UNIVERSIDADE FEDERAL DE OURO PRETO - UFOP

ESCOLA DE MINAS

COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO – CECAU

THIAGO REIS MARTINS

DESENVOLVIMENTO DE UM HELIODON COM POSICIONAMENTO AUTOMATICO

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Ouro Preto, 2018

Thiago Reis Martins

DESENVOLVIMENTO DE UM HELIODON COM POSICIONAMENTO AUTOMATICO

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Henor Artur de Souza

Ouro Preto
Escola de Minas - UFOP
Mar/2018

R375d Reis, Thiago Martins.
Desenvolvimento de um Heliodon com posicionamento automático [manuscrito] / Thiago Martins Reis. - 2018.

64f.: il.: color; tabs.

Orientador: Prof. Dr. Henor Artur de Souza.

Monografia (Graduação). Universidade Federal de Ouro Preto. Escola de Minas. Departamento de Engenharia de Controle e Automação e Técnicas Fundamentais.

1. Máquinas e equipamentos - Heliodon. 2. Placas de expansão (Microcomputadores) - Arduíno. 3. Motores de passo. 4. Conforto térmico - carta solar. I. Souza, Henor Artur de. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 681.5

Catalogação: ficha@sisbin.ufop.br

Monografia defendida e aprovada, em 27 de março de 2018, pela comissão avaliadora constituída pelos professores:

Prof. Dr. Henor Artur de Souza - Orientador

Prof. M. Sc. Elias José de Rezende Freitas – Professor Convidado

Profa. Dra. Karla Boaventura Pimenta Palmieri – Professora Convidada

Dedicado ás pessoas especiais que tornaram esse trabalho possível. Ao Sr. João Tomás Vieira de Souza pelo apoio técnico nas incontáveis tardes de labuta e boa proza. À Srta. Lais Diniz de Rezende Meireles por me ajudar a transformar uma "tormenta de ideias" nesse trabalho. Por fim ao Sr. Rafael Vilhena Reis Junior, por ser um verdadeiro pai para mim, não apenas durante o desenvolvimento desse trabalho, mas em toda minha vida.

A todos esses minha eterna gratidão.

RESUMO

O heliodon é um equipamento muito utilizado por arquitetos, cuja a função é simular a incidência de radiação solar em maquetes de projetos. Embora muito antigo, esse equipamento ainda é de operação manual. Neste trabalho desenvolveu-se um heliodon automatizado, que possibilita a simulação da incidência solar de uma configuração (latitude, data e horário) fornecida por um usuário na interface gráfica preparada para esse fim. Para tal é utilizado um Arduino, que interpreta os dados que o usuário insere no computador e controla dois motores de passo responsáveis pelo posicionamento do equipamento. O equipamento proposto neste trabalho simula a incidência solar, salvo algumas limitações de posicionamentos, onde, devido ao *design* escolhido para o aparelho, para azimutes próximos a 90º e 270º a altura solar mínima é limitada proporcionalmente a proximidade.

ABSTRACT

The Heliodon is an equipment widely used by architects, whose the function is to simulate the solar radiation incidence in buildings projects mockups. Although very old, the operation of this device still manual. On this monograph, an automated Heliodon is developed, which allows the simulations of the solar incidence by a configuration (latitude, date and time) provided by an user, on a graphic interface developed for that purpose. For this, an Arduino is used, which interprets the data inserted by the user on the computer and control two stepper motors responsible for positioning the equipment. The equipment proposed in this paper illustrates the solar incidence, except for some positioning limitations, where, due to the equipment chosen design, the minimum solar elevation is limited in azimuths close to 90º and 270º, this limitation is proportional to the proximity.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O heliodon é um aparelho utilizado para simular a incidência solar em maquetes. A partir das informações dessa simulação é possível estudar estratégias para que se possa então estudar estratégias para obstrução ou captação dessa incidência de acordo com as necessidades do projeto. Embora atualmente existem *softwares* capazes de simular esse efeito, o heliodon ainda é amplamente usado em laboratórios e em demonstrações de projetos.

O primeiro heliodon, *Figura 1*, foi proposto por Dufton e Beckett (1932), consistia em uma placa plana montada de modo a poder girar em torno dos eixos verticais e horizontais. Os ângulos são graduados de modo que o eixo vertical corresponda a hora do dia e o eixo horizontal inclina a placa de acordo com a latitude, e, para simular o ângulo de incidência de acordo com a estação, desliza-se a lâmpada na direção vertical.

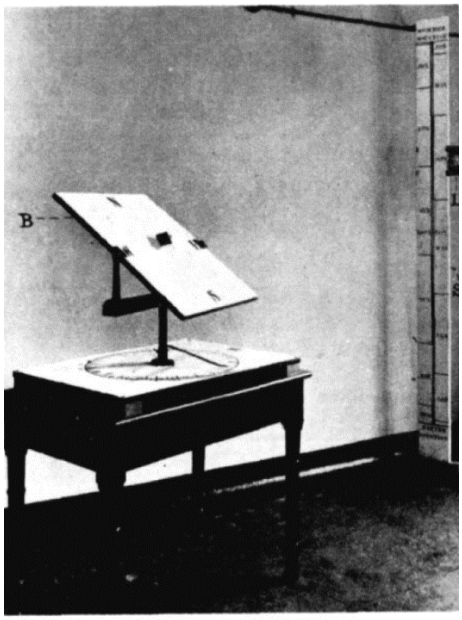

Figura 1 - Primeiro Heliodon.
Fonte: DUFTON; BECKETT, 1932

Nos equipamentos modernos, o plano em que se coloca a maquete permanece fixo enquanto uma lâmpada focal translada por um arco de 180º em torno da base que rotaciona no eixo horizontal como pode ser observado na *Figura 2* ou vertical como pode ser observado na *Figura 3*. Desse modo pode-se simular todas as variáveis (horário, latitude e data) com apenas 2 movimentos.

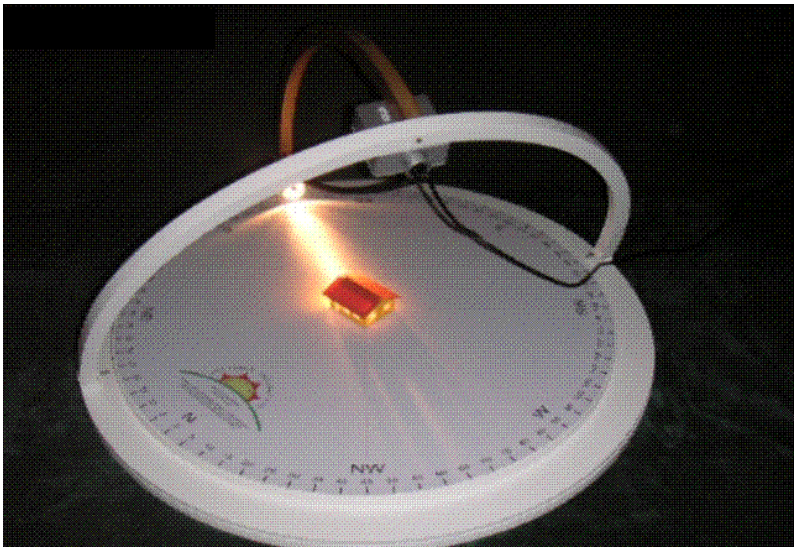

Figura 2 - Heliodon com rotação horizontal.
Fonte: HELIODON ..., 2013

Figura 3 - Heliodon com rotação vertical.
Fonte: UNIGRAN ..., 2010

Ambos os equipamentos mostrados nas *Figuras 2 e 3* possuem marcações na base e no arco, possuem também um *software* que indica quais são os ângulos que esses elementos devem estar para simular uma determinada situação (incidência solar em uma determinada latitude, dia, horário). Porém, os elementos devem ser movidos manualmente pelo operador. Com o intuito de automatizar esse procedimento, este trabalho propõe-se o desenvolvimento de um heliodon com posicionamento automatizado.

De modo que para que este simule a incidência solar sobre uma dada maquete apenas inserindo-se, em um programa de computador, a latitude, data e o horário desejado. Dessa forma, o equipamento tem uma maior precisão e simplicidade de operação,

1.1 Objetivo

Este trabalho tem como objetivo desenvolver um heliodon automatizado, que simule o movimento do Sol ao longo de um dia definido pelo usuário e a posição do sol em um dado horário ao longo de um ano.

1.2 Metodologia

O heliodon desenvolvido é acoplado a motores de passo que deslocam uma lâmpada, de modo que essa ilumine uma maquete de uma posição similar à do Sol. Para o controle desses motores é utilizado um Arduino em conjunto com *drivers* de controle.

É criada também uma interface usuário (IU) em C# utilizando o ambiente de desenvolvimento integrado da Microsoft, o Visual Studio, para estabelecer a posição desejada e enviar ao Arduino o movimento a ser descrito pelos motores. O *software* também pode ser utilizado para calcular a altura ou azimute solar, a partir de uma data e latitude.

1.3 Estrutura do trabalho

Este trabalho está composto da seguinte estrutura. No capítulo 1 faz-se uma breve introdução ao assunto estudado, definindo-se o objetivo, metodologia além de apresentar a estrutura do trabalho.

No capítulo 2, apresenta-se uma revisão dos conceitos envolvidos neste estudo, subdividindo em três partes, sendo que a primeira parte esclarece os conceitos de conforto térmico e radiação solar. Na segunda parte, fala-se brevemente sobre a geometria da insolação envolvendo conceitos como altura solar e azimute, e uma explicação sobre cartas solares. E na terceira parte, apresenta-se uma explicação sobre os materiais utilizados no projeto.

No capítulo 3, descreve-se a metodologia de implementação e o processo de montagem do aparelho. É subdividido em três partes, sendo que na primeira explica-se a montagem da mesa e a mecânica do seu movimento. Na segunda parte, é explicado como funciona o controle do equipamento e dos movimentos da lâmpada, ou seja, como os parâmetros fornecidos pela interface usuário são traduzidos em sinais de controle para o motor. Na terceira parte, descreve-se o *software* a ser utilizado pelo usuário, as formas de inserções de dados e as fórmulas para a obtenção dos parâmetros enviados aos motores.

No capítulo 4, mostram-se os testes realizados, assim como os resultados obtidos e as limitações físicas e matemáticas do projeto e também como essas são contornadas. No capítulo 5, apresentam-se as considerações finais do estudo e sugestões para melhoramentos do projeto em trabalhos futuros. Por fim apresentam-se as referências utilizadas.

2 REVISÃO BIBLIOGRAFICA

2.1 Conforto Térmico

O conforto térmico é uma situação de satisfação do indivíduo com o ambiente onde se encontra, sendo que essa afeta, diretamente, sua saúde e seu desempenho em atividades diversas (ASHRAE, 2013). Além disso, é diretamente proporcional ao índice de qualidade de vida de uma população, e inversamente proporcional ao consumo de energia para o condicionamento térmico de ambientes.

Vários fatores influem no conforto térmico tais como: velocidade do vento, umidade relativa do ar, temperatura do ar, tipo de atividade exercida e consequentemente a produção de calor metabólico pelos ocupantes do ambiente, e também tipo de vestimentas usado (NOGUEIRA et al., 2012).

Em relação a um ambiente construído, um dos principais fatores a ser observado para uma situação de conforto é a incidência solar que, segundo Grimm (1999), além da influência direta nos ventos e nas correntes marítimas, é responsável por praticamente todo o aquecimento da superfície do planeta.

A troca de energia entre o Sol e a Terra se dá por meio da radiação eletromagnética. que é uma onda progressiva de campos elétricos e magnéticos que se move com velocidade constante (aproximadamente $3*10^8 m/s$). Essa radiação é definida pela sua frequência (ou comprimento de onda) e pela sua amplitude (HALLYDAY et al., 2007).De toda a radiação emitida pelo Sol, 99% estão compreendida entre os espectros ultravioletas e infravermelho.

2.1.1 Radiação Infravermelha

Descoberta em 1800, pelo astrônomo Willian Hershell, a radiação infravermelha é a fração do espectro eletromagnético compreendida nos comprimentos de ondas entre 1mm e 740nm, e representa, segundo Grimm (1999), 49% da radiação solar que atinge o planeta. “É uma radiação de menor potencial de ação fotoquímica sobre os materiais orgânicos, mas, é a de maior capacidade de promover aquecimento radiante, o que amplia a velocidade das reações químicas nocivas[...]” (CASIMIRO, 2010, p. 33). Esse processo de aquecimento radiante torna essa radiação a principal responsável pela troca de calor entre o Sol e a Terra.

2.1.2 Radiação Ultravioleta

A radiação ultravioleta está compreendida entre os comprimentos de ondas de 400nm e 100nm. Essa radiação possui ação bactericida e fungicida, por conta disso, é usada em vários processos de descontaminação. Embora seja facilmente absorvida pela atmosfera, parte dessa radiação chega até o solo, de modo que exposição prolongada ao Sol pode causar câncer de pele, devido as ondas de menores comprimentos (abaixo de 320nm). (CASIMIRO, 2010)

2.1.3 Radiação Visível

A radiação visível corresponde a 43% da radiação solar que chega a terra (GRIMM, 1999). É a porção do espectro eletromagnético que pode ser percebida pelas células sensitivas do olho humano. É comumente subdividida em faixas de frequência que o sistema nervoso humano define como diferentes cores (CASIMIRO, 2010). Na *Figura 4* podem-se observar as faixas de frequência correspondentes a cada cor.

