Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica

BRUNO VINÍCIUS DE ARAUJO CORREIA

Criação de uma Interface Gráfica e de uma Nova Porta de Comunicação para o Posicionador Automático de Antenas do LEMA/DEE

Campina Grande, Paraíba

Dezembro, 2015

BRUNO VINÍCIUS DE ARAUJO CORREIA

Criação de uma Interface Gráfica e de uma Nova Porta de Comunicação para o Posicionador Automático de Antenas do LEMA/DEE

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Área de Concentração: Eletrônica Embarcada

Orientador: Rômulo Raimundo Maranhão do Valle

Campina Grande, Paraíba

Dezembro, 2015

BRUNO VINÍCIUS DE ARAUJO CORREIA

Criação de uma Interface Gráfica e de uma Nova Porta de Comunicação para o Posicionador Automático de Antenas do LEMA/DEE

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Rômulo Raimundo Maranhão do Valle
Orientador

Campina Grande, Paraíba
Dezembro, 2015

Dedico este trabalho à minha família, a quem devo esse sonho realizado.

Agradecimentos

Se faz necessário agradecer as pessoas importantes que foram essenciais nessa fase da minha vida que termina com esse trabalho.

O primeiro agradecimento vai aos meus pais Wellington e Oneida, que são os pilares da minha formação. Exemplos de esforço, dedicação e afeto nos quais me espelho todos os dias.

Ao meu irmão Pedro Victor, pelo companheirismo mostrado nos momentos de altos e baixos ao longo de minha trajetória.

À minha companheira Renally Leal por todo amor compartilhado ao longo desses anos.

À todos os professores do departamento de engenharia elétrica pelos ensinamentos repassados ao longo do curso, em especial ao professor Rômulo Valle pela assistência prestada e pela oportunidade de estágio que me foi dada.

Aos meus amigos de curso Carlos Ângelo, Ravi Barreto, André Fiuza, Rodrigo Nicodemos, Pablo Silvestre, João Marcelo, Juliano Leal, Luciano Freires, Talles Gonçalves e André Guimarães pelos momentos de alegria e frustração e por todas as noites de estudos em que não seriam possíveis sem o incentivo um do outro. Amizades que serão levadas com carinho para sempre.

Ao recente amigo Kléber minha mais profunda gratidão pelos conhecimentos repassados, pelos equipamentos generosamente emprestados e pelas dicas que foram de importância vital para a conclusão bem sucedida desse estágio.

"Toda reforma interior e toda mudança para melhor dependem exclusivamente da aplicação do nosso próprio esforço."

- Immanuel Kant

Resumo

Este relatório apresenta os estudos efetuados para a criação de uma interface gráfica para o programa do posicionador automático de antenas, equipamento utilizado no Laboratório de Eletromagnetismo e Microondas Aplicados (LEMA) da Universidade Federal de Campina Grande para efetuar medições do diagrama de irradiação de antenas. Também apresenta as características da mudança da porta de comunicação utilizada entre o programa e o posicionador.

Palavras-chaves: Arduino, Qt Creator, Posicionador Automático de Antenas, Posicionador.

Abstract

This report presents a study made for the criation of a grafic user interface for the antenna automatic positioner program, device utilized in Applied Microwave and Electromagnetics Laboratory (LEMA) of the Federal University of Campina Grande to perform measurements of power diagrams in antennas. Also presents the port communication change used between the program and the positioner.

Keywords: Antenna automatic positioner, Arduino, Qt Creator, Positioner.

Lista de ilustrações

Lista de abreviaturas e siglas

DEE	Departamento de Engenharia Elétrica
LEMA	Laboratório de Eletromagnetismo e Microondas Aplicados
UFCG	Universidade Federal de Campina Grande

Sumário

1 Introdução

Este relatório possui a descrição das atividades desenvolvidas do dia 08/10/2015 ao dia 27/11/2015 referentes ao estágio supervisionado obrigatório para a conclusão do curso de graduação em engenharia elétrica pela Universidade Federal de Campina Grande. O estágio teve carga horária total de 182 horas, com carga semanal de 25 horas e foi efetuado no LEMA/DEE.

1.1 Contextualização

O LEMA é um laboratório para realização de experimentos e pesquisas na área de microondas e eletromagnetismo aplicado pertencente ao DEE da UFCG, dando suporte às disciplinas de graduação e pós-graduação dessa área. Esse laboratório serve tanto para ensino quanto para projetos de pesquisa e desenvolvimento e nele que se deu a realização do estágio.

Esse laboratório possui um equipamento responsável pela medição no diagrama de irradiação, denominado "posicionador automático de antena". O posicionador serve para auxiliar nas medições efetuadas tanto em experimentos da disciplina do laboratório de antenas do curso de graduação em engenharia elétrica da UFCG quanto em projetos de pesquisas efetuados no laboratório.

1.2 Motivação

Esse aparelho foi desenvolvido em 2002 e possui conexão com o computador do usuário que está efetuando as medições através de uma porta paralela, visto que nessa época era a melhor forma de conexão existente. O programa que controla o posicionador não possui interface gráfica, sendo uma aplicação de console ou terminal (ARAGãO, 2003).

Com a atual tecnologia existente, esse tipo de conexão se tornou obsoleta, havendo necessidade de uma atualização em relação à conexão utilizada. Também é vantajoso a atualização do programa com a implementação de uma interface gráfica para o usuário, visto que a plataforma se torna mais intuitiva e de fácil entendimento.

1.3 Objetivos Gerais e Específicos

O estágio realizado teve como objetivo o estudo de plataformas e dispositivos que pudessem auxiliar na implementação mudanças no posicionador que pudessem trazer

vantagens para sua utilização.

Como objetivos específicos é possível citar o estudo da arquitetura de computadores para a implementação de uma porta de conexão mais atual ao programa e o estudo de plataformas para o desenvolvimento de interfaces gráficas para aplicações do windows, sistema operacional utilizado nos computadores dos laboratórios. Também se fez necessário o estudo e a utilização do dispositivo Arduino, responsável pelo controle da rotação do motor de passo utilizado no pisicionador.

