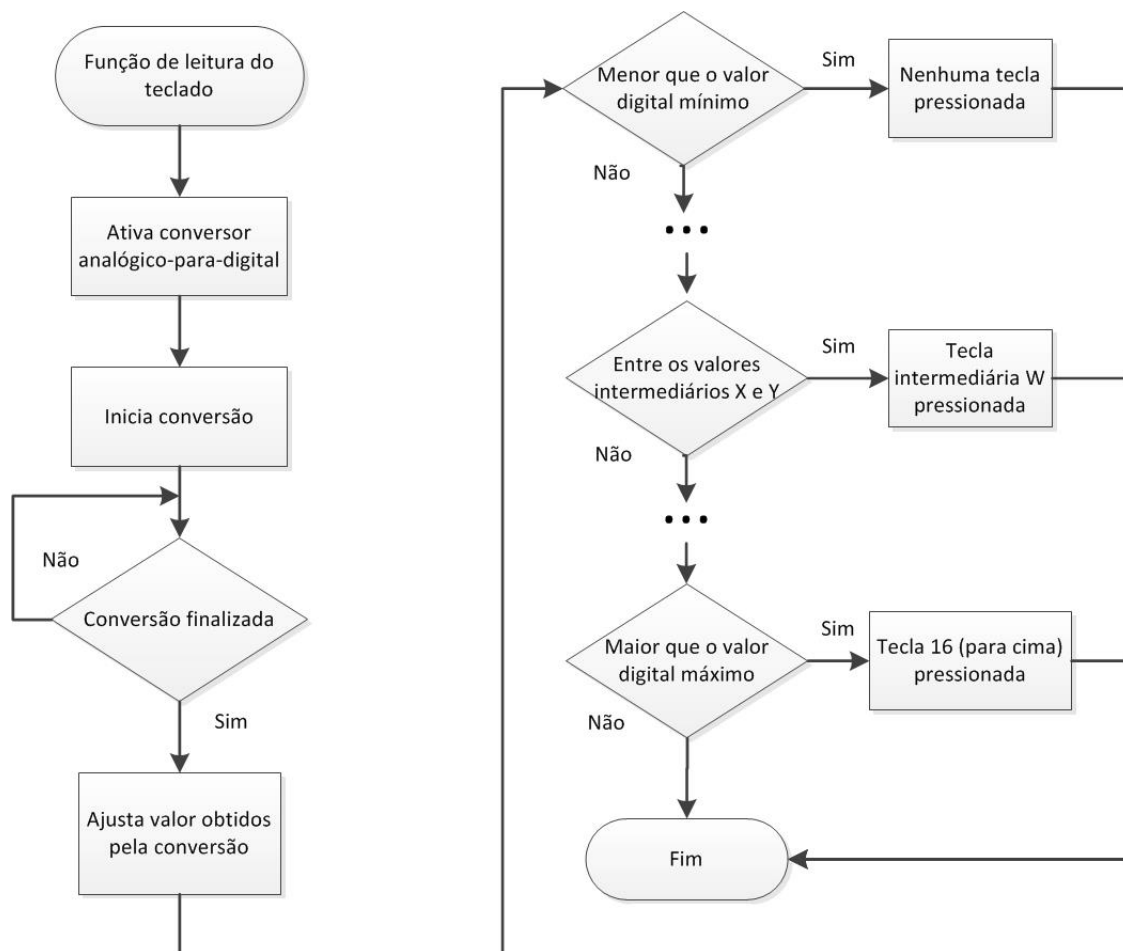


Figura 39 - Diagrama de blocos da função de leitura do teclado.

A função começa declarando as variáveis que serão utilizadas (vanalog e tecla), seta os bits necessários para ativar o conversor analógico-digital, inicia a conversão, espera a conversão acabar e então registra o valor digital obtido na conversão na variável vanalog, juntando os registradores ADRESH e ADRESL. Em seguida, vanalog é testado para ver em que intervalo de valores o resultado se encontra: se o teste é verdadeiro, o valor de tecla correspondente é registrado na variável tecla e o valor desta é retornado.

4.4.6 Lógica circular de teste do teclado

O programa do sistema como um todo repete basicamente a seguinte estrutura de três etapas: exibe uma informação no LCD, verifica o teclado e realiza a ação escolhida pelo usuário. De início essa estrutura apenas ficaria dentro de uma função circular, um *looping* como a função “while”, porém apenas com isso ela repetiria a exibição da tela e o teste de qual ação foi escolhida sem necessidade.

Para que isso não aconteça foi utilizada com frequência a seguinte estrutura para verificar o teclado:

```
while(tecla != x){
    tecla=teclado();
}
```

Essa função testa repetidamente o teclado até que a variável “tecla” receba o valor de “x”.

Por exemplo, na função main():

```
tela 1 (sat, x1, y1);
while (tecla != 12 && tecla != 11 && tecla != 1){
    tecla = teclado ();
}
if(tecla==12){
    ...
}
If(tecla==11){
    ...
}
```

Esse trecho exibe a tela 1, que pergunta se o usuário quer selecionar ou gravar um preset ou realizar o ajuste manual e em seguida fica preso no sentença “tecla =

teclado()” até que o usuário escolha umas das opções: ESC para sair (tecla=1), 1 para preset (tecla=12) ou 2 para o ajuste manual (tecla=11).

Se não fosse utilizada essa estrutura o programa imprimiria a tela várias vezes seguidas, fazendo-a piscar, e realizaria os testes das ações repetidamente, de forma que o usuário poderia apertar um botão rapidamente enquanto esses testes são realizados e ele não seria identificado.

Ainda há um looping parecido utilizado como trava-teclas:

```
while(tecla!=0){
    tecla=teclado();
}
```

Nesse caso, após uma ação ser escolhida o programa fica preso em uma repetição do teste do teclado até que o usuário solte a tecla pressionada (ou seja, “tecla” recebe o valor 0, nenhuma tecla está pressionada).

4.4.7 Controle dos motores e funções de movimento manual

Para controlar os motores o PIC envia apenas três sinais para os drivers: pulso, direção e *enable*. O último, quando em nível alto, destrava os eixos dos motores, permitido que eles sejam movimentados manualmente. Essa função é utilizada na interrupção de zeragem do programa.

O sinal de pulso faz com que o motor dê um passo e o sinal de direção escolhe a direção em que ele se movimenta.

A função de pulso é bastante simples e apenas envia sinal alto por um milissegundo para o pino apropriado do PIC quando é acionada. No caso do motor azimutal, o pino é o RC2 (17), e no caso do motor de elevação o pino é o RB3 (36).

Inicialmente era planejado ligar o driver do motor de elevação nos pinos restantes do PORTC do PIC. Isso não foi feito porque os pinos RC4 e RC5 (23 e 24) são exclusivos para entrada de dados (D+ e D-) e não podem ser utilizados como saídas, por isso essa ligação foi feita no PORTB.

O passo escolhido para o motor e ajustado nas chaves manuais do driver é o de $0,1125^\circ$ ($1/16$), para a maquete montada para demonstração essa precisão e torque são o bastante.

A função de movimento manual e a informação de pulso para o motor funcionam de acordo com os diagramas mostrados na Figura 40.

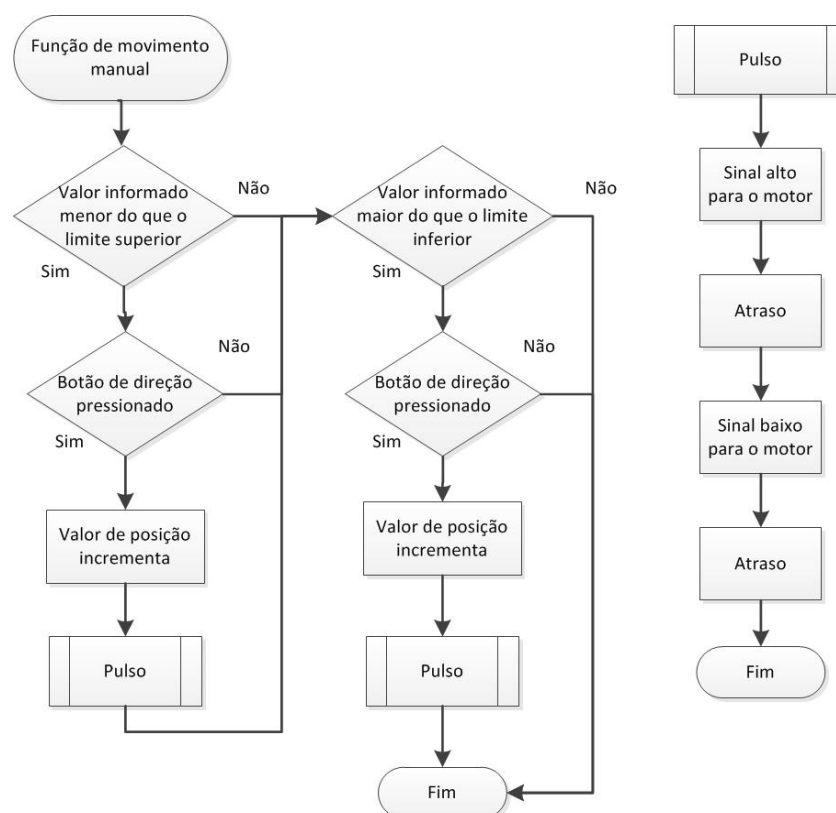


Figura 40 - Diagrama de blocos da função de movimentação manual.

Ambas as funções para movimentação manual de *azimuth* e elevação têm a mesma estrutura de funcionamento. Elas são utilizadas quando o usuário escolhe alterar a posição azimutal ou de elevação manualmente, pressionando as teclas da seta para a direita (13) ou da seta para a esquerda (14) ou para baixo (15) ou para cima (16). Quando a tecla 13, por exemplo, é pressionada, o valor de x é subtraído de 0,1125 para ser exibido no LCD, a saída do PIC ligada ao pino de direção do driver é setado como 0, e um pulso é dado.

Essa função também é utilizada dentro da função `digitar()`, que efetivamente simula o botão de direção apropriado estar sendo “segurado” até o valor digitado pelo usuário ser atingido.