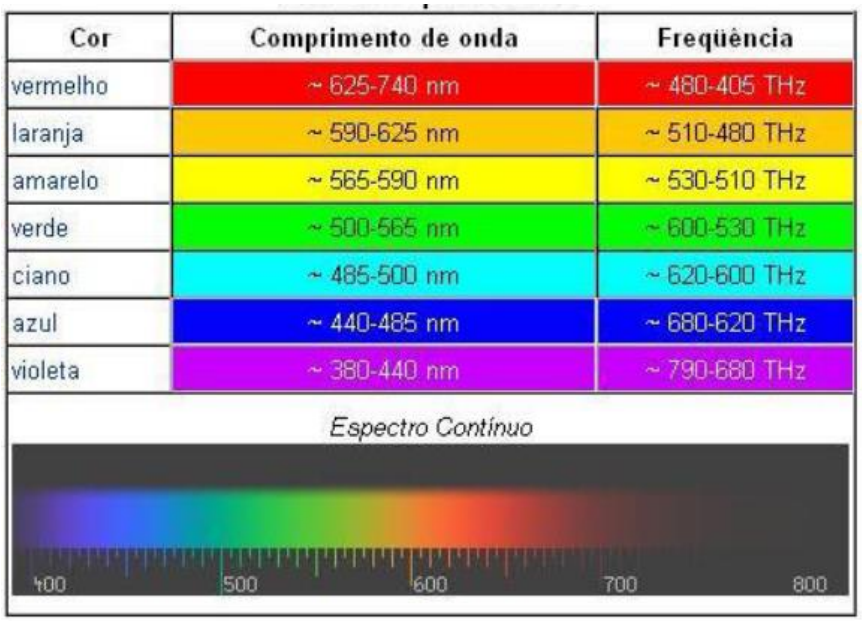

Figura 4 - Comprimentos de onda e de frequências da radiação visível.
Fonte: CASIMIRO, 2010

A radiação visível tem atuação fotoquímica, produzindo fenômenos de oxidação. A partir da absorção desses tipos de ondas, que os vegetais conseguem realizar a fotossíntese (BITTENCOURT, 2004).

2.2 Geometria da Insolação

Segundo Gonzaga (2013), o eixo de rotação da terra faz um ângulo de 66,5º com o plano de sua órbita de translação. Essa inclinação é responsável pela variação da posição do Sol para o mesmo horario ao longo do ano, como mostrado na *Figura 5*.

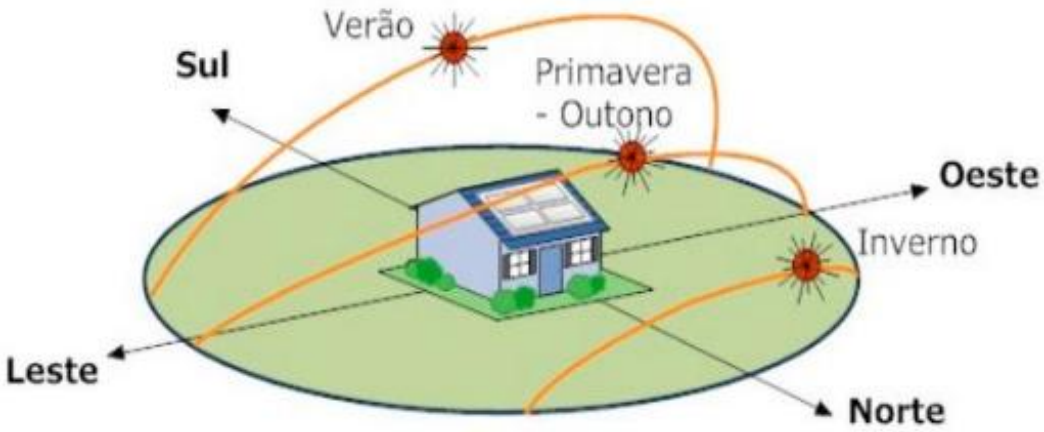

Figura 5 - Trajetória do Sol em diferentes épocas do ano.
Fonte: GONZAGA, 2013

A posição do sol em relação a Terra, influi diretamente na quantidade de energia que chega até ao solo. Quanto maior a trajetória da radiação solar pela atmosfera, mais radiação é absorvida, e essa trajetória como se pode ver na *Figura 6* é proporcional ao ângulo de incidência solar.

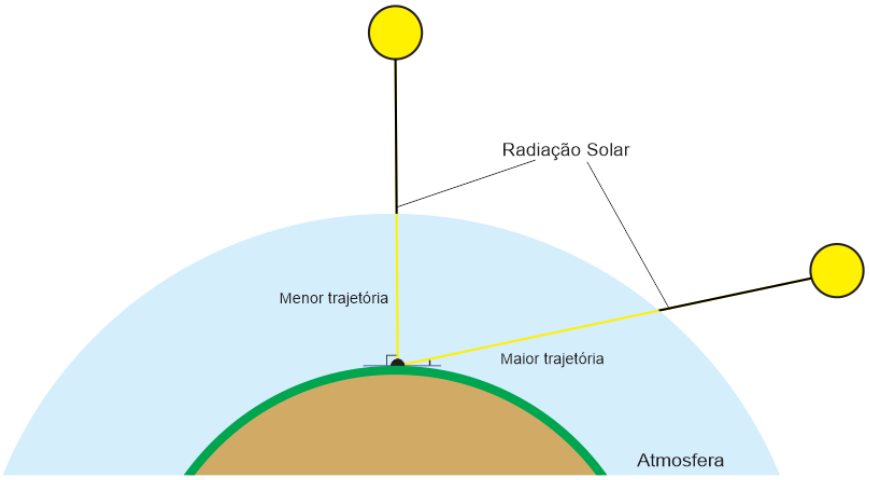

Figura 6 - Trajetória da radiação de acordo com a posição solar.

2.2.1 Altura Solar e Azimute

Sabe-se que o Sol é o centro do sistema solar e que a Terra, assim como todos os demais planetas, se movimentam ao seu redor. Porém, para facilitar a explicação e o mapeamento da posição do Sol no céu refere-se a variação da posição do Sol como trajetória ou deslocamento do Sol na abóboda celeste. Considerando isso, pode-se classificar a altura solar como o ângulo formado entre o plano observado e a posição do Sol. O azimute, por sua vez, é equivalente ao ângulo entre o norte geográfico e a direção em que se encontra o astro.

Como se pode observar na *Figura 7*, a altura solar varia de 0 a 90º atingindo seu máximo por volta das 12:00h e seu mínimo ao nascer e pôr do Sol. O azimute e altura solar variam de acordo com o horário, a data, e a posição geográfica (latitude) do observador.

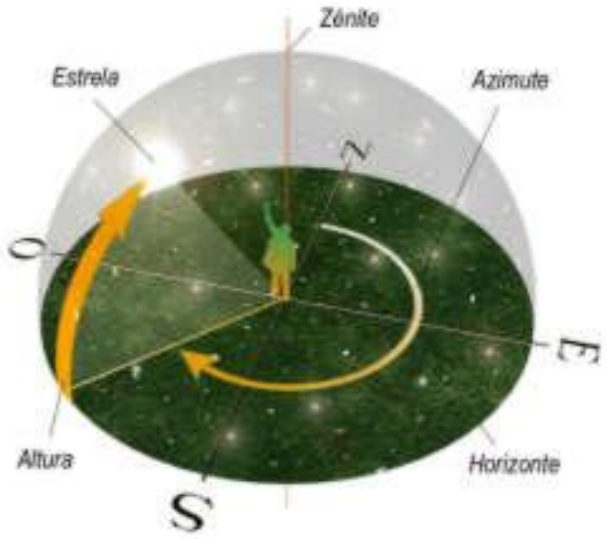

Figura 7 - Azimute e Altura Solar.
Fonte: NAUTILUS ..., 2018

2.2.2 Cartas Solares

Cartas solares, Figura *8*, são diagramas que mapeiam a posição solar na abóboda terrestre. Existe uma carta solar para cada latitude. “A carta solar para uma determinada latitude pode ser usada para determinar a posição solar em termos de altura e azimute para qualquer horário do ano” (BROWN; DEKAY, 2004, p. 31), sendo que o processo inverso também é possível. Usando as cartas solares podem-se também obter informações como o período do ano de maior insolação de uma faixada e os horários do alvorecer e crepúsculo.

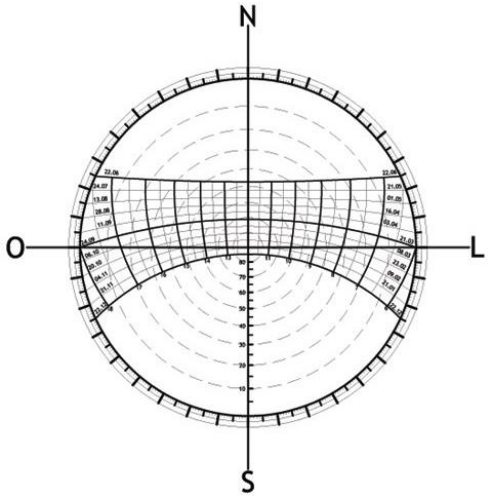

Figura 8 - Carta Solar.
Fonte: FOLHAAZERO, 2008

Na carta solar, as linhas grossas horizontais representam a data, as linhas verticais os horários, os círculos concêntricos a altura solar e as marcações radiais o azimute, *Figura 8.* (BROWN, DEKAY, 2004).

Na Figura *9* mostra-se um exemplo de como utilizar as cartas solares. Definido uma data e um horário, encontra-se o ponto de intercepção das duas linhas. Para encontrar o azimute, desenha-se uma reta interceptando esse ponto ao centro da carta e estende-se a reta até a marcação radial mais próxima. Para encontrar a Altura solar, desenha-se um arco, concêntrico aos círculos que liga o ponto até a marcação (localizada no eixo vertical.). Em caso de pontos intermediários, aproxima-se da marcação mais próxima ou utiliza-se de interpolação, de acordo com a precisão necessitada.

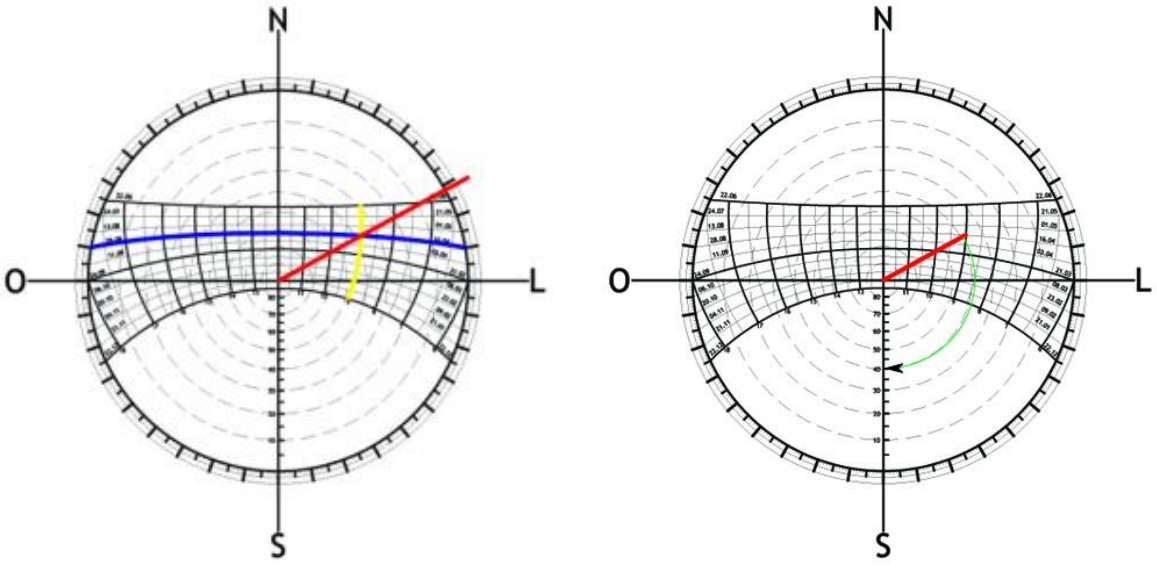

Figura 9 - Exemplo de utilização de cartas solares.
Fonte: FOLHAAZERO ..., 2008

2.2.3 Cálculo da altura e azimute solar

Além das cartas solares, é possível utilizar de expressões matemáticas para calcular o azimute e a altura solar. Para isso são utilizadas como variáveis a data, o horário e a latitude.