1.4 Metodologia

A primeira etapa do estágio consistiu em estudar e desenvolver o programa que controla o posicionador automático de antenas. Nessa parte foram estudadas várias opções de plataformas, entre elas o Visual Studio 2015 (SOLE, 2015). Optou-se pela utilização da plataforma Qt Creator, um ambiente de desenvolvimento integrado que oferece desenvolvimento de aplicações multi-plataforma de maneira rápida e fácil (BLANCHETTE; SUMMERFIELD, 2008).

A segunda etapa se deu através do estudo da arquitetura computacional para utilização de uma conexão entre o computador e o posicionador. Primeiramente, foi proposta a utilização de uma porta serial que fizesse essa conexão. Posteriormente foi escolhido para uso uma porta usb que simula uma porta serial, conectada com um dispositivo arduino que é responsável pelo movimento do aparelho (ŠVALJEK, 2015).

A última parte do estágio foi desenvolvida através de experimentos e testes que visavam comprovar a eficiência do programa e da conexão utilizada.

1.5 Organização do Relatório

O relatório apresentará na seção (2) a explicação minuciosa da constituição e do funcionamento do posicionador automático de antenas e na seção (3) descreverá como foi desenvolvida a criação da interface gráfica do programa. Na seção (4) serão utilizados os conhecimentos de arquitetura computacional para a descrição da porta de comunicação utilizada e como ela foi implementada. Na seção (5) serão comentados os resultados obtidos no laboratório, comparando a atual versão do sistema com a mais antiga. Na última seção (6), será concluído o relatório fazendo uma análise das atividades desenvolvidas e com propostas de idéias que poderão melhorar o equipamento.

2 Posicionador Automático de Antenas

Um dos experimentos realizados no laboratório consiste na medição do diagrama de irradiação de uma antena, através do uso de um posicionador automático de antenas apresentado na figura 1, dispositivo controlado através de um computador ligado à ele, e de um medidor de potência fixo em algum ponto do laboratório. O posicionador possui uma haste central onde a antena é fixada, e gira 360° em intervalos determinados pelo usuário (ARAGãO, 2003).

Figura 1 – Posicionador Automático de Antenas

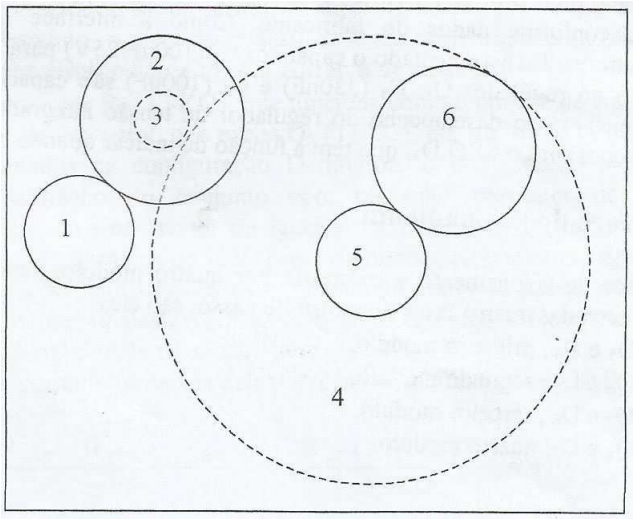
O programa que controla o dispositivo possui três opções de movimentação:

- Posicionamento Manual - Nessa opção o usuário entra com o valor do ângulo desejado, o posicionador vai girar até a posição do ângulo de entrada. Depois de finalizada a movimentação, o programa pede para o usuário entrar com o valor lido no medidor de potência;
- Posicionamento Automático - Nela o usuário digita o passo, em graus, desejado para que a antena descreva uma circunferência completa em intervalos iguais. A cada fim de movimentação o programa novamente pede ao usuário o valor lido no medidor de potência;

- Ajuste do Posicionador - Essa opção serve para ajustar o ponto de referência do posicionador. Possui duas opções, uma de avanço, onde ela vai avançar em relação ao ponto atual, e uma opção de atraso.

O equipamento utiliza um motor de passo de um antigo acionador de disquete de 5 1/4 para fazer a movimentação no eixo da haste instalada. Esse motor é do tipo híbrido com 200 passos/volta e cada fase do motor é acionada por vez para um correto sentido de rotação. Também é utilizado, uma redução mecânica apresentada na figura 2, que utiliza um conjunto de engrenagens fazendo a transferência do movimento do motor de passo até o eixo do posicionador. Essa redução tem como característica o aumento do torque e de sua resolução, permitindo uma maior precisão nas medições como também a utilização de antenas mais pesadas. Através dessas engrenagens, obteve-se um número de 43,21 passos do motor por ângulo no eixo da antena e um ângulo de 0,02314° no eixo da antena por passo do motor.

Figura 2 – Redução mecânica utilizada no posicionador



Através do estudo dessas informações, passou-se para a próxima etapa do estágio, que é a criação da interface gráfica do programa utilizado no posicionador automático de antenas.