4.4.8 Função `digitar()`

Essa função funciona informando ao usuário qual casa decimal digitar por vez (centena, dezena, unidade, decimal 1 e decimal 2) e indo para um *looping* de teclado. Quando o usuário digita um número, esse número é ajustado para a casa decimal apropriada ($x \cdot 100$, para centena, por exemplo) o programa pede que seja digitada a próxima casa, ou volta para a centena caso a última casa decimal já tenha sido digitada. Nessa sequência, o usuário ainda pode pressionar ESC para cancelar a mudança e voltar à função anterior ou OK para aceitar um novo valor. Um exemplo é mostrado na Figura 41.

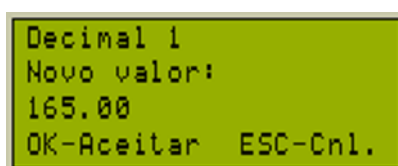


Figura 41 - Tela de digitação.

O teste de “Realizar Movimentação” descrito no fluxograma da Figura 43 aparece na forma de uma variável indicadora na programação, a variável “func”. Caso essa função tenha sido chamada dentro da função `main()`, “func” faz com que OK passe para a movimentação automática da antena. Caso essa função tenha sido chamada dentro da função de gravação de posições, o botão OK apenas retorna o valor digitado, sem realizar a movimentação.

No primeiro caso, quando pressionado o botão OK, é realizado um teste para verificar se o novo valor é menor que o antigo. Se sim, é realizado um teste para ver se o valor em questão é de *azimuth* ou elevação utilizando a variável “m” descrita na função `main()` e a movimentação é realizada com a função de movimentação manual e o programa definindo que a tecla apropriada para decrementar a posição (seta para a direita ou seta para baixo) foi pressionada, até que o valor da antena seja igual ao valor informado.

Se o valor é maior que o antigo, o mesmo processo é realizado, porém as teclas simuladas pelo programa são as que incrementariam a posição (seta para a esquerda ou para cima).

Ao realizar a movimentação a tela exibida é uma das mostradas na Figura 42, nas quais os respectivos valores são atualizados a cada passo e a reticência da palavra “movimentando” acompanha mudando a quantidade de pontos.



Figura 42 - Tela indicando a movimentação.

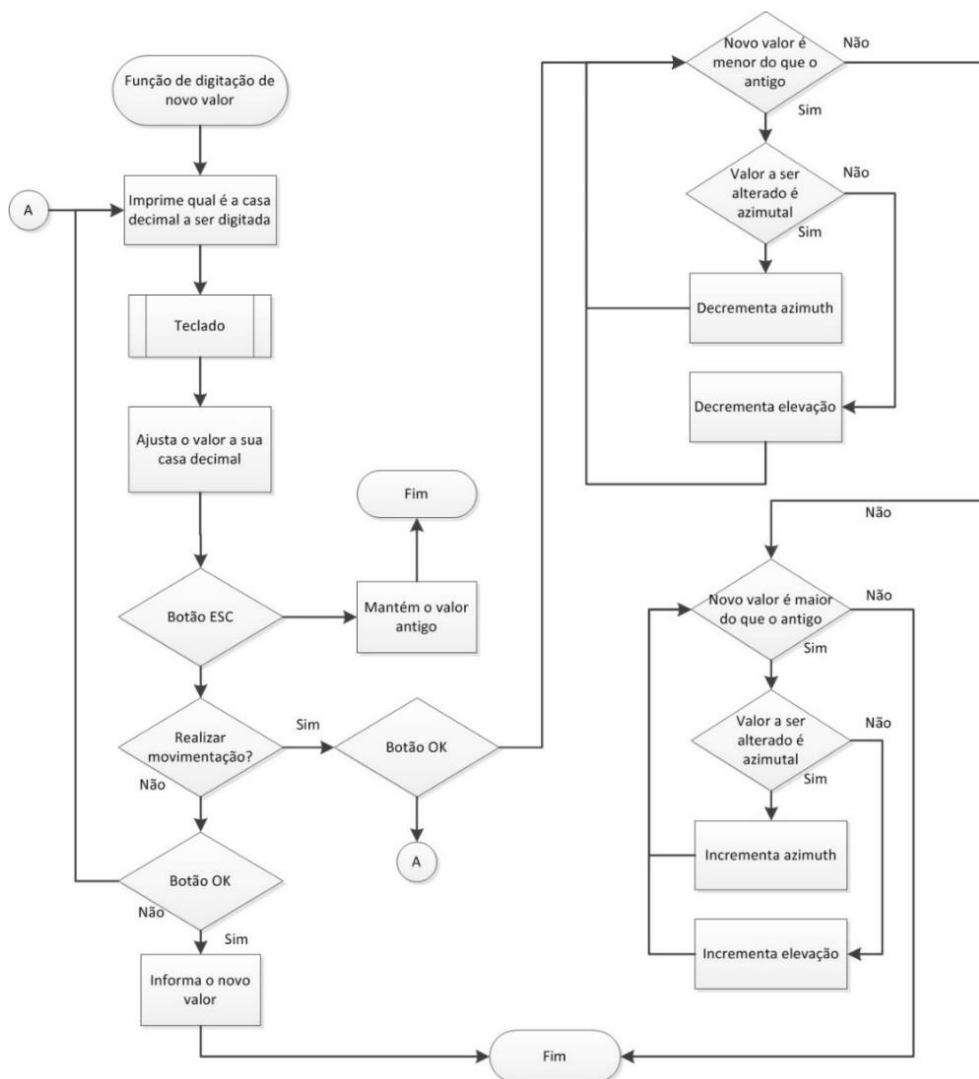


Figura 43 - Diagrama de blocos da função de digitação de novo valor.

4.4.9 Memória EEPROM

A memória EEPROM do PIC é utilizada para armazenar os valores de *azimuth* e elevação atuais e os valores das posições pré-configuradas. Os valores de *azimuth* e elevação são variáveis *float*, ou seja, possuem 4 bytes. Dessa forma a memória EEPROM do PIC é capaz de armazenar 32 conjuntos de *azimuth* e elevação, 31 posições gravadas mais a posição atual (0).

O valor atual de *azimuth*, que é modificado manualmente ou via digitação, é armazenado na posição inicial 0, e por ter 4 bytes ocupa as posições 0, 1, 2 e 3 da memória. O valor de elevação é armazenado na posição inicial 4 e ocupa também as três posições subsequentes.

A variável *sat* é responsável por apontar para essas posições segundo as seguintes fórmulas:

$$\begin{aligned} \text{posição inicial de azimuth} &= \text{sat} * 8 \\ \text{posição inicial de elevação} &= (\text{sat} * 8) + 4 \end{aligned}$$

Dessa forma, a posição inicial de azimuth de *sat* 0 (posição atual) é 0 ($0 * 8$), e a de elevação é 4 ($0 + 4$), as posições iniciais de *sat* 23 são 184 e 188, e as de *sat* 31, 248 e 252.

A biblioteca do compilador conta com uma função para ler um valor na memória EEPROM e uma função para escrever, porém elas lidam com apenas 1 byte de informação, sendo que as variáveis a serem armazenadas nesse programa contam com 4 bytes de informação.

Para realizar a operação necessária, foram adaptadas duas funções disponibilizadas no site helmpcb.com [13], citado na bibliografia, e que estão sob o comentário “Funções de leitura/gravação de variáveis *float* na EEPROM” na programação no Apêndice A.

Essas funções lidam com o conceito de “ponteiro”, ou pointer, que são objetos de programação especiais que “apontam” para endereços da memória de programa flash em que uma variável comum está armazenada. Em relação aos ponteiros são utilizados dois operadores: “&” e “*”, sendo que “&” indica um endereço e “*” permite obter a informação contida naquele endereço.

Dessa forma, quando se declara as seguintes sentenças:

```
float result;  
char *ptr= (char *)&result;
```

“ptr” é um pointer que indica o endereço de memória do primeiro byte da variável “result”. “ptr+1” indicaria o segundo e assim por diante.

A função `Eeprom_Read_Float` utilizada nesse programa basicamente lê os valores armazenados em uma posição da memória EEPROM utilizando a função da biblioteca do MikroC e usa um pointer para armazená-los em um endereço da memória flash. Ao realizar essa operação dentro de uma função for quatro vezes seguidas, ela obtém os quatro bytes que compõe uma variável *float*.

O mesmo princípio se aplica à função `Eeprom_WriteFloat`.

4.4.10 Função de seleção de posições pré-definidas

A função de seleção de posições pré-definidas permite ao usuário selecionar um conjunto de posição de azimuth e de elevação pré-definido dentre vários. Ela utiliza os princípios descritos nas seções anteriores sobre a memória EEPROM e a utilização da variável “sat” como indicador. Essa função segue o fluxograma apresentado na Figura 44:

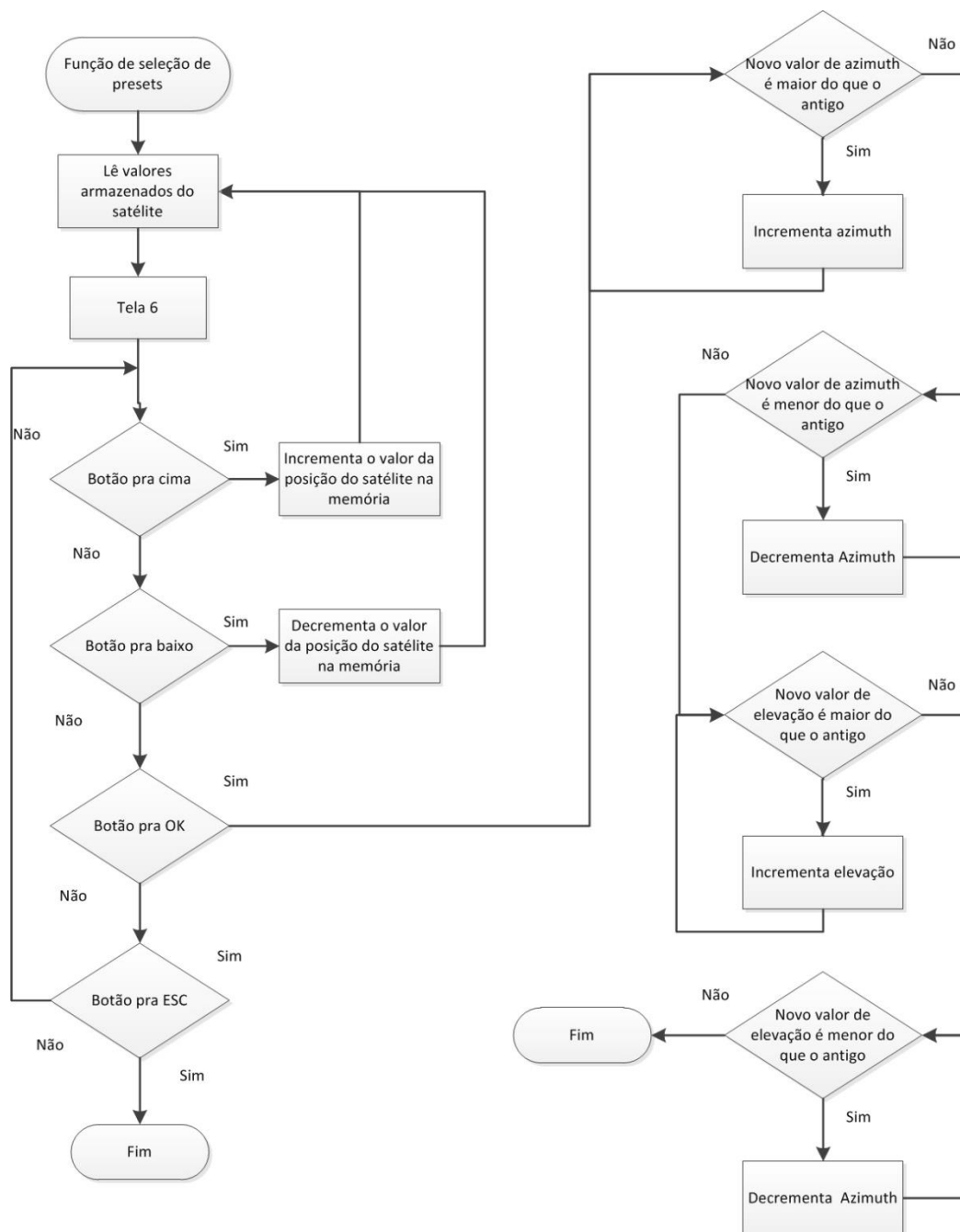


Figura 44 - Diagrama de blocos da função de seleção de posições pré-definidas.

A função lê os valores armazenados nas posições atreladas à sat = 0 e imprime a tela 6, mostrada na Figura 45, para então entrar em um *looping* de teclado que testa as teclas “para cima”, “para baixo”, OK e ESC. Se as teclas “para cima” ou “para baixo” forem pressionadas o programa incrementa ou decrementa o valor de sat e realiza a leitura das variáveis armazenadas na posição relacionada ao novo valor de sat.

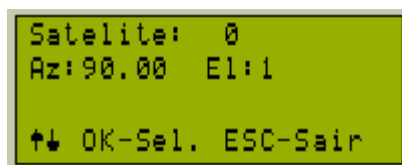


Figura 45 - Tela 6.

Se a tecla OK é pressionada, inicia-se a movimentação para as posições selecionadas. Primeiro no eixo azimutal: se o novo valor for maior do que o antigo, incrementa-se a posição atual, e se for menor, decrementa-se, e depois no eixo de elevação, realizando a mesma sequência. A tela neste caso é apresentada na Figura 46.

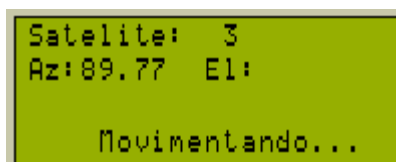


Figura 46 – Tela indicando a movimentação.

4.4.11 Função de gravação de posições pré-definidas

A função de gravação permite ao operador gravar uma nova posição pré-definida relacionada a um dos 31 satélites. Funcionando de acordo com o seguinte fluxograma, apresentado na Figura 47:

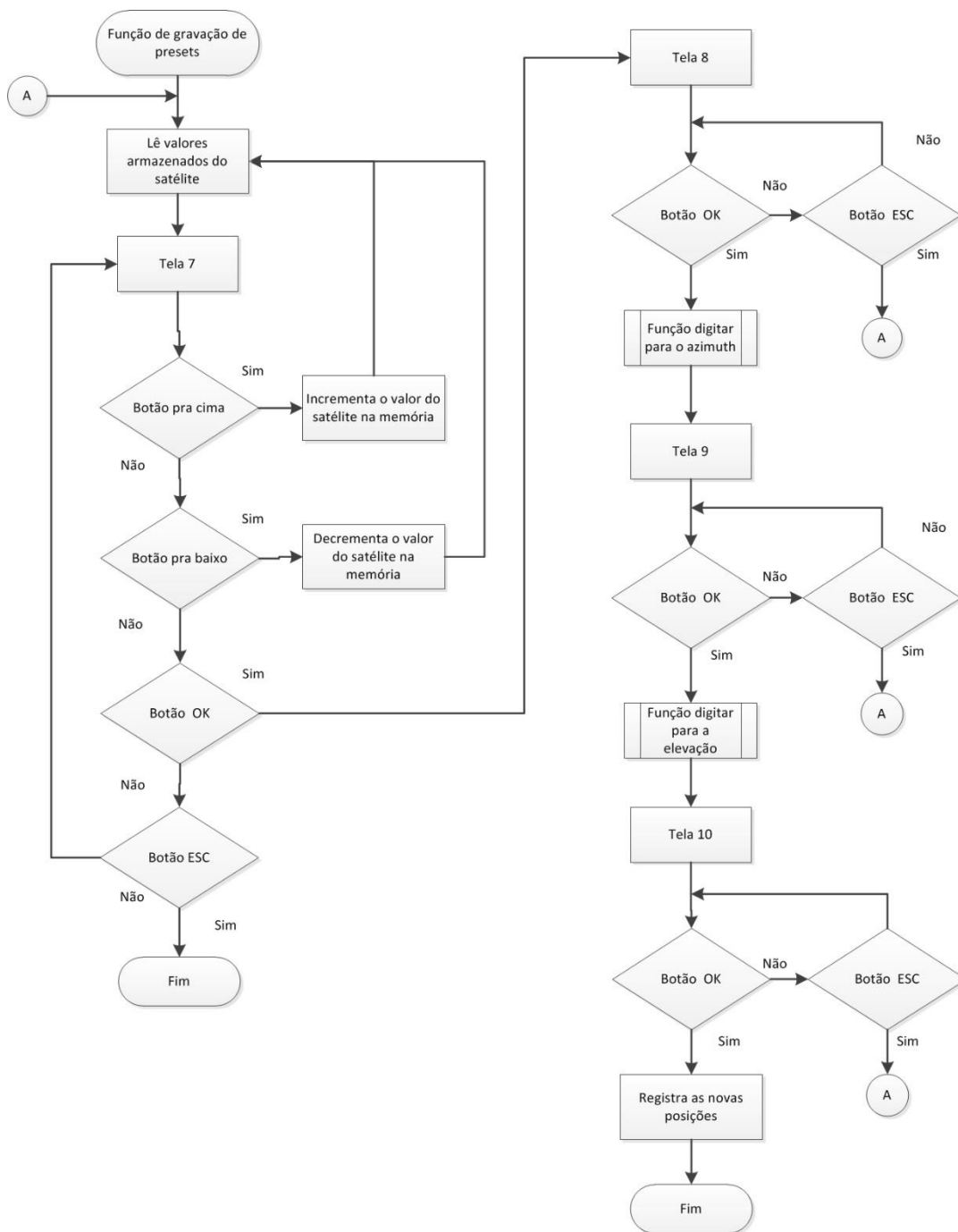


Figura 47 - Diagrama de blocos da função de gravação.

O início da função de gravação é o mesmo da função de seleção: alterar o valor da variável sat para escolher duas posições de memória correspondentes, com a diferença da opção de confirmação disponível: selecionar satélite. Isso acontece na tela 7 (Figura 48).

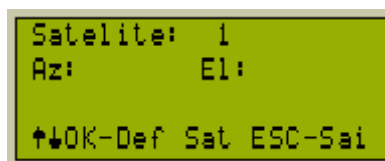


Figura 48 - Tela 7.

Selecionando o satélite ao pressionar o botão OK, o usuário é guiado pelas telas para digitar os valores desejados de *azimuth* e de elevação, mostradas nas Figuras 49 e 50:.



Figura 49 - Tela 8 (A) e tela de digitação (B).

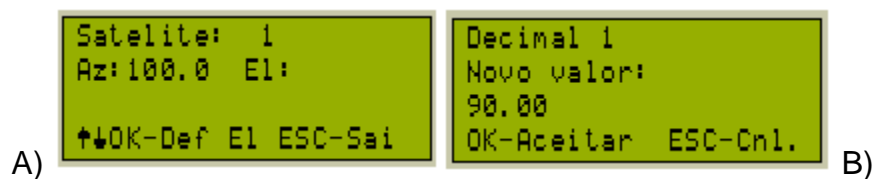


Figura 50 - Tela 9 (A) e tela de digitação (B).

E em seguida o usuário é perguntado se deseja confirmar a gravação da nova posição, para que os valores sejam gravados na memória EEPROM. Mostrado na Figura 51:

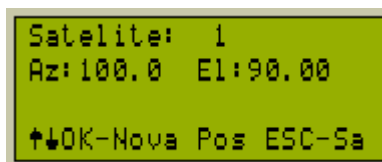


Figura 51 - Tela 10.

5. Resultados

Para a realização dos testes foram utilizados dois metros de cabo para ligar os drivers que ficam dentro de uma caixa, que estaria dentro de um prédio, e os motores acoplados à antena, que ficariam ao ar livre. Em um caso real, seria necessário também proteger os motores contra a umidade.