Segundo Campo (2014), essas variáveis devem ser tratadas antes de serem usadas. No lugar do horário utiliza-se o ângulo horário (ω), que é dado por:

$$\omega = (12 - T)\ x\ 15^{\circ} \quad (1)$$

onde,

$$T = \text{hora} + (\text{minuto}/60) \quad (2)$$

A data, deve ser dada em dias Julianos (J), que indica a quantidade de dias corridos até a presente data, exemplo: o dia 23/Fev. seria em dias Julianos seria 54 enquanto o dia 31/Dez. seria 365. Com isso pode-se calcular a declinação solar (δ) dado em graus por meio da seguinte expressão,

$$\delta = 23{,}45º * sen\left[\frac{360 * (J - 80)}{365}\right] \quad (3)$$

Com estes valores em mãos, pode-se finalmente calcular a altura solar (α) e o azimute(ψ) por meio das seguintes expressões,

$$\alpha = \sin^{-1}[\sin\delta * \sin\Phi + \cos\delta * \cos\Phi * \cos\omega] \quad (4)$$

$$\psi = \cos^{-1}\left[\frac{-sin\ \alpha * sin\,\Phi + \sin\delta}{cos\ \alpha * cos\,\Phi}\right] \quad (5)$$

onde Φ é a latitude terrestre. Com todas estas equações, pode-se calcular os parâmetros necessários para a simulação do posicionamento solar executado pelo heliodon.

3 MATERIAIS E MÉTODOS

A montagem do heliodon divide-se basicamente em três partes. Em primeiro lugar é preciso fazer a montagem do aparelho em si, é utilizado um heliodon com rotação horizontal similar ao mostrado na *Figura 2*. A mesa é feita em madeira MDF, o arco que sustenta a lâmpada é feito a partir de um tubo de alumínio, as conexões entre as partes móveis foram construídas usinando tarugos de nylon.

Em seguida é criado um sistema de posicionamento automatizado da lâmpada, isso é feito através de dois motores de passo modelo NEMA 17 WS17-0035-04-4 de 3.5 kgf.cm. A comunicação entre o computador os motores é feita através de um Arduino UNO e um par de controladores modelo A4988.

Finalmente, a terceira etapa consiste em criar uma interface ao usuário. Esse é desenvolvido na linguagem de programação C#, utilizando o *software* Visual Studio. Essa interface permite ao usuário controlar todo o funcionamento da mesa de forma fácil e intuitiva.

3.1 Montagem do Heliodon

Para deslocar corretamente a fonte de luz sobre a maquete, simulando a incidência que a luz solar teria em determinada data e região sobre um objeto em proporções reais, a mesa que apoia a maquete deve ser graduada, indicando a orientação cardeal. Assim, o usuário pode sua maquete na mesma posição em que a construção real se situaria. Na *Figura 10* mostra-se uma representação do aparelho utilizado neste trabalho.

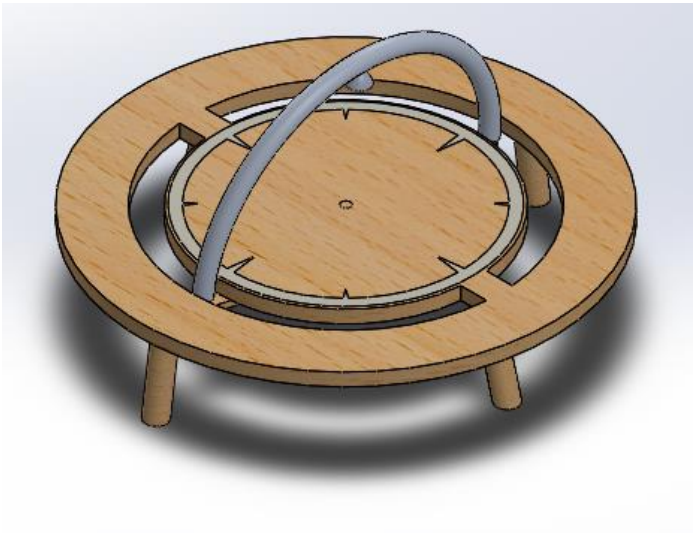

Figura 10 - Modelo em 3D do Heliodon.

Devido à proximidade da lâmpada à maquete, qualquer ângulo entre os raios de luz produzidos gerara uma sombra distorcida. Com isto em vista, escolhe-se uma iluminação a base de LED’s, pois esses possuem uma baixa abertura de feixe luminoso.

Para que se desloque sobre a maquete, a lâmpada é posicionada no centro de um arco que passa por cima da mesa. O arco gira tanto na vertical quanto na horizontal, podendo assim atingir qualquer altura solar e azimute. Para que os deslocamentos ocorram, o arco é fixado em uma peça posicionada sob a mesa, que rotacional no eixo vertical, como mostrado na Figura 11.

Figura 11 - Rotação no eixo vertical.

O movimento no eixo horizontal é realizado pelos rolamentos que prendem o arco na mesa, como mostrado na Figura 12.

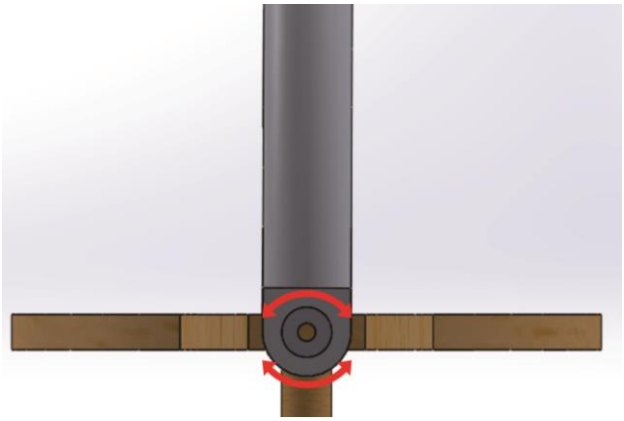

Figura 12 - Rotação no eixo horizontal.

Como o torque dos motores é muito baixo, são utilizadas roldanas de redução para possibilitar o deslocamento em torno do eixo vertical. A roldana do eixo possui 66,8mm de diâmetro,

enquanto a roldana do motor possui 28,80mm de diâmetro, resultando assim em uma redução de 1 para 2,319.

Todos esses movimentos são realizados de forma automática por dois motores de passo, de forma que o usuário interaja com o equipamento apenas via *software*, sem a necessidade de deslocar a mesa manualmente.

3.2 Arduino UNO

O Arduino UNO é uma placa controladora de fácil uso, desenvolvida para favorecer o desenvolvimento de protótipos. A placa possui uma barra de pinos que permite um fácil encaixe e desencaixe de fios, além de vir equipada com todos os componentes necessários para seu funcionamento. Ela pode ser alimentada via USB ou por qualquer fonte CC de até 12V.

A placa utiliza um microprocessador Atmega328 e possui 20 pinos de entrada/saída digitais, dos quais 6 podem ser usados como saídas PWM e 6 como entradas analógicas. Utiliza ainda um oscilador cerâmico de 16MHz como *clock*. Embora o Atmega328 não possua conexão USB, o Arduino contém um processador Atmega16U2 programado como conversor USB-serial, podendo, dessa forma, comunicar-se com outros dispositivos via USB de forma simples e rápida (ARDUINO..., 2018).

3.3 Motor de passo

O motor de passo é caracterizado por realizar uma rotação fixa a cada comando. Ele é um tipo de motor elétrico que funciona por meio de pulsos, a cada pulso o motor rotaciona uma angulação pré-determinada (geralmente 1,8º). Normalmente o motor é utilizado em situações em que se deve realizar um deslocamento determinado, como no caso de impressoras, micro-ondas e em alguns elementos de robótica.

O estator do motor de passo é composto por dois ou mais pares de polos, que são responsáveis pelo deslocamento do rotor. O rotor por sua vez é constituído de um imã permanente em forma de engrenagem, sendo que os polos se alternam a cada dente do imã. A quantidade de dentes no disco determina a quantidade de passos por revolução do rotor. Por exemplo, um imã de 64 dentes teria 64 passos por revolução e se deslocaria o equivalente a 5,625º por passo. Na Figura 13 apresenta-se um esquema de motor de passo de 2 polos com o rotor de 16 dentes.

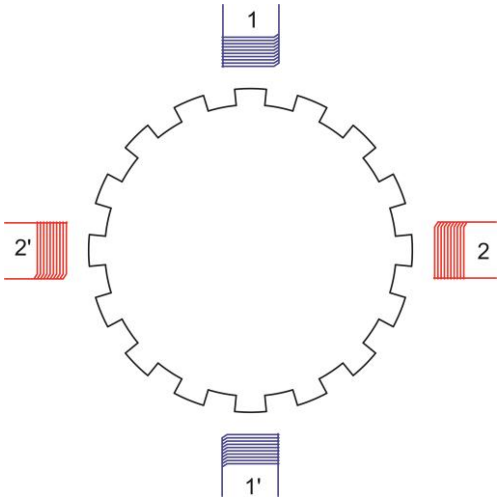

Figura 13 - Esquema Motor de Passo

Utilizando uma entrada lógica mais complexa, é possível energizar simultaneamente ambos os polos, diferindo apenas na intensidade da corrente, e dessa forma o motor pode deslocar um micro passo, ou seja, pode deslocar uma fração de um passo. Por exemplo: um motor de 64 dentes que realiza um micro passo equivalente a $^1/_2$ passo se desloca 2,8125º, o mesmo motor realizando um micro passo equivalente a $^1/_4$ de passo se desloca 1,40625º.

Para o projeto é utilizado um motor de passo do modelo NEMA 17 WS17-0035-04-4 de 3.5 kgf.cm de torque (343,23275 mN.m). Este motor possui uma tensão nominal de 8 volts e é composto por duas fases, realizando 200 passos por revolução e deslocando-se o equivalente a 1,8º por passo.

3.4 Driver de Controle

Cada terminal do Arduino pode fornecer uma corrente elétrica de apenas 40mA e tensão de 5V. Para fornecer a tensão e a corrente de alimentação nominal do motor, 8V e 400mA para o utilizado nesse projeto, é necessário um circuito para amplificar os sinais de comando. Com esse intuito, é utilizado o driver controlador A4988 da Allegro MicroSystems, esse é um driver para motores de passo de fácil operação. Possui controladores de corrente e tensão e pode operar entre 8V e 35V e fornecer corrente de até 2A (ALLEGRO MICROSYSTEMS, 2016).

O driver A4988, mostrado na Figura 14, além de funcionar como amplificador de tensão e corrente, auxilia também no controle do motor. É possível fazer o motor deslocar um micro passo apenas enviando um sinal logico ao pino *'step'* e alterar sua direção alterando o sinal

logico no pino ‘dir’ simplificando muito o programa de controle. (ALLEGRO MICROSYSTEMS, 2016). É possível também alterar a resolução dos micro passos via *software* utilizando os pinos MS1, MS2 e MS3 conforme apresentado na Tabela.

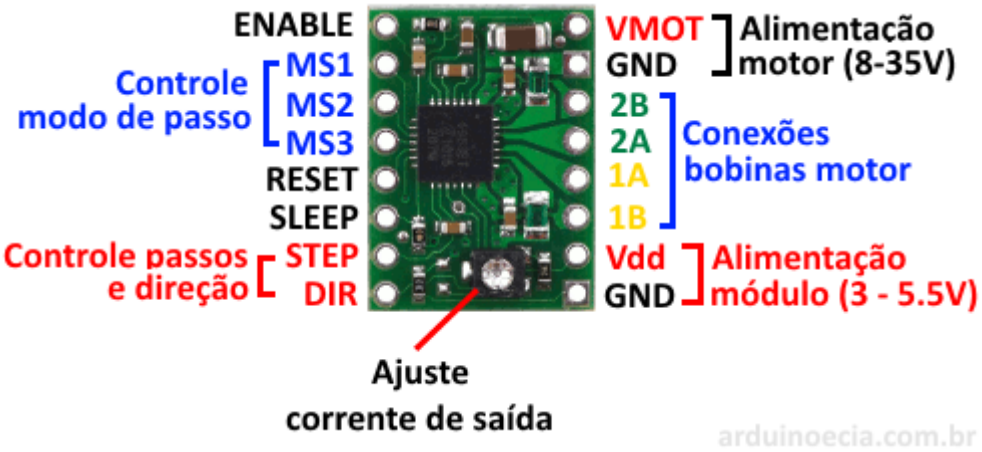

Figura 14 – Esquema de pinos do driver A4988.
Fonte: ARDUINO...,2015.

Tabela 1 - Nível Lógico x Resolução de Micro Passo.

MS1	MS2	MS3	Resolução de Micro Passo
Nível Lógico			
Baixo	Baixo	Baixo	Passo Inteiro
Alto	Baixo	Baixo	$^1/_2$ Passo
Baixo	Alto	Baixo	$^1/_4$ Passo
Alto	Alto	Baixo	$^1/_8$ Passo
Alto	Alto	Alto	$^1/_{16}$ Passo

Fonte: ALLEGRO MICROSYSTEMS, 2016.

3.5 Controle do Equipamento

O movimento da lâmpada é feito por motores de passo. Um deles é responsável pela rotação no eixo vertical, que representa o azimute. A altura solar é representada pela rotação do arco no eixo horizontal, que é realizada por outros dois motores.