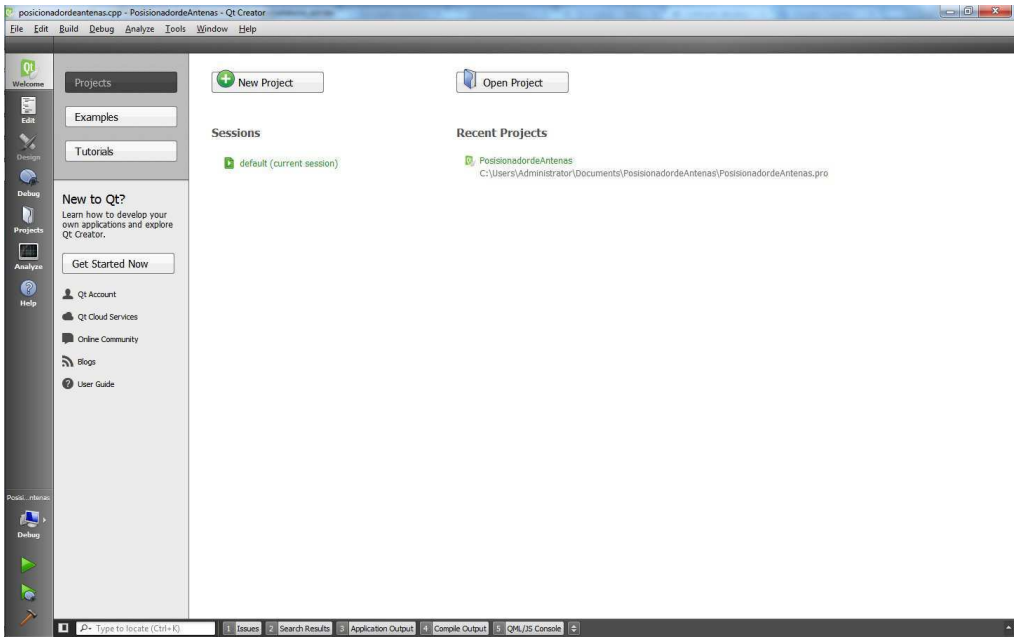
3 Interface Gráfica para o Programa

Para a criação da interface gráfica de *softwares*, várias são as opções de ferramentas e linguagens utilizadas apresentadas na literatura. A ferramenta escolhida para essa função foi o Qt Creator, que é implementada na linguagem C++ e de fácil entendimento, sendo seu funcionamento bastante intuitivo. A linguagem utilizada por ele possui grande compatibilidade com a linguagem C, utilizada no programa anterior, sendo necessário apenas uma adaptação do código.

3.1 Qt Creator

Qt Creator é um ambiente de desenvolvimento integrado multi-plataforma baseado em C++, JavaScript e QML (BLANCHETTE; SUMMERFIELD, 2008). Ele possibilita a criação de aplicativos nas mais diversas plataformas como Linux, Android e Windows. A figura 3 apresenta a janela inical do programa.

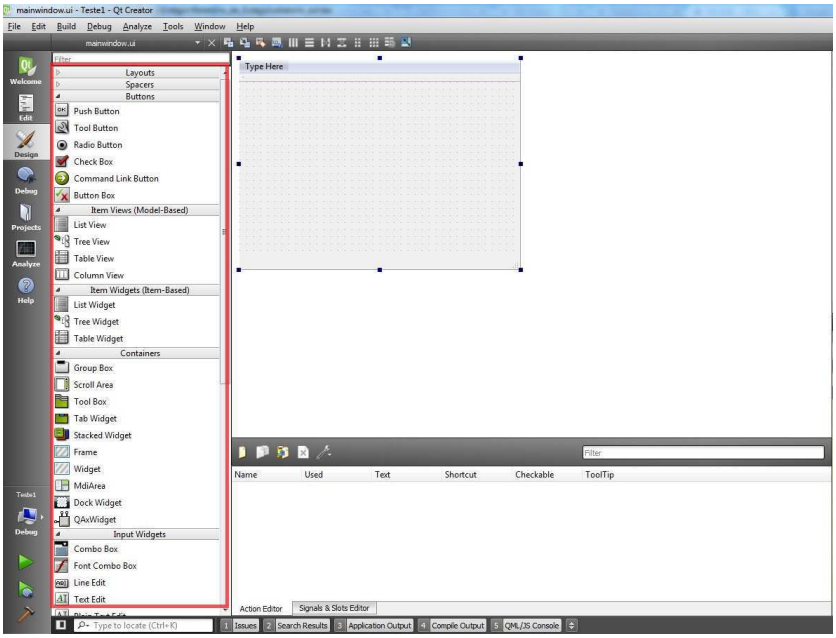
Figura 3 – Tela inicial do *software* Qt Creator



Para programação de aplicativos desenvolvidos no Windows, a ferramenta poderá utilizar dois tipos de compiladores, o MSVC (*Micrsoft Visual C++*) ou MinGW (*Minimalist GNU for Windows*). O compilador utilizado no desenvolvimento do programa do posicionador foi o MinGW, que vem embutido no aplicativo de instalação do próprio Qt Creator.

Para a construção das janelas que serão utilizadas no programa, o Qt Creator possui uma opção de "arrastar e soltar" blocos básicos como botões, marcadores, barra de progresso entre outros, como pode ser visto na parte destacada da figura 4. Essa versatilidade apresentada pelo programa facilita seu uso, sendo essa sua principal vantagem comparado com outras ferramentas.

Figura 4 – Opções de blocos utilizados nas janelas do programa



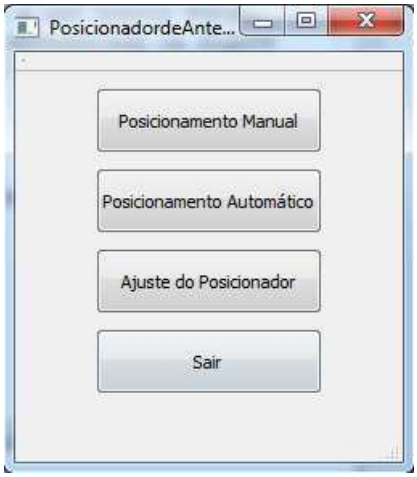
Em cada janela criada, uma classe a ela relacionada também é criada, sendo adicionadas as funções aplicadas nessa respectiva classe, seguindo uma programação orientada a objeto indicada na utilização da linguagem C++.

Outra vantagem do Qt Creator é a vasta documentação encontrada na internet (QT...,). Através do fórum do *site* principal do programa, podemos postar dúvidas e devido à grande difusão dessa plataforma para diversas aplicações, muitos usuários discutem e respondem as questões efetuadas. Também há uma grande diversidades de tutoriais em texto, como em vídeos mostrando os passos iniciais para o uso correto dessa ferramenta.

3.2 Programa Desenvolvido para o Posicionador

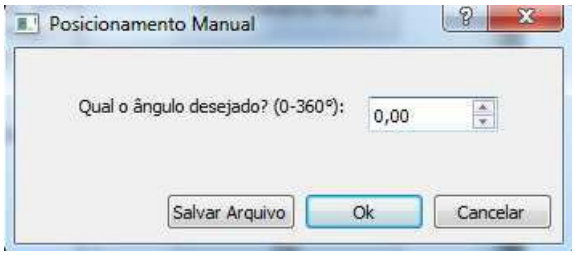
As etapas presentes no programa desenvolvido para o posicionador se mantiveram basicamente as mesmas, com exceção de algumas poucas modificações. Como já foi mencionado na seção (2), a tela inicial do programa possui 3 opções principais e uma quarta função para sair do programa. Essa tela é apresentada na figura 5.