Segundo a especificação dos motores utilizados, estes deveriam consumir uma potência de 8,82W a 2,1A/4,2V por bobina. Os drivers nominalmente possuem corrente máxima de 3,5 A, sendo que utilizando a configuração para corrente de 2 A a potência resultante deveria ser próxima da informada ($2\text{ A} \times 2\ \Omega = 4\text{ V}$, $4\text{ V} \times 2\text{ A} = 8\text{ W}$). No entanto, medindo a corrente em série com a bobina do motor e a saída do driver, obteve-se um valor muito abaixo da corrente nominal, fato que provavelmente ocorreu por alguma limitação neste dispositivo.

Quando na configuração máxima (3,5 A), a corrente resultante media em série com a saída do driver e bobina é de apenas 1,2 A, gerando uma tensão de cerca de 2,64 V e uma potência de aproximadamente 3,168 W. Essa potência não foi suficiente para movimentar ou sustentar o conjunto completo da maquete, principalmente no eixo de elevação, uma vez que o centro do peso não fica concentrado próximo ao centro do eixo do motor. Com isto, a movimentação do conjunto não foi impedida, mas ficou bastante limitada. Por tratar-se de um dispositivo comercial, o fornecedor foi contatado para analisar esse comportamento, sendo que não foi obtida resposta até a conclusão deste texto.

Também há de se ressaltar que o motor e a estrutura montada foram escolhidos apenas para demonstração do circuito de controle e por conveniência em relação ao controle de ângulos, não representando o que deveria ser aplicado na realidade. Isso se dá por duas razões: primeiro porque um motor de passo na escala e com o torque necessário para movimentar uma antena motorizada real seria muito caro, e segundo porque, da maneira como a estrutura foi montada, com apenas o eixo do motor sustentando as posições escolhidas diretamente, os motores precisam ficar ligados para que a antena mantenha a posição fixa, consumindo potência desnecessariamente. Uma estrutura mecânica mais elaborada que sustentasse a

posição da antena sem a necessidade de motores acionados, utilizando roscas sem fim como no caso real, seria o ideal.

Testes realizados com os motores desacoplados da estrutura da maquete (sem carga mecânica) mostraram que a movimentação destes nas direções, velocidade (proporcional aos tempos da programação) e precisão (em graus) ocorreu adequadamente e conforme o projetado.

6. Análise de custos do circuito de controle desenvolvido

A análise dos custos dos componentes utilizados mostrou a possibilidade de obtenção de um circuito funcional de controle com custos relativamente baixos e muito mais baratos que um sistema convencional de controle motorizado de antenas. A Tabela 7 apresenta a relação de custos dos componentes utilizados no circuito de controle.

Componentes	Preço total (R\$)	Data da consulta
PIC18F4550	28,00	17/01/2018
LCD	42,42	17/01/2018
Cristal	1,00	17/01/2018
Resistores	0,60	17/01/2018
Capacitores	0,30	17/01/2018
Botões	4,86	17/01/2018
Total	77,18	

Tabela 7 - Custos do circuito de controle

Nesta avaliação não foram incluídos os custos aproximados dos motores de passo nem dos drivers utilizados, pois entendeu-se que o circuito de controle, teoricamente, pode operar com motores e drivers de diversos tamanhos e potências diferentes, escolhidos com as características da antena utilizada. O custo total do projeto e confecção da estrutura foi de R\$ 500,00 e foram utilizadas aproximadamente 300 horas de trabalho para execução.

7. Conclusão e trabalhos futuros

Foi desenvolvido neste trabalho, como objetivo principal, um sistema que permita controlar à distância uma antena motorizada de recepção de sinais de satélite, de maneira a facilitar a operação da mesma em caso de emergência e utilização no dia-a-dia.

O sistema desenvolvido permite, ao guiar um usuário por menus através de uma interface, receber comandos e enviar outros equivalentes para dispositivos de controle dos motores, sendo estes utilizados para realizar a movimentação de uma antena. Para tal, três formas de movimentação foram implementadas: movimentação para posições pré-definidas, movimentação manual ou via digitação de novos valores de posição. A interface desenvolvida também permite gravar diversas posições possíveis para a antena.

A demonstração do funcionamento com uma maquete foi implementada, porém a realização de testes mais apurados não foi possível, uma vez que os drivers utilizados não forneceram a potência mínima necessária para movimentar apropriadamente os motores. Os objetivos iniciais do projeto foram atingidos, o sistema de controle é capaz de guiar um usuário através da interface, receber seus comandos, e enviar os comandos equivalentes para os drivers de controle. Apenas a parte de demonstração com maquete não foi realizada completamente devido à baixa potência dos *drivers* mencionada anteriormente.

Neste caso sugere-se primeiramente a substituição dos *drivers* ou adaptação do circuito para que o mesmo consiga fornecer a potência necessária, melhorando os torques dos motores para permitir a movimentação da antena de demonstração. Além disso, é válido utilizar outros motores mais apropriados para o que seria a estrutura real, como servo-motores ou mesmo motores DC com a posição controlada através do tempo e PWM ou de *resolvers* ou *encoders*, deste de que seja possível obter com estes modelos uma precisão de pelo menos 1°.

Trabalhos em outras áreas de conhecimento voltadas para a mecânica também podem ser realizados no sentido de aperfeiçoar a estrutura montada para demonstração ou até aproximá-la do caso real.

O projeto desenvolvido apresentou limitações de hardware, no presente caso, limitação de potência nos drivers dos motores. Assim, é sugestão deste grupo desenvolver um sistema elétrico/eletrônico de maneira a diminuir as limitações observadas no circuito desenvolvido.

Além disto, sugere-se o desenvolvimento de um sistema de medição de sinais a ser implementado junto ao circuito. Com isto, pode-se diminuir ou mesmo eliminar a necessidade de uso de um analisador de espectro. Esse sistema pode, inclusive, utilizar uma das entradas do conversor analógico-digital do PIC, alimentada por um circuito detector de radiofrequência, de maneira a permitir integração entre o sistema de movimentação com o sistema de medição da intensidade do sinal a ser captado. Para tal, pode ser utilizado o CI LT5534, que realiza a função de detectar potência nas frequências entre 50MHz e 3GHz, e informa uma tensão proporcional de 0V a 3V [14], dependendo de sua configuração.

Referências bibliográficas

- [1] ABTA, “Evolução do Número de Assinantes de TV por Assinatura”, disponível em: http://www.abta.org.br/dados_do_setor.asp Acessado em 02/02/2018.
- [2] SACCHITIELLO, Bárbara, “TV paga perde quase 1 milhão de assinantes em 2017” <http://www.meioemensagem.com.br/home/ultimas-noticias/2018/02/01/tv-paga-perde-quase-1-milhao-de-assinantes-em-2017.html> Acessado em 02/02/2018.
- [3] KRAUS, J.D., Antenas. Guanabara Koogan, 1983.
- [4] Nova Eletrônica, “Como apontar uma antena parabólica usando o Google Maps” <http://blog.novaeletronica.com.br/como-apontar-antena-para-satelite-usando-o-google-maps/> Acessado em 14/02/2018.
- [5] Microwaves101, “Frequency Letter Bands”, Encontrado em: <https://www.microwaves101.com/encyclopedias/frequency-letter-bands> Acessado em 05/12/2017.
- [6] Andrew, “4.5M Earth Station Antennas”, Encontrado em <https://www.stssa.com/Downloads/Products/Antenna/Andrew/4.5m.pdf> Acessado em 14/02/2018.
- [7] New Era Systems, “ANDREW ASC APC 100 ANTENNA CONTROLLER”, encontrado em: <http://newerasystems.net/shop/antennas/andrew-asc-apc-100-antenna-controller/> Acessado em 14/11/2017.
- [8] Microchip Technology Inc., "PIC18F2455/2550/4455/4550 Data Sheet" - 2009 - DS39632E

- [9] BRITES, Felipe Gonçalves; SANTOS, Vinícios P. A. “Motor de passo”, disponível em: <<http://studylibpt.com/doc/1268940/tutorial-motor-de-passo>> Acessado em 15/11/2017

- [10] *Datasheet* do motor de passo AK23/15F6FN1.8 encontrado em: <<http://www.neomotion.com.br/motor-de-passo/>> Acessado em 15/11/2017.

- [11] Manual do *driver* baseado em TB6600 da DFRobots encontrado em: <https://www.dfrobot.com/wiki/index.php/TB6600_Stepper_Motor_Driver_SKU:_DRI0043>Acessado em 15/11/2017

- [12] ORISE Technology Co., Ltd, “Specification For LCD Module 2004A” - 2007

- [13] HELM PCB, “Reading & writing structs, floats and other objects to EEPROM”, disponível em: <http://helmpcb.com/software/reading-writing-structs-floats-and-other-objects-to-eprom>>, acessado em 20/11/2017

- [14] Linear Technology Corporation, “LT5534” – 2004 – LT 1210 Rev. C.

Apêndice A

Nesse apêndice é apresentado o código fonte utilizado no circuito de controle:

```
/******
```

Nome do programa:

Sistema de controle de antena de recepção motorizada.

Autores:

Felipe Monari da Silva;

Luiz Antonio Vilioti;

Lucas Augusto Ferreira.

Ultima revisão:

24/01/2018.