A programação do Arduino, que controla os motores, é dividida em duas partes. Ao iniciar o programa ambos os motores movimentam o equipamento até pressionarem as chaves de final

de curso, e então retornam para a posição zero (azimute = 0º e altura = 180º), iniciando a segunda etapa.

Nessa etapa, o Arduino executa a rotina de verificar a porta serial, à espera de um comando. Uma vez recebido, o controlador transforma o sinal em valores numéricos, referentes ao número de passos que devem ser desenvolvidos por ambos os motores, para que esses desloquem o equipamento até a posição desejada. Em seguida, informa ao computador que a operação foi concluída e retorna a sua rotina inicial.

Durante o deslocamento entre uma configuração e outra, ambos os motores se movimentam simultaneamente, fazendo com que o movimento resultante da lâmpada seja mais natural, assemelhando-se à movimentação do sol.

3.6 Interface Usuário

Para o controle do heliodon, é projetado uma interface usuário simples e intuitiva. Como mostrado na Figura 15.

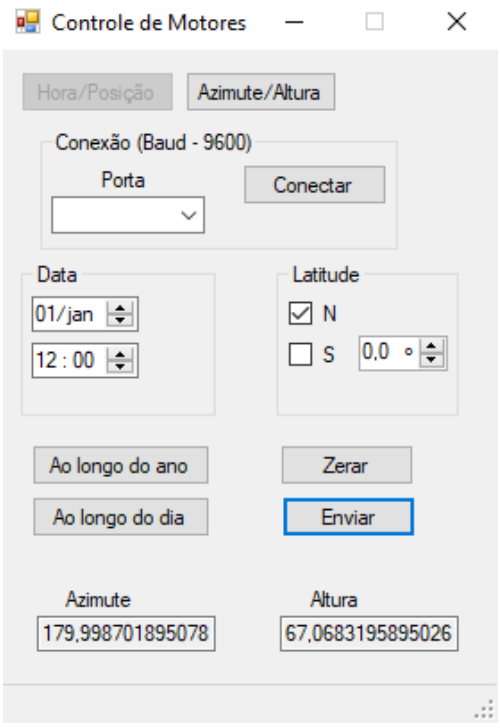

Figura 15 - Interface Usuário.

Os dois primeiros botões, Figura 16, alteram a forma de inserção dos dados. O botão Hora/Posição inicia-se apertado e faz com que o programa calcule a altura e o azimute por meio

da latitude, data e horário informados pelo usuário. Enquanto isso, o botão Azimute/Altura permite ao usuário inserir diretamente a altura e o azimute desejados.

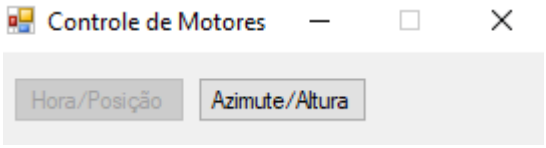

Figura 16 - Botões Modo de funcionamento.

O grupo Conexão, mostrado na Figura 17, inicia a conexão de dados com a placa da mesa. Esta conexão é feita em uma velocidade de sinalização de 9.600 bits/segundos (*bauds* = 9600). O campo Porta mostra ao usuário as portas seriais disponíveis. Para que se escolha a porta em que se encontra o controlador. O botão Conectar inicia a conexão com a placa Arduino e quando pressionado transforma-se no botão Desconectar, como mostrado na Figura 18, que interrompe a conexão.

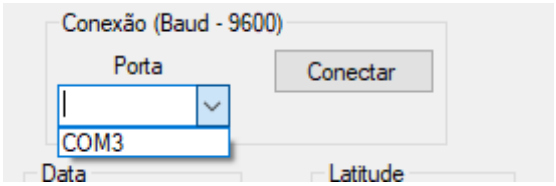

Figura 17 - Grupo conexão

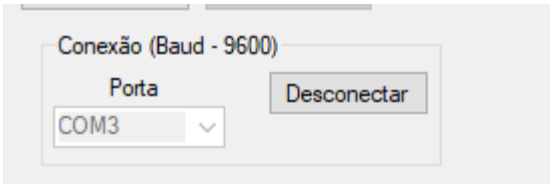

Figura 18 - Botão Desconectar.

Os campos ‘Data’ e ‘Latitude’, Figura 19, apenas disponíveis no modo hora/posição, permitem que sejam selecionadas as variáveis de entrada. Assim que o usuário modifica um dos campos, o azimute e a altura são calculados e modificados.

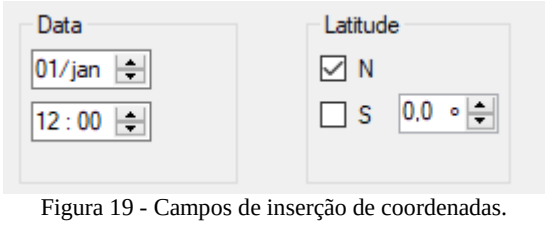

Figura 19 - Campos de inserção de coordenadas.

Os botões de envio, Figura 20, dão o comando para a movimentação do heliodon. O botão ‘Zerar’ inicia a rotina ‘zerar mesa’ do controlador que move a mesa para a posição zero. O botão ‘Enviar’ transforma os parâmetros em números de passos a serem executados pelos motores, comparam com a posição atual do equipamento e enviam um sinal para que a mesa movimente-se somente o necessário para chegar à posição desejada.

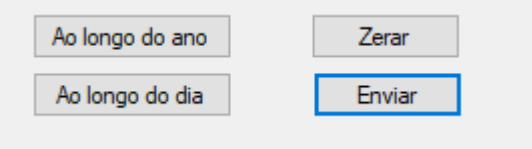

Figura 20 - Botões de envio

Os botões “Ao Longo do Ano” e “Ao Longo do Dia” simulam o movimento do Sol de acordo com o comando dado. Para simulação ao longo do ano, o programa utiliza a latitude e a hora, informadas pelo usuário, e varia a data de 1º de janeiro a 31 de dezembro, de 2 em 2 dias. E assim, reposiciona a lâmpada a cada uma destas variações.

Para o comando “Ao Longo do Dia”, o processo é semelhante: a data e a latitude são fixadas e o horário varia do amanhecer ao anoitecer, calculados pelo programa em intervalos de 10 em 10 minutos.

Os campos “Azimute” e “Altura”, Figura 21, mostram ao usuário o azimute e a altura, em graus, para os dados fornecidos nos campos “Data e Latitude”. No modo Azimute/Altura, Figura 22, o campo fica em branco, afim de que o usuário forneça a altura e o azimute desejados.

Figura 21 - Campos Azimute e Altura

Figura 22 - IU modo Azimute/Altura

4 RESULTADOS

Após finalizado o projeto e a montagem, iniciou-se a etapa de testes, a fim de encontrar e solucionar possíveis problemas e dificuldades de utilização. Neste capítulo, são retratados testes e dificuldades na construção do equipamento.

Ainda durante a fase de projeto do heliodon proposto são observadas algumas limitações fundamentais. A primeira delas refere-se às posições inalcançáveis da lâmpada devido ao *design* do aparelho, que é coberta em detalhes na seção 4.1. Outro problema encontrado é uma singularidade na expressão utilizada para o cálculo do azimute pelo *software*. Na sessão 4.2 descreve-se este problema e as soluções para eles encontradas.

4.1 Limitações do posicionamento

Para possibilitar o movimento do arco em torno da maquete, esse é apoiado abaixo da mesa, como pode ser observado na Figura *11*. Isso porém, impede que o equipamento assuma certas posições. Para alturas diferentes de 90º é impossível a simulação de azimutes menores que 10º ou maiores que 345 (entre $-15º$ e 10º). Alturas abaixo de 45º não podem ser simuladas em azimutes menores que 15º.

Como o arco não pode dar uma volta completa em torno da mesa, os azimutes acima de 90º são alcançados por meio de ângulos de altura complementares, ou seja, para azimutes maiores que 180º, tem-se

$$Azimute_{mesa} = Azimute_{real} - 180º \quad (6)$$

$$Altura_{mesa} = 180º - Altura_{real} \quad (7)$$

Na *Figura 23,* mostra-se os ângulos de azimute alcançados pela mesa e as limitações de alturas nesses ângulos.

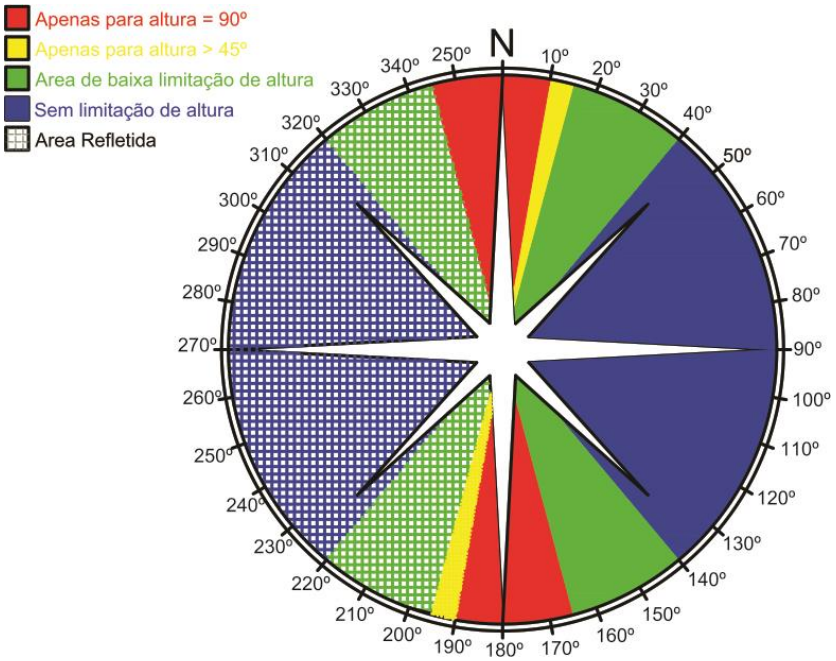

Figura 23 - Limitações de posicionamento.

Para contornar esse problema, é implementado, no *software* do aparelho, uma rotina que verifica se as coordenadas enviadas encontram-se na zona inalcançável, caso verificado que sim, o *software* altera a altura solar enviada para a altura mais próxima alcançável naquele azimute.

4.2 Singularidades nas equações

Como visto no Capitulo 2, utiliza-se a seguinte equação para o cálculo do azimute:

$$\psi = \cos^{-1}\left[\frac{-sin\,\alpha * sin\,\Phi + \sin\delta}{cos\,\alpha * cos\,\Phi}\right] \quad (5)$$

Pode-se observar que a equação possui duas singularidades, para os casos $\alpha = 90º$ ou $\Phi = 90º$, quando executadas no computador, essas equações retornam “NaN” (Not a Number) como resultado, o que causa erro nas outras etapas do programa.

Uma solução simples para o problema é somar ao denominador um valor próximo ao infinitesimal (10^{-5}), dessa forma contorna-se a divisão por 0 sem alterar o resultado da expressão.

4.3 Testes realizados

4.3.1 Testes do Software de Controle

Para testar o *software* do Arduino é utilizado o monitor serial da interface de desenvolvimento deste, Figura 24. Digita-se manualmente o comando que é enviado pela interface usuário e verifica-se a resposta do programa à esses comandos.

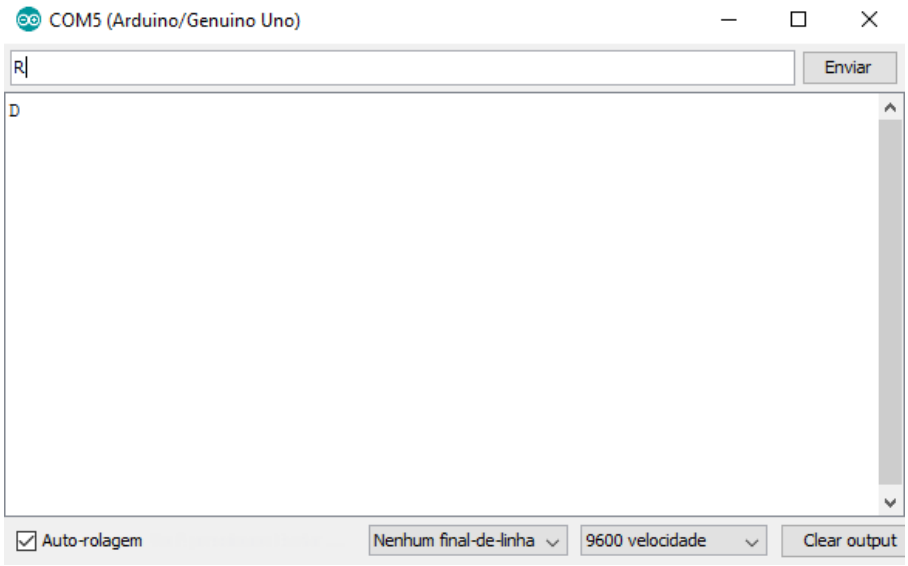

Figura 24 - Monitor Serial Arduino.