Figura 5 – Tela inicial do programa do posicionador



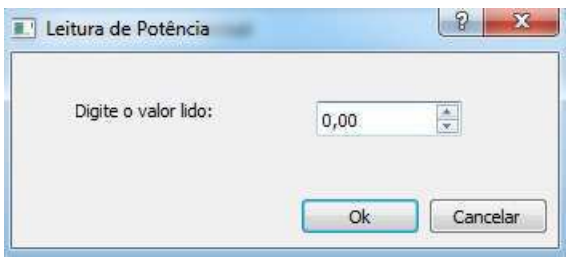
A primeira função do programa vai utilizar o posicionamento manual. Nela o usuário entrará com o valor específico, em graus, para que o posicionador efetue a rotação até esse determinado ponto. A janela responsável pela entrada desse valor é apresentada na figura 6.

Figura 6 – Janela do posicionamento manual



Após entrar com um valor entre 0 e 360 e clicar no botão ok, o programa abrirá uma nova janela, denominada leitura de potência. Essa janela ficará aberta enquanto o movimento estiver ocorrendo até o usuário entrar com o valor lido no medidor de potência quando o movimento da antena atingir o valor esperado. A figura 7 mostra essa janela. Depois que o usuário entra com o valor de potência, essa janela é fechada retornando para a janela da figura 6.

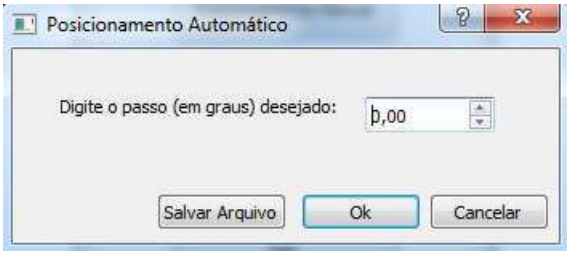
Figura 7 – Janela de leitura de potência para posicionamento manual



Em seguida, se o desejar, o usuário poderá salvar a informação acionando o botão "salvar arquivo".

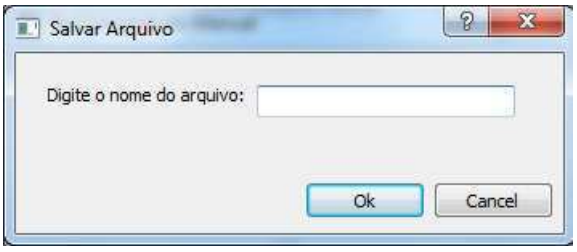
O funcionamento da opção de posicionamento automático é praticamente igual ao do posicionamento manual, sendo que o valor esperado pelo programa corresponde ao passo, também em graus, que o posicionador vai utilizar para fazer uma volta completa em torno do seu eixo. Por exemplo, se na primeira janela do posicionamento automático o usuário entrar com um valor de passos de 120, significa que a antena percorrerá 120° e abrirá a janela de leitura de potência. Depois de digitado o valor, ela percorrerá mais 120° e novamente pedirá um valor de potência a ser digitado pelo usuário. Por fim, depois de digitado o valor, ela percorrerá os últimos 120° e irá pedir um novo valor de potência, agora para o ângulo 360°. A janela do posicionamento automático é mostrada na figura 8.

Figura 8 – Janela do posicionamento automático



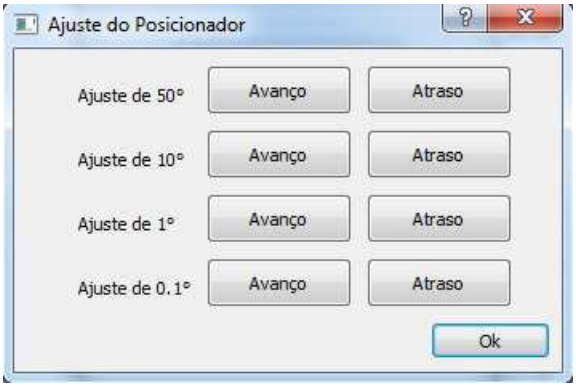
Nota-se que para as janelas de posicionamentos manual e automático, existe um botão denominado "salvar arquivo". Esse botão poderá ser clicado quando todas as medições estiverem efetuadas, quando será aberta uma nova janela apresentada na figura 9. Essa janela apresenta um bloco de entrada, onde é possível ao usuário digitar o nome do arquivo que será salvo com as informações das medições. A extensão do arquivo que mostra os dados coletados é .txt.

Figura 9 – Janela para opção de salvar os dados coletados



Como última opção do menu principal da figura 5, é apresentada a função de ajuste do posicionador. Essa função tem como objetivo posicionar a antena em teste no ponto de referência desejado, sendo esse o ponto 0° considerado pelo programa. Quando essa opção é selecionada, uma nova janela é aberta apresentando as opções de movimentos que poderão ser efetuados pelo posicionador, mostrado na figura 10.

Figura 10 – Janela de ajuste do posicionador



A opção de ajuste possui quatro possibilidades de movimento, sendo elas de ângulo 50, 10, 1 e 0.1 graus. Cada uma dessas possibilidades possuem mais duas opções, uma de avanço e uma de atraso, garantindo assim um ajuste mais fino se comparada com o programa anterior.

O código adaptado está apresentado em anexo, separado pelas funções utilizadas em cada janela ou funções que não tiveram necessidade de criação de uma janela.

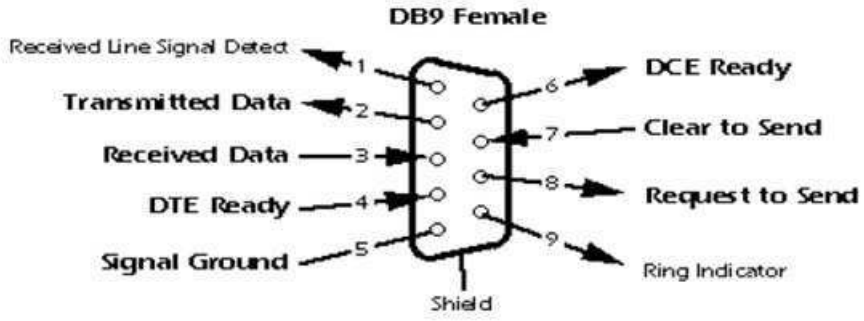
Depois de terminado o desenvolvimento do programa, foi estabelecida como meta o estudo e implementação da porta do computador que fará a comunicação entre o programa explicado e o equipamento utilizado.