Especificações:

PIC18F4550;

Cristal de 20MHz.

```
*****/
```

```
/******
```

Funções Auxiliares

```
*****/
```

```
//Funções de leitura/gravação de variáveis float na EEPROM
```

```
float Eeprom_Read_Float(unsigned int addr){           //Leitura
    float result;
    unsigned char *ptr= (unsigned char *)&result;
    unsigned char i;

    for (i=0;i<4;i++)
        *(ptr++)=Eeprom_Read(addr++);

    return result;
}
```

```
void Eeprom_WriteFloat(unsigned int addr, float value){ //Gravação
    unsigned char *ptr= (unsigned char *)&value;
    unsigned char i;

    for (i=0;i<4;i++)
        Eeprom_Write(addr++,*(ptr++));
}
```

//Vetores de caracteres especiais do LCD

```
const char car_cima[] = {4,14,31,31,4,4,4,4};
const char car_baixo[] = {4,4,4,4,31,31,14,4};
const char car_direita[] = {0,4,6,31,6,4,0,0};
const char car_esquerda[] = {0,4,12,31,12,4,0,0};
```

//Funções dos caracteres

```
void seta_cima(char pos_row, char pos_char) {        //Seta para cima
```

```

char i;
  Lcd_Cmd(64);
  for (i = 0; i<=7; i++) Lcd_Chr_CP(car_cima[i]);
  Lcd_Cmd(_LCD_RETURN_HOME);
  Lcd_Chr(pos_row, pos_char, 0);
}

void seta_dir(char pos_row, char pos_char) {          //Seta para a direita
  char i;
  Lcd_Cmd(72);
  for (i = 0; i<=7; i++) Lcd_Chr_CP(car_direita[i]);
  Lcd_Cmd(_LCD_RETURN_HOME);
  Lcd_Chr(pos_row, pos_char, 1);
}

void seta_esq(char pos_row, char pos_char) {          //Seta para a esquerda
  char i;
  Lcd_Cmd(80);
  for (i = 0; i<=7; i++) Lcd_Chr_CP(car_esquerda[i]);
  Lcd_Cmd(_LCD_RETURN_HOME);
  Lcd_Chr(pos_row, pos_char, 2);
}

void seta_baixo(char pos_row, char pos_char) {        //Seta para baixo
  char i;
  Lcd_Cmd(88);
  for (i = 0; i<=7; i++) Lcd_Chr_CP(car_baixo[i]);
  Lcd_Cmd(_LCD_RETURN_HOME);
  Lcd_Chr(pos_row, pos_char, 3);
}

```

```
//Telas
```

```
void tela0(){

    Lcd_Out( 1, 1," Controle de antena ");
    Lcd_Out( 2, 1,"   de recepcao   ");
    Lcd_out( 3, 1,"   Projetado por:  ");

    Lcd_out( 4, 1," Felipe M. Silva  ");
    delay_ms(750);
    Lcd_out( 4, 1," Luiz A. Vilioti   ");
    delay_ms(750);
    Lcd_out( 4, 1," Lucas Augusto   ");
    delay_ms(750);

}
```

```
void tela1(char sat, char x1[8], char y1[8]){

    bytetostr(sat, sat);
    Lcd_out( 1, 1, "Satelite:");
    Lcd_out( 1, 10, sat);
    Lcd_out( 1, 13, "      ");

    Lcd_out( 2, 1, "Az:");
    Lcd_out( 2, 4, x1);

    Lcd_out( 2, 11, "El:");
    Lcd_out( 2, 14, y1);

    Lcd_out( 3, 1, "1-Presets      ");
```

```

    Lcd_out( 4, 1, "2-Manual      ");

}

```

```

void tela2(char sat, char x1[8], char y1[8]){

```

```

    bytetostr(sat, sat);
    Lcd_out( 1, 1, "Satelite:");
    Lcd_out( 1, 10, sat);
    Lcd_out( 1, 13, "      ");

    Lcd_out( 2, 1, "Az:");
    Lcd_out( 2, 4, x1);

    Lcd_out( 2, 11, "El:");
    Lcd_out( 2, 14, y1);

    Lcd_out( 3, 1, "3-Movimentar Azimuth");
    Lcd_out( 4, 1, "4-Movimentar Elev.");

}

```

```

void tela3(char sat, char x1[8], char y1[8]){

```

```

    bytetostr(sat, sat);
    Lcd_out( 1, 1, "Satelite:");
    Lcd_out( 1, 10, sat);
    Lcd_out( 1, 13, "      ");

    Lcd_out( 2, 1, "Az:");
    Lcd_out( 2, 4, x1);

```

```
Lcd_out( 2, 11, "El:");
Lcd_out( 2, 14, y1);
```

```
Lcd_out( 3, 1, "Movimentar Azimuth");
seta_dir( 4, 1);
Lcd_out( 4, 2, "Decr.");
seta_esq(4, 7);
Lcd_out( 4, 8, "Incr.");
Lcd_out( 4, 13, "OK-Tecl.");
```

```
}
```

```
void tela4(char sat, char x1[8], char y1[8]){
```

```
    bytetostr(sat, sat);
    Lcd_out( 1, 1, "Satelite:");
    Lcd_out( 1, 10, sat);
    Lcd_out( 1, 13, "    ");
```

```
Lcd_out( 2, 1, "Az:");
Lcd_out( 2, 4, x1);
```

```
Lcd_out( 2, 11, "El:");
Lcd_out( 2, 14, y1);
```

```
Lcd_out( 3, 1, "Movimentar Elev.");
seta_baixo( 4, 1);
Lcd_out( 4, 2, "Decr.");
seta_cima(4, 7);
Lcd_out( 4, 8, "Incr.");
Lcd_out( 4, 13, "OK-Tecl.");
```

```
}
```

```
void tela5(char sat, char x1[8], char y1[8]){
```

```
    bytetostr(sat, sat);
```

```
    Lcd_out( 1, 1, "Satelite:");
```

```
    Lcd_out( 1, 10, sat);
```

```
    Lcd_out( 1, 13, "    ");
```

```
    Lcd_out( 2, 1, "Az:");
```

```
    Lcd_out( 2, 4, x1);
```

```
    Lcd_out( 2, 11, "El:");
```

```
    Lcd_out( 2, 14, y1);
```

```
    Lcd_out( 3, 1, "5-Selecionar preset ");
```

```
    Lcd_out( 4, 1, "6-Gravar novo preset");
```

```
}
```

```
void tela6(char sat[8], char x1[8], char y1[8]){
```

```
    Lcd_out( 1, 1, "Satelite:");
```

```
    Lcd_out( 1, 10, sat);
```

```
    Lcd_out( 2, 1, "Az:");
```

```
    Lcd_out( 2, 4, x1);
```

```
    Lcd_out( 2, 11, "El:");
```

```
    Lcd_out( 2, 14, y1);
```

```
    seta_cima( 4, 1);
```



```
seta_baixo(4, 2);
Lcd_out( 4, 3, " OK-Sel. ESC-Sair ");
```

```
}
```

```
void tela7(char sat[8], char x1[8], char y1[8]){
```

```
    Lcd_out( 1, 1, "Satelite:");
```

```
    Lcd_out( 1, 10, sat);
```

```
    Lcd_out( 2, 1, "Az:");
```

```
    Lcd_out( 2, 4, x1);
```

```
    Lcd_out( 2, 11, "El:");
```

```
    Lcd_out( 2, 14, y1);
```

```
    seta_cima( 4, 1);
```

```
    seta_baixo(4, 2);
```

```
    Lcd_out( 4, 3, "OK-Def Sat ESC-Sai");
```

```
}
```

```
void tela8(char sat[8], char x1[8], char y1[8]){
```

```
    Lcd_out( 1, 1, "Satelite:");
```

```
    Lcd_out( 1, 10, sat);
```

```
    Lcd_out( 2, 1, "Az:");
```

```
    Lcd_out( 2, 4, x1);
```

```
    Lcd_out( 2, 11, "El:");
```

```
    Lcd_out( 2, 14, y1);
```

```
    seta_cima( 4, 1);
```

```

seta_baixo(4, 2);
Lcd_out( 4, 3, "OK-Def Az ESC-Sai");

}

```

```

void tela9(char sat[8], char x1[8], char y1[8]){
    Lcd_out( 1, 1, "Satelite:");
    Lcd_out( 1, 10, sat);

    Lcd_out( 2, 1, "Az:");
    Lcd_out( 2, 4, x1);

    Lcd_out( 2, 11, "El:");
    Lcd_out( 2, 14, y1);

    seta_cima( 4, 1);
    seta_baixo(4, 2);
    Lcd_out( 4, 3, "OK-Def El ESC-Sai");

}

```

```

void tela10(char sat[8], char x1[8], char y1[8]){
    Lcd_out( 1, 1, "Satelite:");
    Lcd_out( 1, 10, sat);

    Lcd_out( 2, 1, "Az:");
    Lcd_out( 2, 4, x1);

    Lcd_out( 2, 11, "El:");
    Lcd_out( 2, 14, y1);

    seta_cima( 4, 1);

```

```

seta_baixo(4, 2);
Lcd_out( 4, 3, "OK-Nova Pos ESC-Sai");

}

```

// Funções de acionamento dos motores

```

void pulsoAz(){                                //Pulso do motor azimuthal
    portc.rc2 = 1;
    delay_ms(1);
    portc.rc2 = 0;
    delay_ms(50);
    portc.rc2 = 1;
    delay_ms(1);
}

```

```

void pulsoEl(){                                //Pulso do motor de elevação
    portb.rb3 = 1;
    delay_ms(1);
    portb.rb3 = 0;
    delay_ms(50);
    portb.rb3 = 1;
    delay_ms(1);
}

```

//Ajuste da variável no display

```

void ajustx(float x, char x1[8]){

```

```

    x1[6]=0;
    if(x<100.00){

```

```
x1[5]=0;
}
if(x<10.00){
x1[4]=0;
}
if(x<1.00){
x1[3]=0;
}
if(x<0.10){
x1[1]=0;
}
if(x<0){
x1[0]=0;
}
}

void ajusty(float y, char y1[8]){

y1[5]=0;
if(y<10.00){
y1[4]=0;
}
if(y<1.00){
y1[3]=0;
}
if(y<0.10){
y1[1]=0;
}
if(y<0.00){
y1[0]=0;
}
```

```
}
```