São enviados sequencialmente os comandos: ‘R’ (comando para iniciar a rotina ‘Resetar Mesa’), ’MxxxAyyy’ (comando que inicia a rotina ‘Move Mesa’, que movimenta a mesa em xxx passos e do arco em yyy passos). Posteriormente é implementada a rotina de reestabelecimento de contato que é chamada enviando o comando ‘T’. Para cada comando enviado o controlador deve realizar a rotina correspondente e enviar para a porta serial o caractere ‘D’ que tem como objetivo informar ao computador que a execução da rotina está concluída.

Esses testes possibilitaram testar e ajustar o funcionamento do controlador, de forma isolada, ou seja, independente da comunicação com o computador. Alguns dos problemas verificados e corrigido nessa etapa a rotina de separação da *string* (tipo de dados em C que armazena um vetor de caracteres) ’MxxxAyyy’ em duas variáveis do tipo *int* (tipo de dados em C que armazena um número inteiro). Esse problema é resolvido usando a função ‘Serial.Parseint()’ da biblioteca Serial do arduino.

4.3.2. Testes da rotina ‘Zerar Mesa’

Ainda utilizando o monitor serial, é testada a função ‘zerar mesa’. Nesta etapa é observado que algumas vezes as chaves de final de curso não estão acionadas. Essas são então substituídas por versões menores e mais leves de se pressionar, após a substituição a rotina funciona como esperado.

4.3.3 Testes de comunicação com a Interface do Usuário.

Em seguida é testada a Interface Usuário (IU) isolado do controlador de forma semelhante à realizada na seção 4.3.1. Para isso, é utilizado um emulador de porta serial, Figura 25. Nessa etapa são pressionados os botões na interface usuário e verificado no emulador se o comando é enviado de forma correta.

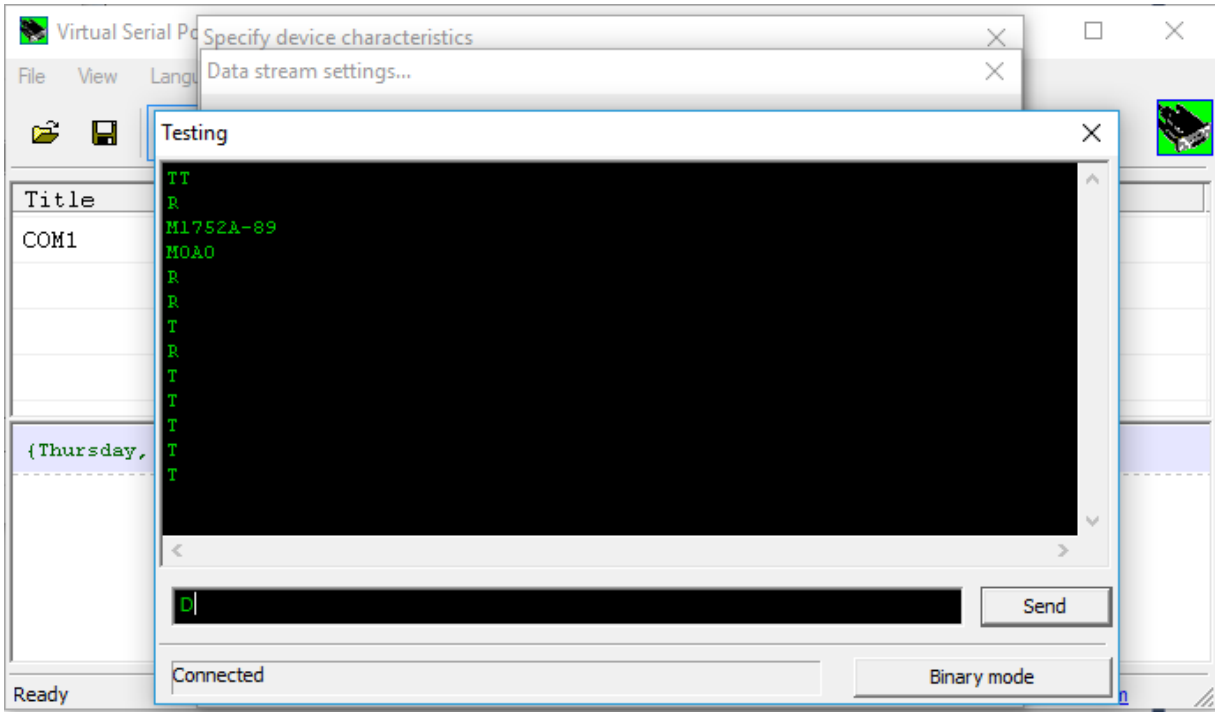

Figura 25 - Emulador de porta serial.

Apesar do funcionamento inicialmente correto, verifica-se que é possível enviar comandos para o equipamento antes que o mesmo tenha concluído a execução do comando anterior. Para evitar esse problema, é criada uma rotina que aguarda a resposta do controlador (caractere ‘D’) antes de permitir ao usuário enviar um novo comando, e caso não haja resposta do controlador em

tempo hábil, envia o caractere ‘T’ que é responsável pela rotina ‘reestabelecer o contato’ do controlador.

4.3.4 Testes dos Motores

Com a comunicação entre a IU e o controlador estabelecida, são testados os motores desacoplados da máquina. É verificado que os motores desenvolvem um deslocamento igual ao comandado em todos os casos.

4.3.5 Testes de Funcionamento do Equipamento Montado

Testa-se então o equipamento montado. Nesse teste é verificada uma diferença significativa entre o comando enviado e o deslocamento realizado. Constata-se que essa diferença é causada pela insuficiência de torque dos motores usados até então, o 26BYJ48. Esses são substituídos por motores NEMA17 de 3,5kgf.cm, citados na sessão 3.3. Uma vez que o controle dos novos motores é dado de forma diferente dos anteriores, é necessária a alteração de grande parte do código do controlador. É necessária também uma nova calibração dos comandos enviados pela IU, uma vez que o deslocamento por passo é diferente entre os motores. Terminadas as alterações são refeitos os testes, nas Figuras 26 a 34 mostra-se paralelamente o comando enviado e o posicionamento resultante desse comando.

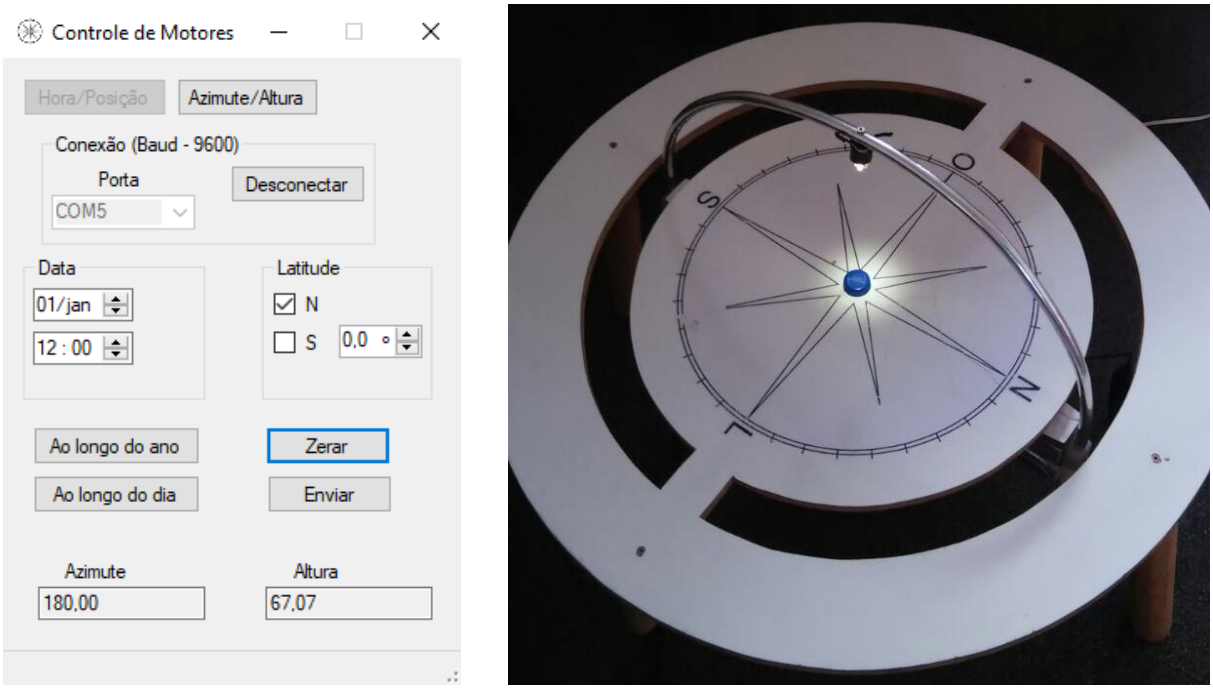

Figura 26 - Teste comando 'Zerar'.

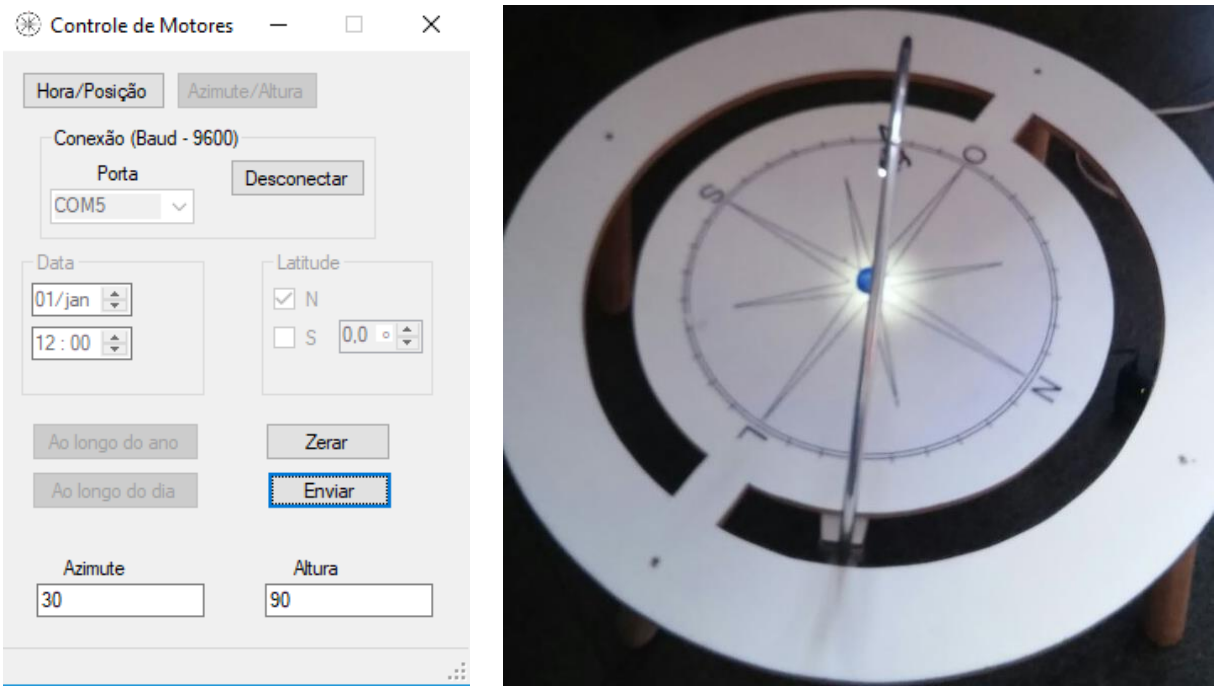

Figura 27 - Teste Azimute 30º Altura 90º.

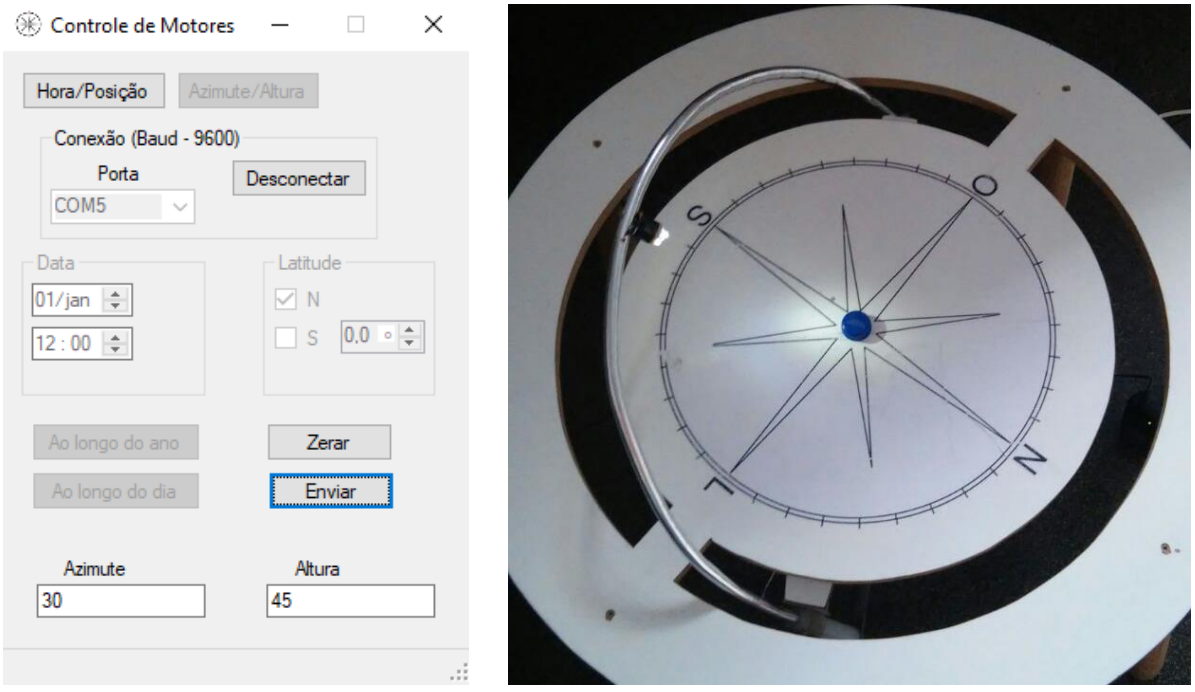

Figura 28 - Teste Azimute 30º Altura 45º.