4 Comunicação Utilizada no Posicionador

Como já foi mencionado na seção (1.2), o programa original utilizava a porta paralela do computador para conexão com o posicionador automático de antenas. Com a universalização da porta usb, esse tipo de conexão se tornou obsoleta, sendo que placas-mãe vendidas no mercado atualmente, não possuem mais esse tipo de conexão, sendo necessário a compra de adaptadores para essa função.

Primeiramente, foi proposta a idéia de utilização de uma porta serial para comunicação, apresentada na figura 11. Essa porta possui 9 pinos de conexão, sendo que apenas 1 pino é responsável pela transmissão, impossibilitando seu uso direto para o controle do posicionador, que necessita de 4 pinos para controlar as fases responsáveis pelo movimento do motor de passo.

Figura 11 – Pinos da porta serial

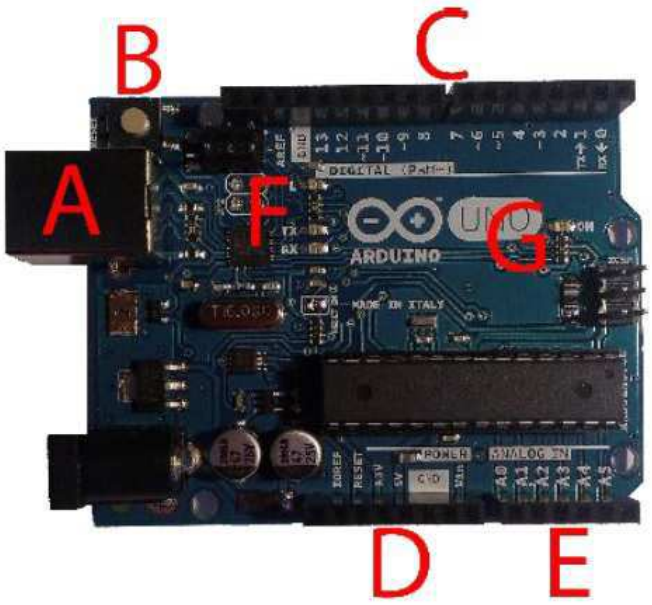


Posteriormente, viu-se a necessidade do uso de um dispositivo que fizesse o controle do motor, sendo escolhido um arduino para realizar essa função. Através dele, o computador simulará uma conexão serial utilizada pelo programa para transmitir as informações até o arduino, que será responsável por implementar a rotina de movimentação do motor a partir da quantidade de passos recebida. O placa do arduino, como suas funções utilizadas para essa aplicação serão descritas na subseção 4.1.

4.1 Descrição da Placa Arduino

Arduino é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão, a qual tem origem em Wiring, e é essencialmente C/C++ (ŠVALJEK, 2015). Pode ser usado para o desenvolvimento de objetos interativos independentes, ou ainda para ser conectado a um computador hospedeiro. A figura 12 mostra a placa arduino e seus elementos.

Figura 12 – Placa arduino e seus elementos



A placa do arduino é composta pelos seguintes elementos:

- **A** - Soquete do cabo usb. Essa entrada serve como alimentação da placa, conexão serial com o computador, que será utilizada pelo programa do posicionador, e para carregar a rotina que será executada por ele;
- **B** - Botão *Reset*. Este botão é usado para resetar a rotina do programa;
- **C** - Pinos digitais. 4 desses pinos serão utilizados para fazer o controle do motor de passo.
- **D** - Pinos de potência. São utilizados para alimentar algum circuito complementar que pode ser utilizado;
- **E** - Pinos analógicos de entrada e saída;
- **F** - Leds Tx/Rx que indicam que a placa se comunicando através da conexão usb;
- **G** - Led que indica que a placa está sendo alimentada e funcionando.

Para o funcionamento do posicionador, o arduino recebe, através da rotina do programa e da conexão usb, o número de passos necessários para o correto posicionamento. Essa quantidade pode ser positiva ou negativa, indicando que a haste do posicionador girará no sentido horário ou anti-horário. Depois de recebido esse valor, a rotina do arduino consiste em acionar os pinos digitais 8, 9, 10 e 11 na ordem correta para movimento em um sentido ou em outro.

Esses pinos digitais estão conectados aos pinos 2, 3, 4 e 5 do cabo de conexão paralela que é ligado ao posicionador, sendo estes pinos responsáveis pelas fases do motor de passo. Essa ligação possibilita uma longa distância entre o computador e o posicionador, visto que o cabo possui grande comprimento. Essa distância melhora os resultados, pois uma proximidade entre o computador e a antena em teste pode interferir na qualidade das medições.

A rotina implementada pelo arduino para receber o número de passos e aplicar a rotação no motor é apresentada em anexo. Uma análise dos resultados obtidos ao longo do estágio serão descritos em (5).

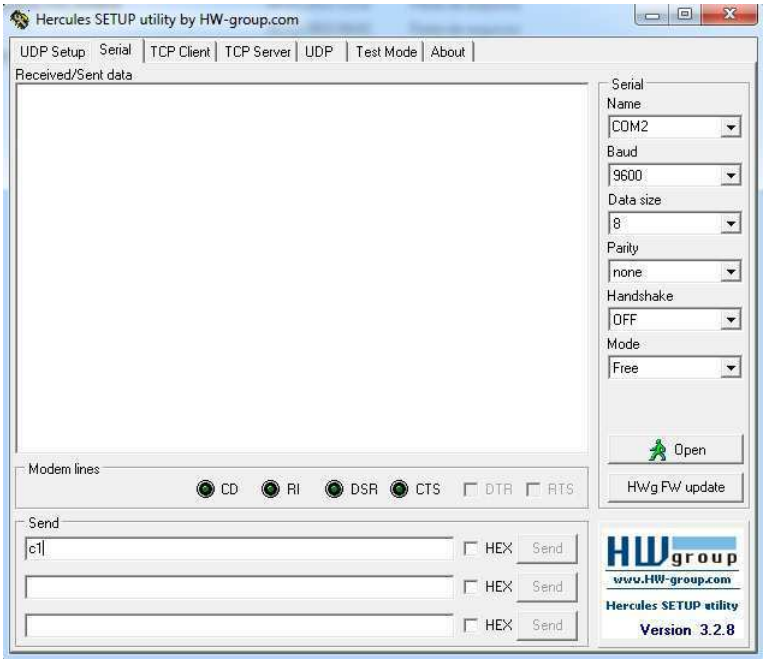
5 Resultados Obtidos no Laboratório

Primeiramente serão analisados os resultados obtidos pelo programa desenvolvido no Qt Creator. O estudo sobre essa plataforma foi de rápida evolução, sendo a parte mais demorada do estágio a escolha do programa que seria utilizado para a construção da interface gráfica.