//Comando de movimentação para os motores

```
float azmanual(float x, char tecla, char x1[8]){
```

```
    if(x<180.00){
        if (tecla==14){
            x=x+0.1125;
            PORTC.RC1=1;
            pulsoAz();
        }
    }
```

```
    if(x>0.00){
        if (tecla==13){
            x=x-0.1125;
            PORTC.RC1=0;
            pulsoAz();
        }
    }
```

```
    x=x+00.002;
    floattostr(x,x1);
    x=x-00.002;
    ajustx(x, x1);
```

```
    return x;
}
```

```
float elmanual (float y, char tecla, char y1[8]){
```

```
    if(y<90.00){
        if (tecla==16){
            y=y+0.1125;
            PORTB.RB2=1;
            pulsoEI();
        }
    }
```

```
    if(y>0.00){
        if (tecla==15){
            y=y-0.1125;
            PORTB.RB2=0;
            pulsoEI();
        }
    }
```

```
    y=y+00.002;
    floattostr(y,y1);
    y=y-00.002;
    ajusty(y, y1);
```

```
    return y;
}
```

```
//Função de leitura do teclado
```

```
char teclado(){
    unsigned int vanalog;
    char tecla;
```

```
delay_ms(100);
```

```
ADCON0.B0=1;
```

```
//Ativa conversor
```

```
ADCON0.B1=1;
```

```
//Inicia conversão
```

```
while(ADCON0.B1);
```

```
//Espera conversão acabar
```

```
vanalog = ADRESL + (ADRESH * 256);    /* Junta valores registrados pela
                                         conversão em uma única variável*/
```

```
if (vanalog<40){
```

```
//Protoboard 2
```

```
tecla=0;
```

```
}
```

```
if (vanalog>41 && vanalog<74){
```

```
tecla=1;
```

```
}
```

```
if (vanalog>75 && vanalog<113){
```

```
tecla=2;
```

```
}
```

```
if (vanalog>114 && vanalog<141){
```

```
tecla=3;
```

```
}
```

```
if (vanalog>142 && vanalog<175){
```

```
tecla=4;
```

```
}
```

```
if (vanalog>176 && vanalog<207){
```

```
tecla=5;
```

```
}
```

```
if (vanalog>208 && vanalog<233){
```

```
tecla=6;
```

```
}
```

```
if (vanalog>234 && vanalog<260){
```

```
tecla=7;
}
if (vanalog>261 && vanalog<313){
tecla=8;
}
if (vanalog>314 && vanalog<375){
tecla=9;
}
if (vanalog>376 && vanalog<425){
tecla=10;
}
if (vanalog>426 && vanalog<487){
tecla=11;
}
if (vanalog>488 && vanalog<573){
tecla=12;
}
if (vanalog>574 && vanalog<672){
tecla=13;
}
if (vanalog>673 && vanalog<787){
tecla=14;
}
if (vanalog>787 && vanalog<940){
tecla=15;
}
if (vanalog>941){
tecla=16;
}

return tecla;
```



```
}
```

```
//Função para digitar novos valores de Az e EI
```

```
float digitar (float a, char b[8], char m, char tecla, char func){
```

```
    char trava=0, cont=0;
```

```
    float valor=0.0, f=a;
```

```
    Lcd_Cmd(_LCD_CLEAR);
```

```
    tecla=0;
```

```
    while (cont<5){
```

```
        if(cont==0){ //Imprime no LCD qual casa decimal o usuário deverá digitar
```

```
        Lcd_out( 1, 1, "Centena      ");
```

```
        }
```

```
        if(cont==1){
```

```
        Lcd_out( 1, 1, "Dezena      ");
```

```
        }
```

```
        if(cont==2){
```

```
        Lcd_out( 1, 1, "Unidade      ");
```

```
        }
```

```
        if(cont==3){
```

```
        Lcd_out( 1, 1, "Decimal 1      ");
```

```
        }
```

```
        if(cont==4){
```

```
        Lcd_out( 1, 1, "Decimal 2      ");
```

```
        }
```

```
    Lcd_out(3, 1, b);
```

```
Lcd_out( 2, 1, "Novo valor:      ");
Lcd_out( 4, 1, "OK-Aceitar  ESC-Cnl.");
```

```
while(tecla == 0){
tecla = teclado();
}
```

```
if (tecla==12 && trava==0){ //Testa qual tecla foi pressionada
    valor=1;                //Atribui o valor equivalente ao da tecla
    cont++;                //Incrementa cont, para o valor atribuido
                           //ir para a casa decimal correta
    trava=1;                //"trava" os próximos testes de tecla
}
if (tecla==11 && trava==0){
    valor=2;
    cont=cont++;
    trava=1;
}
if (tecla==10 && trava==0){
    valor=3;
    cont=cont++;
    trava=1;
}
if (tecla==9 && trava==0){
    valor=4;
    cont=cont++;
    trava=1;
}
if (tecla==8 && trava==0){
    valor=5;
    cont=cont++;
    trava=1;
```

```

}
if (tecla==7 && trava==0){
valor=6;
cont=cont++;
trava=1;
}
if (tecla==6 && trava==0){
valor=7;
cont=cont++;
trava=1;
}
if (tecla==5 && trava==0){
valor=8;
cont=cont++;
trava=1;
}
if (tecla==4 && trava==0){
valor=9;
cont=cont++;
trava=1;
}
if (tecla==3 && trava==0){
valor=0;
cont=cont++;
trava=1;
}

if(cont==1 && trava==1){    //centena
f=(valor*100.000);        //Multiplica pela potência de 10 necessária

if(m){
if(f>100){

```

```

f=100;
}
}
else{
if(f>90){
f=0;
}
}

f=f+0.002;
floattostr(f, b);      //converte o valor para ser impresso no LCD
f=f-0.002;
trava=0;                //"Trava" as atribuições às casas decimais
}

if(cont==2 && trava==1){    //dezena
f=f+(valor*10.000);
if(m){
if(f>180){
f=180;
}
}
else{
if(f>90){
f=90;
}
}
f=f+0.002;
floattostr(f, b);
f=f-0.002;
trava=0;
}

```

```
if(cont==3 && trava==1){    //unidade
f=f+(valor);
if(m){
if(f>180){
f=180;
}
}
else{
if(f>90){
f=90;
}
}
}
```

```
f=f+0.002;
floattostr(f, b);
f=f-0.002;
trava=0;
}
```

```
if(cont==4 && trava==1){    //decimal 1
f=f+(valor*0.100);
if(m){
if(f>180){
f=180;
}
}
else{
if(f>90){
f=90;
}
}
}
```

```
f=f+0.002;
floattostr(f, b);
f=f-0.002;
trava=0;
}
```

```
if(cont==5 && trava==1){    //decimal 2
f=f+(valor*0.010);
if(m){
if(f>180){
f=180;
}
}
else{
if(f>90){
f=90;
}
}
}
```

```
f=f+0.002;
floattostr(f, b);
f=f-0.002;
trava=0;
cont=0;                //Zera cont para que o usuário possa aceitar
}                      //o valor, cancelar ou digitar outro
```

```
if(m){
ajustx(f, b);
}
else{
ajusty(f, b);
```

```

}

trava=0;

if (tecla==1){          //Se ESC for pressionado
Lcd_Cmd(_LCD_CLEAR);    //Limpa LCD
floattostr(a,b);        //Converte a (posição antiga) para char

while(tecla != 0){      //Trava tecla
tecla=teclado();
}

return a;               //Retorna a (posição antiga)
}

if(!func){              //Se a função for chamada dentro de presetgravar

if(tecla==2){

while(tecla != 0){
tecla=teclado();
}

return f;

}

}

if(func){                //Se a função for chamada no voidmain()

if (tecla== 2){
Lcd_Cmd(_LCD_CLEAR);

```

```
while(tecla != 0){
  tecla=teclado();
}
```

```
while(a>f){          //Se o novo valor for menor do que o antigo
  if (m){            //Se "m=1", motor de azimuth é acionado
    tecla=13;        //Como se o botão "para direita" estivesse pressionado
    a=azmanual(a, tecla, b);
```

```
Lcd_out( 2, 1, "Az:");
Lcd_out( 2, 4, b);
```

```
Lcd_out(4, 0, "  Movimentando...");
delay_ms(75);
Lcd_out(4, 0, "  Movimentando...");
delay_ms(75);
Lcd_out(4, 0, "  Movimentando...");
delay_ms(75);
}
```

```
else{                //Se "m=0", motor de elevação é acionado
  tecla=15;          //Como se o botão "para baixo" estivesse pressionado
  a=elmanual(a, tecla, b);
```

```
Lcd_out( 2, 11, "El:");
Lcd_out( 2, 14, b);
```

```
Lcd_out(4, 0, "  Movimentando. ");
delay_ms(75);
Lcd_out(4, 0, "  Movimentando.. ");
delay_ms(75);
Lcd_out(4, 0, "  Movimentando...");
```



```

delay_ms(75);
}
}

```

```

while(a<f){
if (m){
tecla=14;
a=azmanual(a, tecla, b);

```

```

Lcd_out( 2, 1, "Az:");
Lcd_out( 2, 4, b);

```

```

Lcd_out(4, 0,"  Movimentando. ");
delay_ms(75);
Lcd_out(4, 0,"  Movimentando.. ");
delay_ms(75);
Lcd_out(4, 0,"  Movimentando...");
delay_ms(75);
}

```

```

else{
tecla=16;
a=elmanual(a, tecla, b);

```

```

Lcd_out( 2, 11, "El:");
Lcd_out( 2, 14, b);

```

```

Lcd_out(4, 0,"  Movimentando. ");
delay_ms(75);
Lcd_out(4, 0,"  Movimentando.. ");
delay_ms(75);
Lcd_out(4, 0,"  Movimentando...");

```

```

delay_ms(75);
}
}

```

```

tecla = 0;
Lcd_Cmd(_LCD_CLEAR);
return a;
}
}

```

```

while(tecla != 0){
tecla=teclado();
}

```

```

Lcd_Cmd(_LCD_CLEAR);

}

}

```

//Função para selecionar um novo preset

```
void presetsel(char sat, float x, float y, char x1[8], char y1[8], char tecla){
```

```

    char sat2, sat3[8], address, x2[8], y2[8];
    float azi, yp;           //Variavel xp não funciona

```

```

    sat2 = 0;

```

```

    while (tecla != 1){

```

```

address=sat2*8;                                //Leitura da memória RAM
azi=eeprom_read_float(address);
address=(sat2*8)+4;
yp=eeprom_read_float(address);

bytetostr(sat2, sat3);

azi=azi+00.002;
floattostr(azi,x2);
azi=azi-00.002;
ajustx(azi, x2);

yp=yp+00.002;
floattostr(yp,y2);
yp=yp-00.002;
ajusty(yp, y2);

Lcd_Cmd(_LCD_CLEAR);
tela6(sat3, x2, y2);

while (tecla != 16 && tecla != 15 && tecla != 1 && tecla != 2){
tecla=teclado();
}

                                //Seleção do satélite
if (tecla == 16){                //Tecla cima
if(sat2<31){
sat2++;
}
else{
sat2=0;
}
}