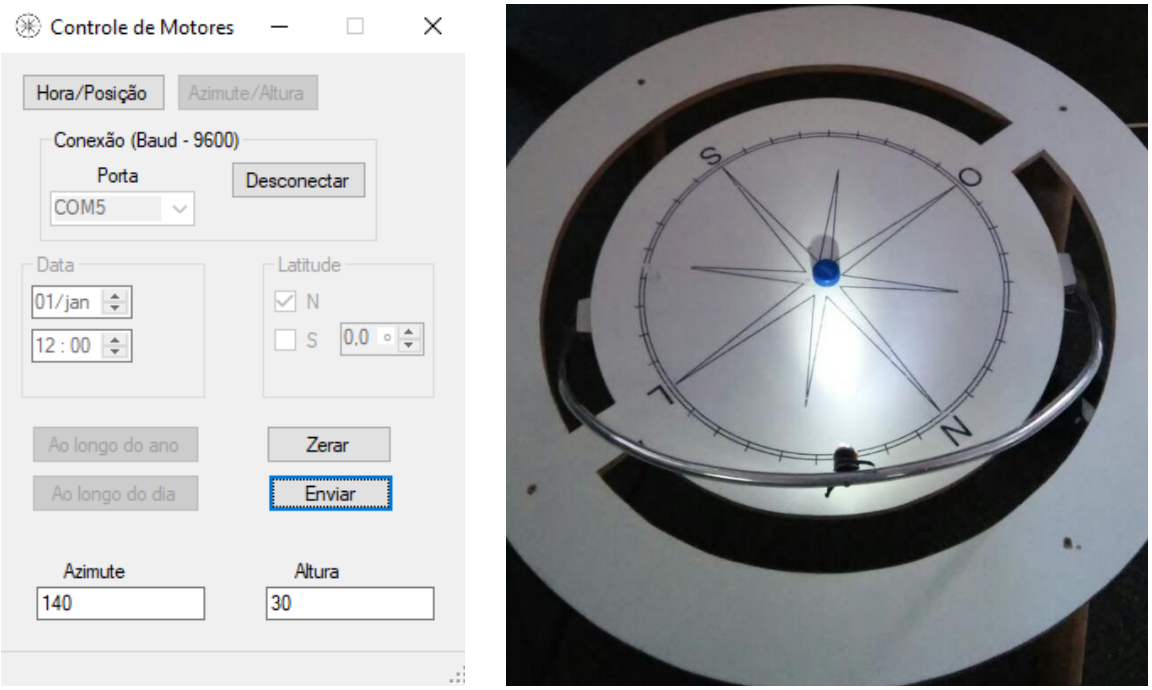

Figura 29 - Teste Azimute 140º Altura 30º.

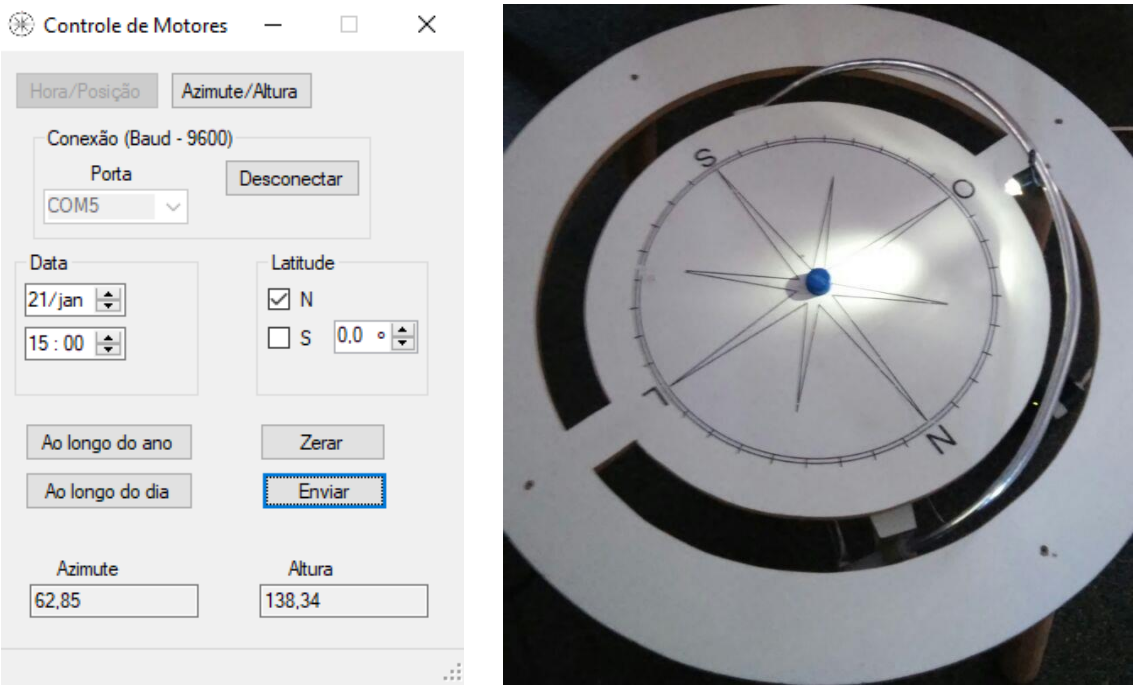

Figura 30 - Teste Dia: 21/Jan. 15h 0º Lat.

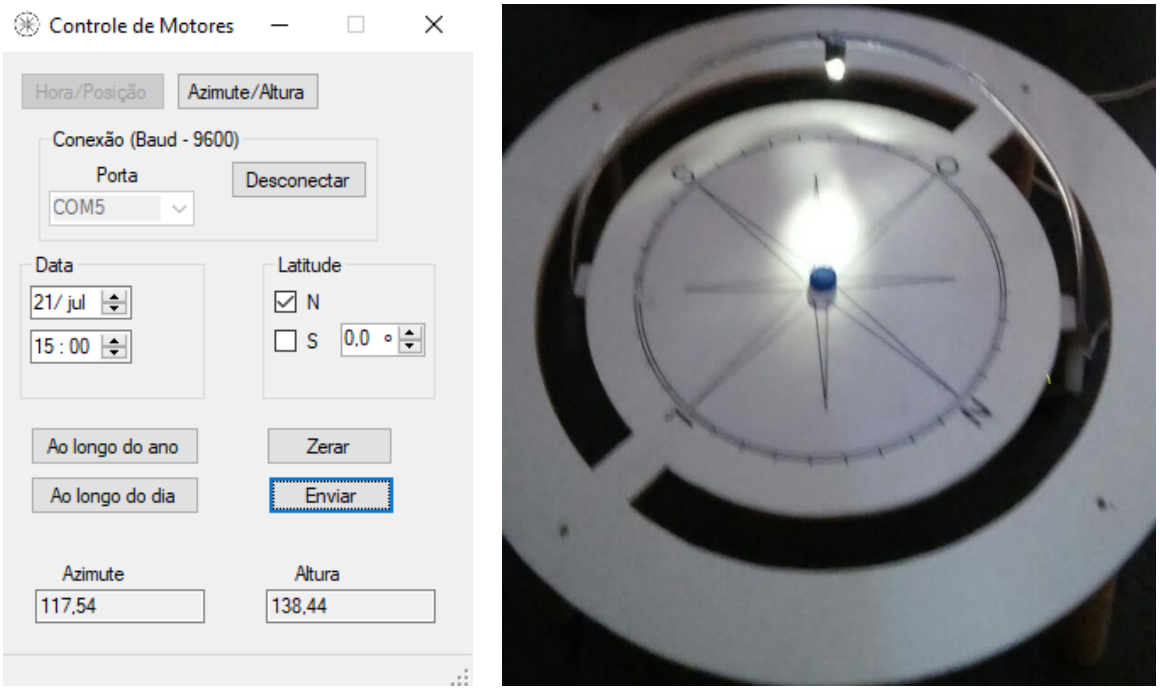

Figura 31 - Teste Dia: 21/Jul. 15h 0º Lat.

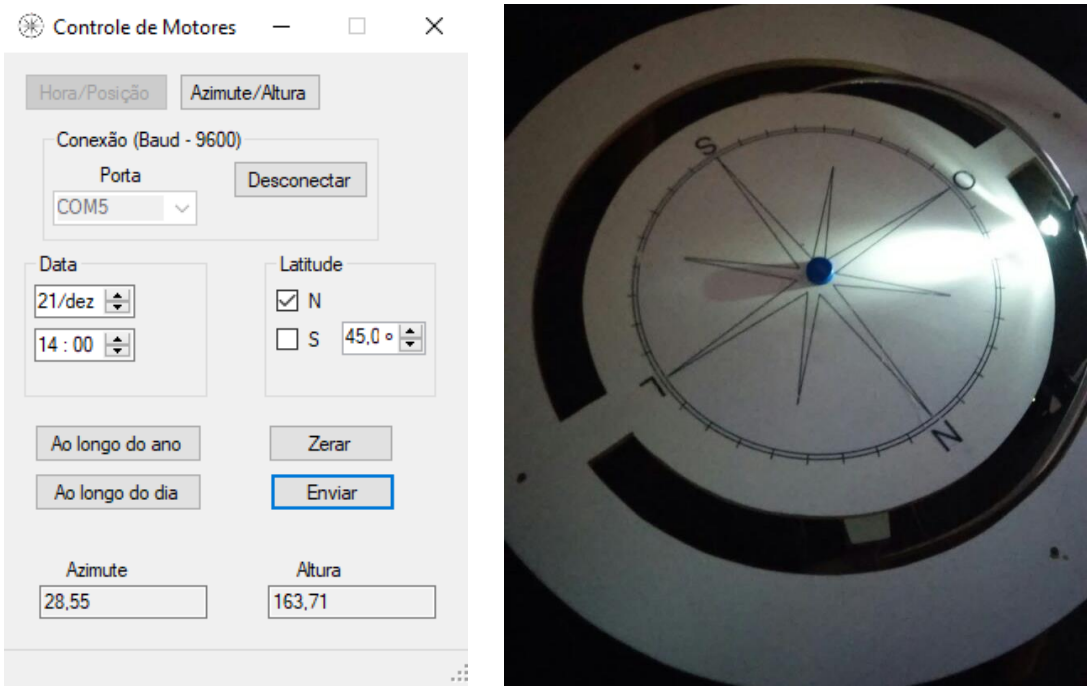

Figura 32 - Teste Dia: 21/Dez. 14h 45º Lat. N.

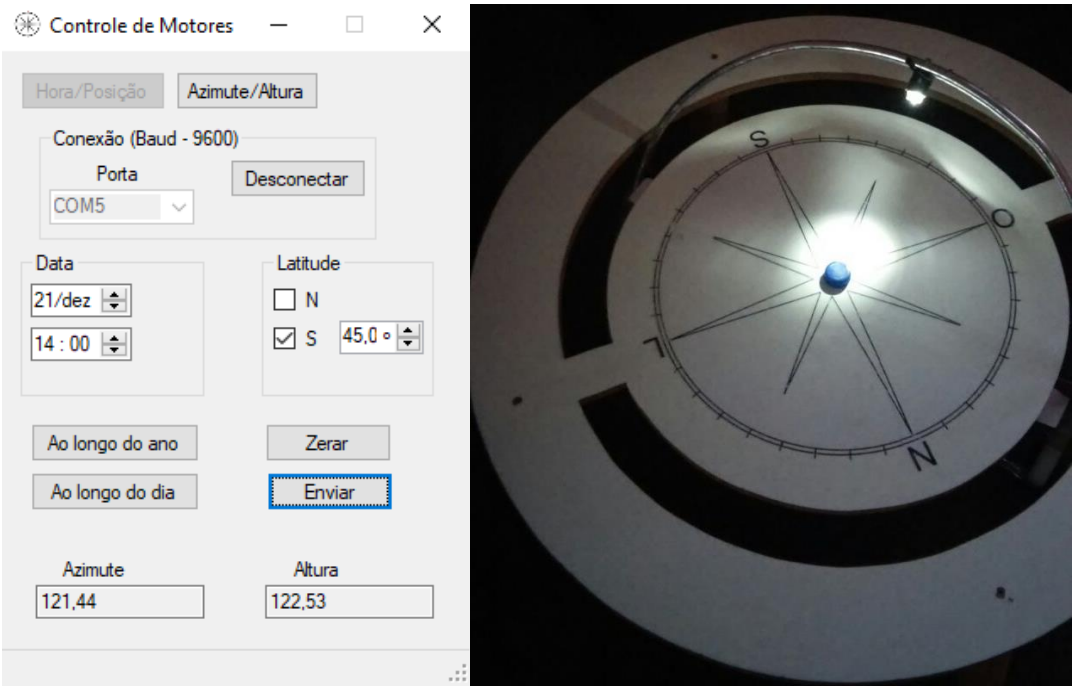

Figura 33 - Teste Dia: 21/Dez. 14h 45º Lat. S.