Foram observados vários *bugs* no programa, sendo todos eles corrigidos ao longo do estágio. O programa está operando e foi testado em dois sistemas operacionais diferentes, o Windows 7 e Windows 10. A possibilidade do uso desse programa no Windows mais recente, aumenta sua longevidade em alguns anos, não sendo necessária uma nova adaptação em um curto período de tempo.

A conexão serial utilizada pelo programa para a comunicação com o arduino foi a parte mais trabalhosa dessa etapa. O programa está feito para utilizar a porta serial "COM2", tendo sido necessária uma configuração do dispositivo arduino para essa porta. Também se faz necessário, através do programa Hercúles, a abertura da porta antes da inicialização do programa, como mostrado no figura **??**. Com a porta aberta, o programa poderá ser rodado, abrindo a o menu inicial.

Figura 13 – Programa Hércules para Comunicação Serial



Ainda em relação ao programa, na pasta onde se encontra o aplicativo existe uma lista de arquivos .dll que são necessários para que o programa possa rodar no sistema operacional Windows.

Os testes efetuados no arduino foram feitos com o uso de um motor com 64 passos, de funcionamento similar ao do motor do posicionador. Esse motor é apresentado na figura 14. Depois de efetuado e conferido o funcionamento do programa com o arduino, só foi necessário modificar o número de passos da rotina do arduino pelo do motor do posicionador.

Figura 14 – Motor de passo utilizado nos testes do arduino

Por fim, foram feitos testes com o próprio posicionador, fase que ainda está em andamento, visto que ainda não se conseguiu uma perfeita movimentação do posicionador. A resolução desse problema deverá ser resolvida nos próximos dias, completando as metas estabelecidas para esse estágio.

6 Conclusão

Diversas cadeiras foram importantes para a conclusão bem sucedida desse estágio. Para o desenvolvimento do programa, foram utilizados vários conteúdos obtidos nas disciplinas de introdução a programação e técnicas de programação. Ainda em relação ao desenvolvimento do *software*, conclui-se que a plataforma Qt Creator é uma poderosa ferramenta para o desenvolvimento de aplicativos voltado para os mais diversos tipos de sistemas operacionais.

Os conceitos da disciplina de arquitetura de sistemas digitais foram utilizados no estudo da porta de comunicação e na aplicação da placa do arduino, dispositivo de fácil implementação e que possui inúmeras possibilidades de utilização.

Como idéias para um melhoramento do posicionador, podemos citar a possibilidade de leitura de potência automática através da saída analógica do leitor, podendo serem utilizadas as entradas analógicas do arduino para essa função.

O posicionador automático de antenas é um equipamento confiável e que facilita o processo de medições do diagrama de potência, processo muito trabalhoso se feito sem o seu auxílio.

Referências

ARAGãO, G. F. *Posicionador Automático de Antenas*. 2003. 12, 14

BLANCHETTE, J.; SUMMERFIELD, M. *C++ GUI Programming with Qt 4*. [S.l.]: Prentice Hall, 2008. 13, 16

QT Forum. <https://forum.qt.io/>. 17

SOLE, A. D. *Visual Studio 2015 Succinctly*. [S.l.]: Syncfusion, 2015. 13

ŠVALJEK, M. *Arduino Succinctly*. [S.l.]: Syncfusion, 2015. 13, 21

ANEXO A – Códigos

A.1 Código do arduino

```
#include <Stepper.h>

#define PASSOS 200

Stepper motor(PASSOS, 8, 9, 10, 11);
int val = 0;

void setup() {
  // put your setup code here, to run once:
  motor.setSpeed(10);
  Serial.begin(9600);
}

void loop() {

  if (Serial.available() > 0) {
    if (Serial.peek() == 'c') { //check for the character that signifies that this
      Serial.read(); //remove the character that signifies this is a command from t
      val = Serial.parseInt(); //store our expected integer into state
    }
    while (Serial.available() > 0){ //Discard everything that we didn't expect
      Serial.read();
    }
    // step one revolution  in one direction:
    Serial.println("Girando");
    motor.step(val);
    delay(500);

  }
}
```

A.2 Códigos do Qt Creator

A.2.1 avancoatrasoaut.cpp

```
#include "avancoatrasoaut.h"
#include <QDebug>
#include <QSerialPort>
#include <QSerialPortInfo>
#include "global.h"
#include "QString"

QSerialPort serial;

AvancoAtrasoAut::AvancoAtrasoAut()
{

}

AvancoAtrasoAut::~AvancoAtrasoAut()
{

}

void AvancoAtrasoAut::AvancoAut()
{
    QString y;
    y.setNum(passo);
    QByteArray x(y.toStdString().c_str());
    x = "c" + x;

    serial.setPortName("COM2");
    serial.setBaudRate(QSerialPort::Baud9600);
    serial.setDataBits(QSerialPort::Data8);
    serial.setParity(QSerialPort::NoParity);
    serial.setStopBits(QSerialPort::OneStop);
    serial.setFlowControl(QSerialPort::NoFlowControl);

    serial.open(QIODevice::ReadWrite);
    serial.write(x);
    serial.QSerialPort::waitForBytesWritten(2000);
```

```
    serial.close();


    //else{
    //    qDebug() << serial.errorString();}

    return;
}
```