```

```

tecla=0;
}

if (tecla == 15){      //Tecla baixo
if(sat2>0){
sat2--;
}
else{
sat2=31;
}
tecla=0;
}

if (tecla == 2){      //Tecla OK

while (tecla != 0){
tecla=teclado();
}

sat=sat2;

while(azi>x){
tecla=14;
x=azmanual(x, tecla, x1);

Lcd_Cmd(_LCD_CLEAR);
Lcd_out( 1, 1, "Satelite:");
Lcd_out( 1, 10, sat3);

Lcd_out( 2, 1, "Az:");
Lcd_out( 2, 4, x1);

```

```
Lcd_out( 2, 11, "El:");
Lcd_out( 2, 14, y1);
```

```
Lcd_out(4, 0, "  Movimentando. ");
delay_ms(75);
Lcd_out(4, 0, "  Movimentando.. ");
delay_ms(75);
Lcd_out(4, 0, "  Movimentando...");
delay_ms(75);
}
```

```
while(azi<x){
tecla=13;
x=azmanual(x, tecla, x1);
```

```
Lcd_Cmd(_LCD_CLEAR);
Lcd_out( 1, 1, "Satelite:");
Lcd_out( 1, 10, sat3);
```

```
Lcd_out( 2, 1, "Az:");
Lcd_out( 2, 4, x1);
```

```
Lcd_out( 2, 11, "El:");
Lcd_out( 2, 14, y1);
```

```
Lcd_out(4, 0, "  Movimentando. ");
delay_ms(50);
Lcd_out(4, 0, "  Movimentando.. ");
delay_ms(75);
Lcd_out(4, 0, "  Movimentando...");
delay_ms(75);
}
```

```

while(yp<y){
  tecla=15;
  y=elmanual(y, tecla, y1);

  Lcd_Cmd(_LCD_CLEAR);
  Lcd_out( 1, 1, "Satelite:");
  Lcd_out( 1, 10, sat3);

  Lcd_out( 2, 1, "Az:");
  Lcd_out( 2, 4, x1);

  Lcd_out( 2, 11, "El:");
  Lcd_out( 2, 14, y1);

  Lcd_out(4, 0, "  Movimentando. ");
  delay_ms(75);
  Lcd_out(4, 0, "  Movimentando.. ");
  delay_ms(75);
  Lcd_out(4, 0, "  Movimentando...");
  delay_ms(75);
}

while(yp>y){
  tecla=16;
  y=elmanual(y, tecla, y1);

  Lcd_Cmd(_LCD_CLEAR);
  Lcd_out( 1, 1, "Satelite:");
  Lcd_out( 1, 10, sat3);

  Lcd_out( 2, 1, "Az:");

```

```
Lcd_out( 2, 4, x1);
```

```
Lcd_out( 2, 11, "El:");
```

```
Lcd_out( 2, 14, y1);
```

```
Lcd_out(4, 0, "  Movimentando...");
```

```
delay_ms(75);
```

```
Lcd_out(4, 0, "  Movimentando...");
```

```
delay_ms(75);
```

```
Lcd_out(4, 0, "  Movimentando...");
```

```
delay_ms(75);
```

```
}
```

```
EEPROM_writefloat(0, x);
```

```
EEPROM_writefloat(4, y);
```

```
tecla=1;
```

```
}
```

```
}
```

```
}
```

```
//Função para gravar novos presets ou alterar os existentes
```

```
void presetgravar(char tecla){
```

```
char sat2, sat3[8], address=0, x2[8], y2[8], cont=0, m, func2=0;
```

```
float yp, azi;
```

```
sat2 = 1;
```

```

while (tecla != 1){

while(cont == 0 && tecla != 1){

address=sat2*8;
azi=eeprom_read_float(address);
address=(sat2*8)+4;
yp=eeprom_read_float(address);

azi=azi+00.002;
floattostr(azi,x2);
azi=azi-00.002;
ajustx(azi, x2);

yp=yp+00.002;
floattostr(yp,y2);
yp=yp-00.002;
ajusty(yp, y2);

bytetostr(sat2, sat3);
Lcd_Cmd(_LCD_CLEAR);
tela7(sat3, x2, y2);

while (tecla != 16 && tecla != 15 && tecla != 1 && tecla != 2){
tecla=teclado();
}

if (tecla == 16){      //Tecla cima
if(sat2<31){
sat2++;
}
}

```



```
else{
    sat2=0;
}
tecla=0;
}

if (tecla == 15){      //Tecla baixo
    if(sat2>0){
        sat2--;
    }
    else{
        sat2=31;
    }
    tecla=0;
}

if(tecla == 2){
    cont++;
}

}

while (cont == 2 && tecla !=1){

    while(tecla!=0){
        tecla=teclado();
    }

    Lcd_Cmd(_LCD_CLEAR);
    tela9(sat3, x2, y2);

    while (tecla != 1 && tecla != 2){
```

```

tecla=teclado();
}

```

```

if(tecla==2){

```

```

while(tecla!=0){
tecla=teclado();
}

```

```

m=0;
yp=digitar(yp, y2, m, tecla, func2);
cont++;
}
}

```

```

while (cont == 1 && tecla !=1){

```

```

while(tecla!=0){
tecla=teclado();
}

```

```

Lcd_Cmd(_LCD_CLEAR);
tela8(sat3, x2, y2);

```

```

while (tecla != 1 && tecla != 2){
tecla=teclado();
}

```

```

if(tecla==2){

```

```

while(tecla!=0){
tecla=teclado();

```

```

}

m=1;
azi=digitar(azi, x2, m, tecla, func2);
cont++;
}
}

while (cont == 3 && tecla !=1){

while(tecla!=0){
tecla=teclado();
}

Lcd_Cmd(_LCD_CLEAR);
tela10(sat3, x2, y2);

while (tecla != 1 && tecla != 2){
tecla=teclado();
}

if(tecla==2){

while(tecla!=0){
tecla=teclado();
}

address=(sat2*8)+4;
eeprom_writefloat(address, yp);
address=(sat2*8);
eeprom_writefloat(address, azi);

```

```

    cont=0;

}

}

}
}

/*****
Definições
*****/

//Pinagem do LCD

sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

//Variáveis

```

```
char sat=0, x1[8], y1[8], tecla=0, m, intrrpt=0;
float x=0, y=0;
```

```
//Interrupt para zeramento
```

```
void interrupt(){

intrrpt=1;

INTCON.INT0IF=0;
}
```

```
/******
```

```
Função principal
```

```
*****/
```

```
void main() {
```

```
char func;
```

```
ADCON0= 0b00000100;          //AN1 selecionado, idle, enabled
```

```
ADCON1= 0b00101100;          /*Deixa AN0, AN1 e AN2 como entradas
analógicas
```

```
e seta AN2 como Vref-*/
```

```
ADCON2= 0b10001010;          //Resultado ajustado à direita, 2TAD e FOSC/2
```

```
RCON.IPEN=0;          //Desabilita os níveis de prioridade dos interrupts
```

```
INTCON = 0b10010000;          //Habilita interrupção INT0
```

```
INTCON2= 0b11000000;          //Interrupção INT0 na borda de subida  
INTCON3= 0b00000000;          //Relacionado ao INT1 e INT2, desabilitados
```

```
TRISA.RA1= 1; //Entrada (Teclado Analógico)  
PORTA.RA1= 0;  
TRISA.RA2= 1; //Entrada (Vref - )  
PORTA.RA2= 0;
```

```
TRISB.RB0=1; //Entrada da interrupção INT0  
PORTB.RB0=0;
```

```
//Controle do motor azimuthal  
TRISC.RC1=0; //Saída (Direção)  
PORTC.RC1=0;  
TRISC.RC2=0; //Saída (Passo)  
PORTC.RC2=1;  
TRISC.RC0=0; //Saída (Enable)  
PORTC.RC0=0;
```

```
//Controle do motor de elevação  
TRISB.RB2=0; //Saída (Direção)  
PORTB.RB2=0;  
TRISB.RB3=0; //Saída (Passo)  
PORTB.RB3=1;  
TRISB.RB1=0; //Saída (Enable)  
PORTB.RB1=0;
```

```
Lcd_Init();  
Lcd_Cmd( _LCD_CURSOR_OFF );  
Lcd_Cmd(_LCD_CLEAR);
```

```
tela0();          //Tela de apresentação
```

```

Lcd_Cmd(_LCD_CLEAR);

while(1)
{

while(lintrprt){           //Enquanto o zeramento não for selecionado

x=Eeprom_read_float(0);
y=Eeprom_read_float(4);

x=x+00.002;
floattostr(x,x1);
x=x-00.002;
ajustx(x, x1);

y=y+00.002;
floattostr(y,y1);
y=y-00.002;
ajusty(y, y1);

tela1(sat, x1, y1);           //Tela principal

while (tecla != 12 && tecla != 11 && tecla != 1){
tecla = teclado();
}

Lcd_Cmd(_LCD_CLEAR);

if (tecla==12){           //seleciona o menu de presets

while(tecla!=0){
tecla=teclado();

```

```

}

while(tecla != 1){

tela5(sat, x1, y1);           //Menu dos presets

while (tecla != 8 && tecla != 7 && tecla != 1){
tecla = teclado();
}
Lcd_Cmd(_LCD_CLEAR);

    if (tecla==8){    //Seleciona a seleção de presets
        presetsel(sat, x, y, x1, y1, tecla);
        Lcd_Cmd(_LCD_CLEAR);
        while(tecla!=0){
            tecla=teclado();
        }
    }

    if (tecla==7){    //Seleciona a gravação de presets
        presetgravar(tecla);
        Lcd_Cmd(_LCD_CLEAR);
        while(tecla!=0){
            tecla=teclado();
        }
    }

}

while(tecla!=0){
tecla=teclado();
}