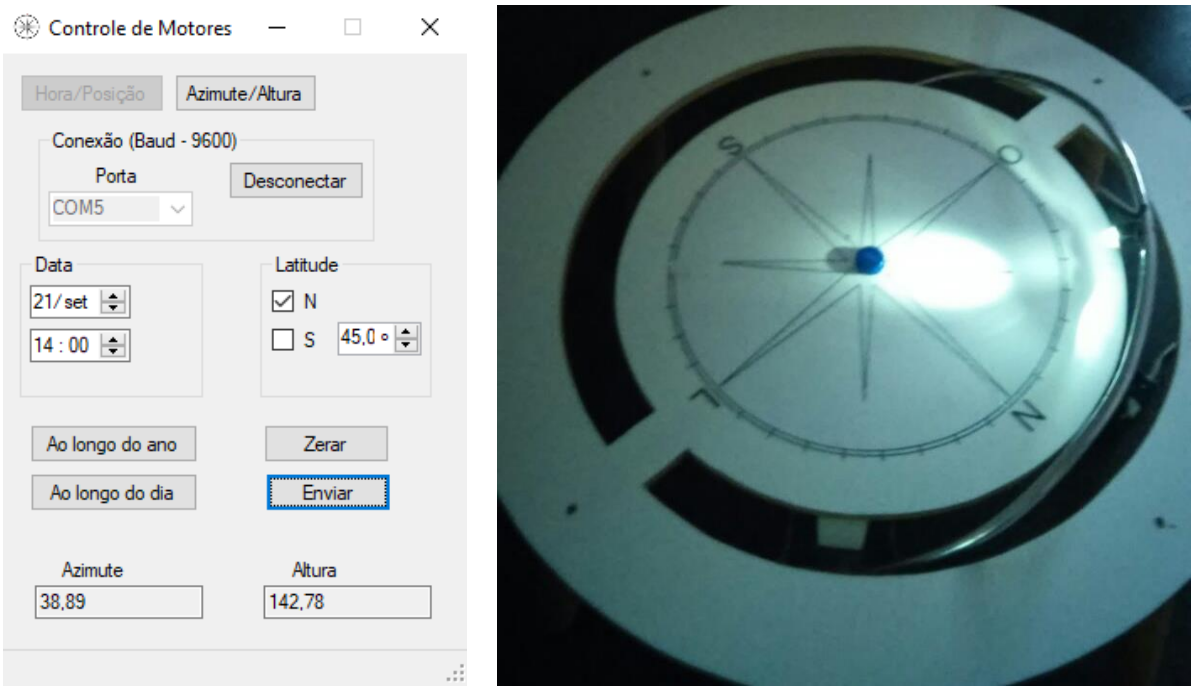

Figura 34 - Teste Dia: 21/Set. 14h 45º Lat. N.

São realizados também testes das funções ‘Ao longo do ano’ e ‘Ao longo do dia’. Não são observados problemas na execução de nenhuma delas.

5 CONSIDERAÇÕES FINAIS

Embora o estudo da insolação possa ser feito, atualmente, por meio de *softwares* de projeto assistido por computador (CAD) o heliodon oferece um apelo visual, extremamente importante na didática e na propaganda. O heliodon automatizado, desenvolvido nesse trabalho, amplia esse benefício, principalmente devido as suas funções de simulação de movimento solar ao longo do dia ou do ano, que, trazem um impacto visual a um projeto arquitetônico bem desenvolvido.

O equipamento desenvolvido simula, com sucesso a incidência solar, salvo as limitações de posicionamentos apresentadas na seção 4.1. A utilização de motores de passo, junto com a função ‘resetar mesa’ provou-se suficiente para o posicionamento correto do equipamento.

A utilização do controlador Arduino UNO simplificou o desenvolvimento do projeto, porém é utilizado apenas 10% de sua capacidade de memória, portanto para o desenvolvimento de uma versão comercial do equipamento, sugere-se a utilização de um controlador mais simples, como o PIC 16F.

A fim de promover uma maior autonomia do equipamento, sugere-se, como incentivo para trabalhos futuros, a substituição da comunicação USB por alguma comunicação remota, o que reduziria a quantidade de cabos que podem, eventualmente, interferir no movimento do heliodon. Alguns controladores populares no mercado, como o ESP32 possui módulos de Wi-Fi e *bluetooth* integrados

Sugere-se ainda o desenvolvimento em tamanho real do equipamento, todo o sistema de controle e a IU desenvolvidos nesse trabalho podem ser aplicados ao novo, sendo necessário apenas a substituição dos motores por modelos com torque compatível com a nova mesa. Com esse intuito apresenta-se no Anexo III um modelo de placa de circuito impresso para ser utilizada com os drivers de controle e ligada ao Arduino, reduzindo assim a fiação do equipamento, e prevenindo possíveis problemas na conexão dos mesmos.

REFERÊNCIAS

ALLEGRO MICROSYSTEMS. DMOS Microstepping Driver with Translator And Overcurrent Protection. **Allegro MicroSystems Website**, Dallas, p. 33, 2016. Disponivel em: <www.allegromicro.com/en/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A4988.aspx >. Acesso em: 26 fevereiro 2018.

AMERICAN SOCIETY OF HEATING, REFRIGERATING AND AIR-CONDITIONING ENGINEERS. **ANSI/ASHRAE 55**: Thermal Environmental Conditions for Human Occupancy. Atlanta, 2013.

.ARDUINO®. **Arduino Website**, 2018. Disponivel em: <https://www.arduino.cc/>. Acesso em: 25 Fevereiro 2018.

ADUINO & Cia. Controle de motor de passo bipolar com o driver A4988. 2015. Disponível em: <https://www.arduinoecia.com.br/2015/03/controle-motor-de-passo-bipolar-driver-A4988.html>. Acesso em: 22 mar. 2018

BITTENCOURT, L. **uso das cartas solares:** diretrizes para arquitetos. 4ª. ed. Maceió: edUFAL, 2004.

BROWN, G. Z.; DEKAY, M. **Sol, Vento & Luz:** estratégias para o projeto de arquitetura. 2ª. ed. Porto Alegre: Bookman, 2004.

CAMPO, M. S. **PROGRAMA PARA O CÁLCULO DA VARIAÇÃO DA DIREÇÃO DE INCIDÊNCIA DOS RAIOS SOLARES AO LONGO DO ANO**. COMBENGE 2013. Gramado - RS: [s.n.]. 2014.

CASIMIRO, M. H. **Pigmentos Minerais Sob Influência de Iluminação Fluorescente**. Ouro Preto: 2010.

DUFTON, A. F.; BECKETT, H. E. THE HELIODQN: AN INSTRUMENT FOR DEMONSTRATING. **Journal of Scientific Instruments**, ago. 1932. 2251-255.

FOLHAAZERO. Trabalhando com carta solar. 2008. Disponível em: < https://folhaazero.wordpress.com/2008/10/19/trabalhando-com-carta-solar/>. Acesso em: 22 mar. 2018.

GONZAGA, D. A. **DESENVOLVIMENTO E ANÁLISE DE DESEMPENHO DE SISTEMA SEGUIDOR SOLAR DE DOIS GRAUS DE LIBERDADE**. Ouro Preto: [s.n.], 2013.

GRIMM, A. M. Meteorologia Básica - Notas de Aula. **Departamento de Física - UFPR**, 1999. Disponivel em: <http://fisica.ufpr.br/grimm/aposmeteo/>. Acesso em: 29 jun. 2015.

HALLIDAY, D.; RESNICK, R.; JEARLWALKER. **Fundamentos de Física**. 7ª. ed. Rio de Janeiro: LTC, v. IV, 2007.

HELIODON. 2013. Disponível em <https://pt.wikipedia.org/wiki/Heliodon> Acesso em: 22 mar. 2018

NAUTILUS. Altura e azimute de um astro. 2018. Disponível em: http://nautilus.fis.uc.pt/astro/hu/movi/azimute.html. Acesso em: 22 mar. 2018.

NOGUEIRA, C. E. C. et al. Avaliação do conforto térmico nas residências convencional e inovadora do “Projeto CASA”, Unioeste, Campus de Cascavel. **Acta Scientiarum. Technology**, Maringá, 34, jan-mar 2012. 3-7.

UNIGRAN. Arquitetura adquire equipamento que ajuda a posicionar edificações. 2010. Disponível em <http://www.unigran.br/noticias/4078-arquitetura-adquire-equipamento-que-ajuda-a-posicionar-edificacoes> Acesso em: 22 mar. 2018.

ANEXO I – CÓDIGO DO CONTROLADOR

```
//pinos:
const int pinDirArco = 6;              //Define o sentido de rotação do arco
const int pinStepArco = 7;             //Envia o comando de movimento para o arco
const int pinDirMesa = 11;             //Define o sentido de rotação da mesa
const int pinStepMesa = 12;            //Envia o comando de movimento para a mesa

const int pinLamp = 8;                 //Alimenta a lâmpada.

const int  pinFCMesa = 10;             //Recebe o sinal da chave de final de curso da mesa
const int  pinFCArco = 9;              //Recebe o sinal da chave de final de curso do arco

//Constantes:
const long int meio_mesa = 1650;     //n de passos p que o arco se desloque até metade da mesa
const long int altoArc = 660;          //n de passos p que o arco se desloque até a altura máxima
const int v = 1000;                    //tempo de espera entre os passos

//Variaveis
int DMesa = 0;
int DArco = 0;
bool fimcurso1 = 0;
bool fimcurso2 = 0;
unsigned int index_Sender = 0;

void establishcontact()        //Informa à IU que a mesa está ociosa
{
        Serial.read();
        Serial.println('D');
        delay(500);
}
```

```
void zerarmesa()                //função que retorna a mesa ao ponto [0,0]
{
        digitalWrite(pinDirMesa, HIGH);
        while (!fimcurso1)
         {
                fimcurso1 = digitalRead(pinFCMesa);
                digitalWrite(pinStepMesa, HIGH);
                delayMicroseconds(v);
                digitalWrite(pinStepMesa, LOW);
                delayMicroseconds(v);
         }
        delay(100);

        digitalWrite(pinDirMesa, LOW);

        for (int j = 0; j < meio_mesa; j++)
         {
                digitalWrite(pinStepMesa, LOW);
                delayMicroseconds(v);
                digitalWrite(pinStepMesa, HIGH);
                delayMicroseconds(v);
        }
        delay(100);
        digitalWrite(pinDirArco, LOW);
        while (!fimcurso2)
        {
                fimcurso2 = digitalRead(pinFCArco);
                digitalWrite(pinStepArco, LOW);
                delayMicroseconds(v);
                digitalWrite(pinStepArco, HIGH);
                delayMicroseconds(v);
        }
        delay(100);
        digitalWrite(pinDirArco, HIGH);
        for (int j = 0; j < altoArc; j++)
        {
                digitalWrite(pinStepArco, LOW);
                delayMicroseconds(v);
                digitalWrite(pinStepArco, HIGH);
                delayMicroseconds(v);
        }
        delay(500);
```

```
        fimcurso1 = 0;
        fimcurso2 = 0;
        establishcontact();
}

void movemotor(int moveMesa, int moveArco)     //move os motores para a posição determinada
{
        int m;
        int a;

        if (moveMesa < 0)
        {
                digitalWrite(pinDirMesa, LOW);
                m = moveMesa * (-1);
        }
          else
        {
                digitalWrite(pinDirMesa, HIGH);
                m = moveMesa;
        }
        if (moveArco < 0)
        {
                digitalWrite(pinDirArco, LOW);
                a = moveArco * (-1);
        }
        else
        {
                digitalWrite(pinDirArco, HIGH);
                a = moveArco;
        }
        while (m  || a) {
                if (m)
                {
                        digitalWrite(pinStepMesa, HIGH);
                        m--;
                }
                if (a)
                {
                        digitalWrite(pinStepArco, HIGH);
                        a--;
                }
```

```
                delayMicroseconds(v);
                digitalWrite(pinStepMesa, LOW);
                digitalWrite(pinStepArco, LOW);
                delayMicroseconds(v);
                }
        establishcontact();
}

void setup()
{
        pinMode(pinDirArco, OUTPUT);
        pinMode(pinDirMesa, OUTPUT);
        pinMode(pinStepArco, OUTPUT);
        pinMode(pinStepMesa, OUTPUT);
        pinMode(pinLamp, OUTPUT);
        pinMode(pinFCArco, INPUT);
        pinMode(pinFCMesa, INPUT);
        digitalWrite(pinLamp, HIGH);
        Serial.begin(9600);
        establishcontact();
}

void loop()
{
        while (!Serial);
        {
        if (Serial.peek() == 'R')
                {
                        Serial.read();
                        zerarmesa();
                }
```

```
        else if (Serial.peek() == 'T')
        {
                Serial.read();
                establishcontact();
        }
        else if (Serial.peek() == 'M')
        {
                DMesa = Serial.parseInt();
                DArco = Serial.parseInt();
                delay(5);
        }

        if ((DMesa == 0) && (DArco == 0))
                establishcontact();
        else
        {
                movemotor(DMesa, DArco);
                DMesa = 0;
                DArco = 0;
        }
}
```