A.2.2 avancoatrasoman.cpp

```
#include "avancoatrasoman.h"
#include <QDebug>
#include <QSerialPort>
#include <QSerialPortInfo>
#include "global.h"
#include "QString"

QSerialPort serial2;

AvancoAtrasoMan::AvancoAtrasoMan()
{

}

AvancoAtrasoMan::~AvancoAtrasoMan()
{

}

void AvancoAtrasoMan::AvanAtraMan()
{
    QString y;
    y.setNum(posfinal);
    QByteArray x(y.toStdString().c_str());
    x = "c" + (x);

    serial2.setPortName("COM2");
```

```
    serial2.setBaudRate(QSerialPort::Baud9600);
    serial2.setDataBits(QSerialPort::Data8);
    serial2.setParity(QSerialPort::NoParity);
    serial2.setStopBits(QSerialPort::OneStop);
    serial2.setFlowControl(QSerialPort::NoFlowControl);

    serial2.open(QIODevice::ReadWrite);
    serial2.write(x);
    serial2.QSerialPort::waitForBytesWritten(2000);
    serial2.close();

    //else{
    //    qDebug() << serial2.errorString();}
    return;
}
```

A.2.3 conversao.cpp

```
#include "conversao.h"
#include "global.h"

//Essa rotina faz a conversão dos valores digitados em ângulos
//para a quantidade de passos necessária

Conversao::Conversao()
{


}

//Conversão para cálculo do número de passos
Conversao::~Conversao()
{
    tam_passo = 360.0/15558.0;
    tam_angulo = 15558.0/360.0;

    //Para posicionamento automático
    qtd_avan = 360/deltang;
    passo_auto = (deltang * tam_angulo);
```

```
    //Para posicionamento manual
    passo_man = (posicao * tam_angulo);
    return;
}
```

A.2.4 global.cpp

```
#include "global.h"

Global::Global()
{

}

Global::~Global()
{

}

//Declaração das variáveis globais
extern float angulo[360] = {0}, potencia[360] = {0};
extern float tam_passo=0, tam_angulo=0, posicao=0, posatual=0;
extern unsigned long qtd_passo=0, passo=0, passo_auto=0, passo_man=0;
extern int tempo=30, posinicial=0, posfinal=0, m=0;
extern float qtd_avan=0, deltang=1;
extern int k=0, acum_passo=0;

/*Descrição das variáveis
angulo       -
potencia     -
tam_passo    - O avanço em graus de cada passo
tam_angulo   -
posicao      - Ângulo absoluto
posatual     - Posição absoluta do posicionador no momento
qtd_passo    -
passo        - Quantidade de passos necessária para o deslocamento
passo_auto   -
```

```
passo_man    -
tempo        -
posinicial   - Ângulo absoluto anterior para calculo do deltangulo
posfinal     - Ângulo absoluto desejado para calculo do deltangulo
qtd_avan     - Quantidade de leituras que serão efetuadas em uma volta
deltang      - Ângulo correspodente a variação do avanço (passo)
acum_passo   -

*/
```

A.2.5 leituradepotencia.cpp

```
#include "leituradepotencia.h"
#include "ui_leituradepotencia.h"
#include "global.h"

LeituradePotencia::LeituradePotencia(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::LeituradePotencia)
{
    ui->setupUi(this);

    //Intervalo de valores admitidos na leitura de potência
    ui->doubleSpinBox->setRange(-100,200);
}

LeituradePotencia::~LeituradePotencia()
{
    delete ui;
}

void LeituradePotencia::on_pushButton_clicked()
{
    //Leitura do valor de potência
    potencia[k] = ui->doubleSpinBox->value();
    angulo[k] = posicao;
    this->close();
}
```

A.2.6 leituradepotencia2.cpp

```
#include "leituradepotencia2.h"
#include "ui_leituradepotencia2.h"
#include "global.h"

LeituradePotencia2::LeituradePotencia2(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::LeituradePotencia2)
{
    ui->setupUi(this);

    //Intervalo de valores admitidos na leitura de potência
    ui->doubleSpinBox->setRange(-100,200);
}

LeituradePotencia2::~LeituradePotencia2()
{
    delete ui;
}

void LeituradePotencia2::on_pushButton_clicked()
{
    //Leitura do valor de potência
    potencia[m] = ui->doubleSpinBox->value();
    angulo[m] = posicao;
    this->close();
}
```

A.2.7 main.cpp

```
#include "posicionadordeantenas.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    PosicionadordeAntenas w;
    w.show();
```

```
    return a.exec();
}
```

A.2.8 posicionadordeantenas.cpp

```
#include "posicionadordeantenas.h"
#include "ui_posicionadordeantenas.h"
#include "posicionadormanual1.h"
#include "posicionadorautomatico1.h"
#include "zerarposicionador.h"

PosicionadordeAntenas::PosicionadordeAntenas(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::PosicionadordeAntenas)
{
    ui->setupUi(this);
}

PosicionadordeAntenas::~PosicionadordeAntenas()
{
    delete ui;
}

//Seleciona a opção de posicionamento manual
void PosicionadordeAntenas::on_pushButton_2_clicked()
{
    PosicionadorManual1 posman;
    posman.setModal(true);
    posman.exec();
}

//Seleciona a opção de posicionamento automático
void PosicionadordeAntenas::on_pushButton_3_clicked()
{
    PosicionadorAutomatico1 posaut;
    posaut.setModal(true);
    posaut.exec();
}
```

```
//Seleciona a opção de ajuste do posicionador
void PosicionadordeAntenas::on_pushButton_4_clicked()
{
    ZerarPosicionador zerpos;
    zerpos.setModal(true);
    zerpos.exec();
}
```

A.2.9 posicionadormanual1.cpp

```
#include "posicionadormanual1.h"
#include "ui_posicionadormanual1.h"
#include "global.h"
#include "leituradepotencia.h"
#include "conversao.h"
#include "salvararquivo.h"
#include "avancoatrasoman.h"

PosicionadorManual1::PosicionadorManual1(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::PosicionadorManual1)
{
    //Variáveis
    k=0;
    passo=0;
    m = 0;

    ui->setupUi(this);

    //Intervalo de valores de ângulo
    ui->doubleSpinBox->setRange(0,360);
}

PosicionadorManual1::~PosicionadorManual1()
{
    delete ui;
}

void PosicionadorManual1::on_pushButton_clicked()
```

```
{
    posinicial = passo;

    //Entrada da posição desejada
    posicao = ui->doubleSpinBox->value();

    //conversão de posição desejada para quantidade de passos
    Conversao conv;
    conv.~Conversao();

    passo=passo_man;
    posfinal=passo-posinicial;

    //Execução do movimento
    AvancoAtrasoMan x;
    x.AvanAtraMan();

    LeituradePotencia leipot;
    leipot.setModal(true);
    leipot.exec();

    k++;
}

void PosicionadorManual1::on_pushButton_3_clicked()
{
    SalvarArquivo arq;
    arq.setModal(true);
    arq.exec();
}
```