```



```
}
```

```
if (tecla==11){ //seleciona o comando manual
```

```
while(tecla!=0){
tecla=teclado();
}
```

```
while (tecla != 1){
```

```
tela2(sat, x1, y1); //Menu de comandos manuais
while (tecla != 10 && tecla != 9 && tecla != 1){
tecla = teclado();
}
```

```
Lcd_Cmd(_LCD_CLEAR);
```

```
if (tecla==10){ //seleciona a movimentação azimutal
```

```
while(tecla!=0){
tecla=teclado();
}
```

```
m=1;
```

```
while(tecla!=1){
```

```
tecla=0;
tela3(sat, x1, y1); //Tela da movimentação azimutal
while (tecla != 14 && tecla != 13 && tecla != 2 && tecla != 1){
```

```

tecla = teclado();
}

if(tecla == 2){ //Aciona a função de digitar
while(tecla!=0){
tecla=teclado();
}
func=1;
x=digitar(x,x1,m, tecla, func);
Eeprom_WriteFloat(0, x);
}

x = azmanual(x, tecla, x1);
Eeprom_WriteFloat(0, x);
}
while(tecla!=0){
tecla=teclado();
}
}

if (tecla==9){      //Seleciona a movimentação de elevação

while(tecla!=0){
tecla=teclado();
}

m=0;

while (tecla != 1){

tecla = 0;
tela4(sat, x1, y1); //Tela de movimentação de elevação

```

```

while (tecla != 16 && tecla != 15 && tecla != 2 && tecla != 1){
    tecla = teclado();
}

    if(tecla == 2){    //Aciona a função digitar
        while(tecla!=0){
            tecla=teclado();
        }
        func=1;
        y=digitar(y,y1,m, tecla, func);
        Eeprom_WriteFloat(4, y);
    }

    y = elmanual(y, tecla, y1);
    Eeprom_WriteFloat(4, y);
}
while(tecla!=0){
    tecla=teclado();
}
}
while(tecla!=0){
    tecla=teclado();
}
}
tecla=0;
}

while(intrrpt){                //Interrupção para zeramento

    tecla=16;

```

```
Lcd_Cmd(_LCD_CLEAR);
```

```
Lcd_Out( 1, 1," Destruar motores ");
```

```
Lcd_Out( 2, 1," Zerar posição? ");
```

```
Lcd_Out( 3, 1,"OK - Zerar ");
```

```
Lcd_Out( 4, 1,"ESC - Cancelar ");
```

```
while(tecla != 0){
tecla=teclado();
}
```

```
while(tecla != 2 && tecla != 1){
tecla=teclado();
}
```

```
if(tecla == 2){
```

```
portc.rc0 = 1;
```

```
portb.rb1 = 1;
```

```
tecla=16;
```

```
while(tecla != 0){
tecla=teclado();
}
```

```
Lcd_Cmd(_LCD_CLEAR);
```

```
Lcd_Out( 1, 1,"Motores destravados ");
```

```
Lcd_Out( 2, 1,"Realize ajuste man. ");
```

```
Lcd_Out( 3, 1,"OK- Finalizar ajuste");
```

```
Lcd_Out( 4, 1,"Az=90.000, El=0.000 ");
```

```
while(tecla != 2){  
    tecla=teclado();  
}
```

```
portc.rc0 = 0;  
portb.rb1 = 0;
```

```
x=90.000000;  
y=0.000000;
```

```
floattostr(x,x1);  
floattostr(y,y1);
```

```
Eeprom_WriteFloat(0, x);  
Eeprom_WriteFloat(4, y);
```

```
}
```

```
Lcd_Cmd(_LCD_CLEAR);  
tecla=0;  
intrrpt=0;
```

```
}
```

```
}
```

```
}
```

Apêndice B

Este apêndice apresenta uma explicação sobre os registradores configurados na programação do apêndice A.

Os registradores configurados para esse programa se resumem aos registradores do módulo conversor analógico-digital e da interrupção INT0: ADCON0, ADCON1 e ADCON2 e RCON, INTCON, INTCON2 e INTCON3.

ADCON0= 0b00000100;

	CHS3- CHS0	GO/DONE	ADON
ADCON0	0001	0	0

Deixa o canal AN1 selecionado e o conversor desligado. Na programação, nos momentos em que é necessário verificar o teclado, o conversor é ativado alterando-se somente os bits 0 e 1.

ADCON1= 0b00101100;

	VCFG0	VCFG0	PCFG3- PCFG0
ADCON1	1	0	1100

Deixa AN0, AN1 e AN2 como entradas analógicas e seta AN2 como Vref-

ADCON2= 0b10001010;

	ADFM	ACQT2- ACQT0	ADCS2- ADCS0
ADCON2	1	001	010

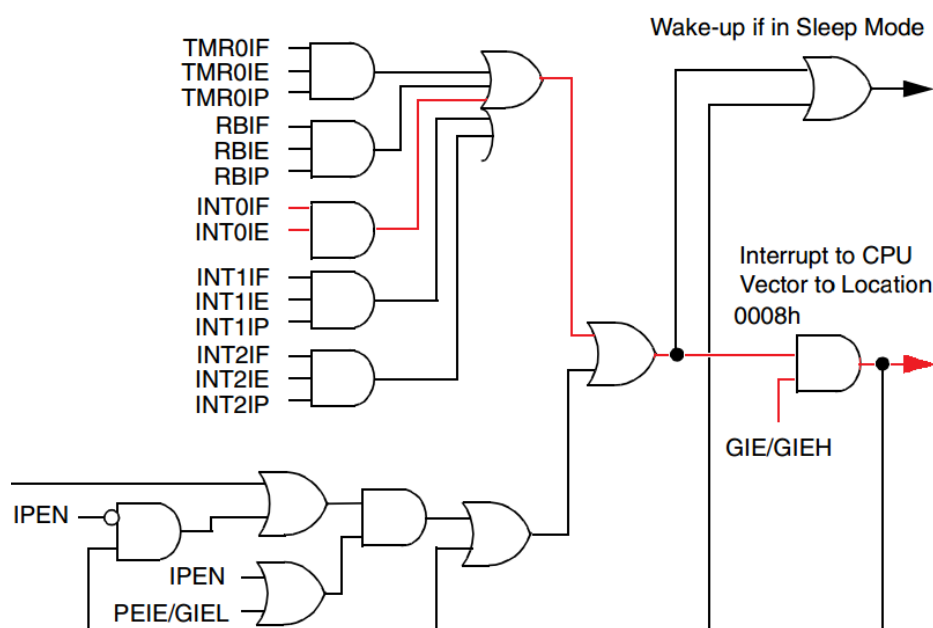
Resultado ajustado à direita, utiliza 2TAD e FOSC/2.

Como apenas o primeiro bit do registrador RCON é utilizado, configura-se apenas ele atrás de comando RCON.IPEN = 0, desabilitando a função de diferenciação de níveis de interrupção.

INTCON = 0b10010000;

	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0F	RBIF
INTCON	1	0	0	1	0	0	0	0

Habilita as interrupções globalmente e habilita a interrupção INT0, além de zerar seu indicador (o *flag* INT0IF). No *datasheet* do PIC18F4550 é possível encontrar um diagrama que mostra a sequência de habilitações necessárias para que uma interrupção possa ocorrer na figura 9-1, na página 98. A figura seguinte indica, destacado em vermelho, as habilitações necessárias nesse projeto:



INT0IF é habilitado automaticamente quando a interrupção é detectada no pino RB0.

	~RBPU	INTEDG0- INTEDG2	TMR0IP	RBIP
INTCON2	1	100	0	0

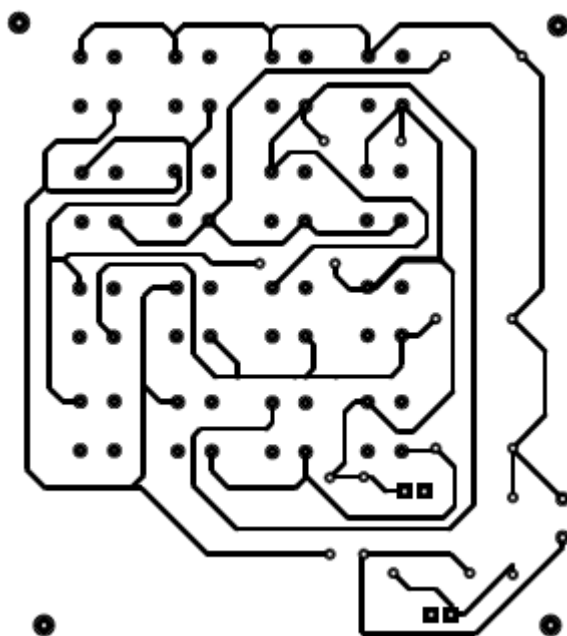
Configura a interrupção INT0 na borda de subida e desabilita o *pull-up* interno dos pinos RB.

	INT2IP	INT1IP	INT2IE	INT1IE	INT2IF	INT1IF
INTCON3	0	0	0	0	0	0

Interrupções INT1 e INT2 desabilitadas.

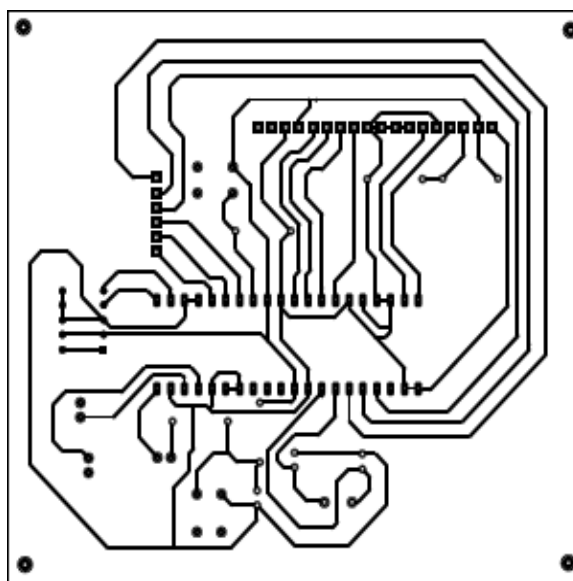
Apêndice D

Neste apêndice está disponível o fotolito pronto para impressão da placa do teclado:



Apêndice E

Neste apêndice está disponível o fotolito pronto para impressão da placa do microcontrolador.



Apêndice F

Neste apêndice está disponível o projeto do circuito do sistema de controle.