ANEXO II – CODIGO DA IU

```
using System;
using System.Threading.Tasks;
using System.IO.Ports;
using System.Windows.Forms;

namespace WindowsFormsApp2
{
    public partial class ControleMotor : Form
    {
        /*Constantes*/
        const double step_value = 0.1125;
        const double redu_mesa = 2.3194444444444444;

        /*Variaveis*/
        int J = 1;
        double lat = 0;
        double time = 12;
        bool flagSerialRdy = false;
        int PosAzi = 0;
        int PosAlt = 0;
        int AziAtual = 0;
        int AltAtual = 0;
        int AziAnt = 00;
        int AltAnt = 00;
        double Alt = 0;
        double Azi = 0;
        DateTime DataAtual;
        DateTime HoraAtual;
```

```
        public void Calcular_AH()
        {
            double horaS = (12 - time)*0.2618;
            double Jt = (((2*Math.PI) * (J - 80)) / 365);
            double Decl = 0.4093*Math.Sin(Jt);
            double latT = Math.PI*lat/180;
            Alt = Math.Asin((Math.Sin(Decl) * Math.Sin(latT)) + (Math.Cos(Decl) *
Math.Cos(latT) * Math.Cos(horaS)));

            if (time>12)
                Alt = Math.PI - Alt;

            if (Alt < 0 || Alt > Math.PI)
            {
                textAzi.Text = "Noite";
                textAlt.Text = "Noite";
            }
            else
            {
                Azi = Math.Acos(((((-1) * Math.Sin(Alt)) * Math.Sin(latT)) + Math.Sin(Decl)) /
(Math.Cos(Alt) * Math.Cos(latT) + 0.0000000001));
                Azi = Azi / Math.PI * 180;
                Alt = Alt / Math.PI * 180;
                textAzi.Text = Convert.ToString(Math.Round(Convert.ToDecimal(Azi),2));
                textAlt.Text = Convert.ToString(Math.Round(Convert.ToDecimal(Alt),2));
                label3.Text = Convert.ToString(Decl / Math.PI * 180);
            }
            return;
        }
```

```
public void Tune_data()
{
   if (Alt < 25)
         Alt = 25;
   if (Azi > 180)
   {
      Azi = Azi - 180;
      Alt = 180 - Alt;
   }
   if (Azi < 15)
   {
      Azi = 15;
      if (Alt < 69)
         Alt = 69;
     if (Alt > 140)
        Alt = 140;
  }
  else if (Azi < 25)
  {
     if (Alt < 35)
        Alt = 35;
     if (Alt > 145)
        Alt = 145;
  }
  else if (Azi > 175)
  {
     Azi = 175;
     if (Alt < 80)
```

```
            Alt = 80;
        if (Alt > 90)
            Alt = 90;
    }
    else if (Azi > 147)
    {
        if (Alt > 145)
            Alt = 145;
    }
    textAzi.Text = Convert.ToString(Azi);
    textAlt.Text = Convert.ToString(Alt);
}

public void Send_data()
    {
        try
        {
            Tune_data();
            AziAtual = Convert.ToInt16(((Azi-90) / step_value) * redu_mesa);
            AltAtual = Convert.ToInt16((Alt - 90) / step_value);
            PosAzi = AziAtual - AziAnt;
            PosAlt = AltAtual - AltAnt;
            if (PosAzi != 0)
                AziAnt = AziAtual;
            if (PosAlt != 0)
                AltAnt = AltAtual;
            label6.Text = Convert.ToString(PosAzi);
            label7.Text = Convert.ToString(PosAlt);
            serialPort1.WriteLine(String.Format("M{0}A{1}", PosAzi, PosAlt));
```

```
            label3.Text = (String.Format("M{0}A{1}", PosAzi, PosAlt));
            if (PosAlt > 0 || PosAzi > 0)
                flagSerialRdy = false;
        }
        catch (FormatException)
            MessageBox.Show("Valor invalido");
        catch (InvalidOperationException)
            MessageBox.Show("Por Favor, Conecte-se a mesa");
    }

    void GetAvaiiablePorts()
    {
        comboBox1.Items.Clear();
        String[] ports = SerialPort.GetPortNames();
        if (ports.Length == 0)
            comboBox1.Items.Add("(vazio)");
        else
            comboBox1.Items.AddRange(ports);
    }

    void ZerarData() {
        DataAtual = SelectData.Value;
        SelectData.Value = new System.DateTime(2001, 1, 1, 0, 0, 0, 0);
        Calcular_AH();
    }

    void ZerarHora()
    {
        HoraAtual = SelectData.Value;
```

```
    SelectHora.Value = new System.DateTime(2001, 1, 1, 0, 0, 0, 0);
    Calcular_AH();
}

public ControleMotor()
{
    InitializeComponent();
    Calcular_AH();
}

private void ControleMotor_Load(object sender, EventArgs e)
{
    timer1.Start();
}

private void Button1_Click(object sender, EventArgs e)
{
    if (!serialPort1.IsOpen)
    {
        MessageBox.Show("Por favor, conecte-se à mesa");
    }
    else if (textAzi.Text == "" || textAlt.Text == "")
    {
        MessageBox.Show("Informe o Azimute e a altura desejada.");
    }
    else if (textAzi.Text == "Noite" || textAlt.Text == "Noite")
    {
        MessageBox.Show("Sol se encontra abaixo da linha do horizonte.");
    }
```

```
        else if (!flagSerialRdy)
        {
            MessageBox.Show("mesaOucupada");
            serialPort1.WriteLine("T");
        }

        else if (serialPort1.IsOpen)
            Send_data();
        else
            MessageBox.Show("Erro Desconhecido");
    }

    private void Timer1_Tick(object sender, EventArgs e)
    {
        label9.Text = Convert.ToString(flagSerialRdy);
    }

    private void SerialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
    {
        if (serialPort1.ReadChar() == 'D')
            flagSerialRdy = true;
        serialPort1.ReadLine();
    }

    private void Sul_Click(object sender, EventArgs e)
    {
        Norte.Checked = false;
        Sul.Checked = true;
        if (lat > 0)
```

```
        lat = lat * (-1);
    Calcular_AH();
}

private void Norte_Click(object sender, EventArgs e)
{
    Sul.Checked = false;
    Norte.Checked = true;
    if (lat < 0)
        lat = lat * (-1);
    Calcular_AH();
}

private void SelectLat_ValueChanged(object sender, EventArgs e)
{
    lat = Decimal.ToDouble(SelectLat.Value);
    if (Sul.Checked)
        lat = lat * (-1);
    Calcular_AH();
}

private void SelectHora_ValueChanged(object sender, EventArgs e)
{
    double hora = SelectHora.Value.Hour;
    double min = SelectHora.Value.Minute;
    time= hora + (min / 60);
    Calcular_AH();
}
```

```
private void SelectData_ValueChanged(object sender, EventArgs e)
{
    J = SelectData.Value.DayOfYear;
    Calcular_AH();
}

private void Button2_Click(object sender, EventArgs e)
{
    ZerarHora();
    if (flagSerialRdy)
    {
        for (int i = 0; i <= 96; i++)
        {
            SelectHora.Value = SelectHora.Value.AddMinutes(15);
            SelectHora.Text = SelectHora.Value.ToString();
            Calcular_AH();
            if (textAzi.Text != "Noite")
            {
                Send_data();
                for (int j = 0; j < 1000; j++)
                {
                    if (!flagSerialRdy)
                        Task.Delay(10).Wait();
                }
                if (!flagSerialRdy)
                {
                    MessageBox.Show("Mesa não responde");
                    i = 96;
                    flagSerialRdy = true;
```

```
            }
        }
    }
}
else
{
    SelectHora.Value = HoraAtual;
    MessageBox.Show("mesaOucupada");
    serialPort1.WriteLine("T");
}
}

private void Button_Man_Click(object sender, EventArgs e)
{
    groupBox1.Enabled = false;
    groupBox2.Enabled = false;
    button2.Enabled = false;
    button3.Enabled = false;
    Button_Man.Enabled = false;

    Button_Aut.Enabled = true;
    textAzi.ReadOnly = false;
    textAlt.ReadOnly = false;

    textAzi.Text = "";
    textAlt.Text = "";
}

private void Button_Aut_Click(object sender, EventArgs e)
```

```
{
    groupBox1.Enabled = true;
    groupBox2.Enabled = true;
    button2.Enabled = true;
    button3.Enabled = true;
    Button_Man.Enabled = true;

    Button_Aut.Enabled = false;
    textAzi.ReadOnly = true;
    textAlt.ReadOnly = true;

    Calcular_AH();
}

private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    serialPort1.PortName = comboBox1.Text;
    serialPort1.Open();
}

private void ComboBox1_DropDown(object sender, EventArgs e)
{
    GetAvaiiablePorts();
}

private void Button_OpenPort_Click(object sender, EventArgs e)
{
    if (comboBox1.Text == "")
    {
```

```
        MessageBox.Show("Por favor selecione uma porta para a conexão");
    }
    else  if (comboBox1.Text == "(vazio)")
    {
        MessageBox.Show("Ver**ifique se o cabo da mesa está conectado");**
    }
    else
    {
        try
        {
            serialPort1.Open();
            comboBox1.Enabled = false;
            Button_OpenPort.Visible = false;
            Button_ClosePort.Enabled = true;
            Button_ClosePort.Visible = true;
            Task.Delay(100).Wait();
            try
            {
                serialPort1.Write("T");
            }
            catch (TimeoutException)
                MessageBox.Show("timeout");
        }
        catch (UnauthorizedAccessException)
            MessageBox.Show("Acesso Negado");
    }
}

private void Button_ClosePort_Click(object sender, EventArgs e)
```

```
{
    serialPort1.Close();
    comboBox1.Enabled = true;
    Button_OpenPort.Visible = true;
    Button_ClosePort.Enabled = false;
    Button_ClosePort.Visible = false;
}

private void Button3_Click(object sender, EventArgs e)
{
    ZerarData();
    if (flagSerialRdy)
    {
        for (int i = 0; i <= 72; i++)
        {
            SelectData.Value = SelectData.Value.AddDays(5);
            SelectData.Text = SelectData.Value.ToString();
            Calcular_AH();
            if (textAzi.Text != "Noite")
            {
                Send_data();
                for (int j = 0; j < 1000; j++)
                {
                    if (!flagSerialRdy)
                        Task.Delay(10).Wait();
                }
                if (!flagSerialRdy)
                {
                    MessageBox.Show("Mesa não responde");
```

```
                i = 72;
                flagSerialRdy = true;
            }
        }
    }
    else
    {
        SelectData.Value = DataAtual;
        MessageBox.Show("mesaOucupada");
        serialPort1.WriteLine("T");
    }
}

private void Button4_Click(object sender, EventArgs e)
{

    if (!serialPort1.IsOpen)
        MessageBox.Show("Por favor, conecte-se à mesa");

    else if (flagSerialRdy == false)
    {
    MessageBox.Show("mesaOucupada");
    serialPort1.WriteLine("T");
    }

    else
    {
        try
```

```
            {
                serialPort1.WriteLine("R");
                flagSerialRdy = false;
                AziAnt = 0;
                AltAnt = 0;
            }
            catch (InvalidOperationException)
                MessageBox.Show("Por favor, conecte - se à mesa");
    }
}

private void ComboBox1_SelectedIndexChanged_1(object sender, EventArgs e)
{
    serialPort1.PortName = comboBox1.Text;
}

private void TextAzi_TextChanged(object sender, EventArgs e)
{
    try
    {
        Azi = Convert.ToDouble(textAzi.Text);
    }
    catch (FormatException)
}

private void TextAlt_TextChanged(object sender, EventArgs e)
{
    try
    {
```

```
            Alt = Convert.ToDouble(textAlt.Text);
        }

        catch (FormatException
    }
  }
}
```

/* Os itens label3, label6, label7 e label9 são utilizados apenas com o intuito de informar ao desenvolvedor algumas variáveis ocultas durante o teste do softwares. Portanto, ambos estão configurados como ‘não visíveis’ na compilação final do código. */

ANEXO III – SUGESTÃO DE PCI