A.2.10 posicionadorautomatico1.cpp

```
#include "posicionadorautomatico1.h"
#include "ui_posicionadorautomatico1.h"
#include "global.h"
#include "leituradepotencia2.h"
#include "conversao.h"
#include "salvararquivo2.h"
```

```
#include "avancoatrasoaut.h"
#include "QTimer"

PosicionadorAutomatico1::PosicionadorAutomatico1(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::PosicionadorAutomatico1)
{
    ui->setupUi(this);

    //Intervalo de valores de ângulo
    ui->doubleSpinBox->setRange(0,360);

    //Variáveis

}

PosicionadorAutomatico1::~PosicionadorAutomatico1()
{
    delete ui;
}

void PosicionadorAutomatico1::on_pushButton_clicked()
{
    m = 0;

    //Entrada de dados
    deltang = ui->doubleSpinBox->value();

    //Conversão de variáveis
    Conversao conv;
    conv.~Conversao();

    passo=passo_auto;

    AvancoAtrasoAut x;

    //Execução da rotina
    for(m=0;m<qtd_avan;m++)
    {
```

```
        posicao=posicao+deltang;

        //função(avanço)
        x.AvancoAut();

        LeituradePotencia2 leipot;
        leipot.setModal(true);
        leipot.exec();
    }

    k=qtd_avan;
}

void PosicionadorAutomatico1::on_pushButton_3_clicked()
{
    SalvarArquivo2 arq;
    arq.setModal(true);
    arq.exec();
}
```

A.2.11 salvararquivo.cpp

```
#include "salvararquivo.h"
#include "ui_salvararquivo.h"
#include "global.h"
#include <QString>
#include <QFile>
#include <QTextStream>

SalvarArquivo::SalvarArquivo(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::SalvarArquivo)
{
    ui->setupUi(this);
}

SalvarArquivo::~SalvarArquivo()
{
    delete ui;
```

```
}

void SalvarArquivo::on_pushButton_clicked()
{
    writefile();
}

void SalvarArquivo::writefile()
{
    QString nomearq = ui->lineEdit->text();
    nomearq = nomearq + ".txt";
    QFile file(nomearq);
    file.open(QIODevice::WriteOnly|QIODevice::Text);
    QTextStream out(&file);
    QString x;
    out<<"Arquivo gerado pelo Posicionador de Antenas\n";
    out<<"Desenvolvimento orientado por:\n";
    out<<"Professor Rômulo Raimundo Maranhão do Valle\n";
    out<<"Desenvolvido por Galba Falcao Aragao (Agosto de 2002)\n";
    out<<"Interface grafica feita por Bruno Vinicius (Novembro de 2015)\n\n";
    out<<"Angulo(Graus)    Potencia(dB)\n";

    int b;
    for(b=0;b<k;b++)
    {
        x = QString::number(angulo[b]);
        out<<x<<"                ";
        x = QString::number(potencia[b]);
        out<<x<<"\n";
        angulo[b] = 0;
        potencia[b] = 0;
    }
    k = 0;
    file.close();
    this->close();
}
```

A.2.12 salvararquivo2.cpp

```
#include "salvararquivo2.h"
#include "ui_salvararquivo2.h"
#include "global.h"
#include <QString>
#include <QFile>
#include <QTextStream>

SalvarArquivo2::SalvarArquivo2(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::SalvarArquivo2)
{
    ui->setupUi(this);
}

SalvarArquivo2::~SalvarArquivo2()
{
    delete ui;
}

void SalvarArquivo2::on_pushButton_clicked()
{
    writefile();
}

void SalvarArquivo2::writefile()
{
    QString nomearq = ui->lineEdit->text();
    nomearq = nomearq + ".txt";
    QFile file(nomearq);
    file.open(QIODevice::WriteOnly|QIODevice::Text);
    QTextStream out(&file);
    QString x;
    out<<"Arquivo gerado pelo Posicionador de Antenas\n";
    out<<"Desenvolvimento orientado por:\n";
    out<<"Professor Rômulo Raimundo Maranhão do Valle\n";
    out<<"Desenvolvido por Galba Falcao Aragao (Agosto de 2002)\n";
    out<<"Interface grafica feita por Bruno Vinicius (Novembro de 2015)\n\n";
```

```
    out<<"Angulo(Graus)    Potencia(dB)\n";

    int b;
    for(b=0;b<m;b++)
    {
        x = QString::number(angulo[b]);
        out<<x<<"                ";
        x = QString::number(potencia[b]);
        out<<x<<"\n";
        angulo[b] = 0;
        potencia[b] = 0;
    }

    m = 0;
    posicao = 0;

    file.close();
    this->close();
}
```

A.2.13 zerarposicionador.cpp

```
#include "zerarposicionador.h"
#include "ui_zerarposicionador.h"
#include "global.h"
#include "avancoatrasoman.h"
#include "conversao.h"

ZerarPosicionador::ZerarPosicionador(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::ZerarPosicionador)
{
    ui->setupUi(this);
}

ZerarPosicionador::~ZerarPosicionador()
{
    delete ui;
}
```

```
void ZerarPosicionador::on_pushButton_clicked()
{
    this->close();
}

void ZerarPosicionador::on_pushButton_2_clicked()
{
    posicao = 50;

    Conversao conv;
    conv.~Conversao();

    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}


void ZerarPosicionador::on_pushButton_4_clicked()
{
    posicao = 10;

    Conversao conv;
    conv.~Conversao();

    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}
```

```
void ZerarPosicionador::on_pushButton_5_clicked()
{
    posicao = 1;

    Conversao conv;
    conv.~Conversao();

    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}


void ZerarPosicionador::on_pushButton_6_clicked()
{
    posicao = 0.1;

    Conversao conv;
    conv.~Conversao();

    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}


void ZerarPosicionador::on_pushButton_3_clicked()
{
    posicao = -50;

    Conversao conv;
    conv.~Conversao();
```

```
    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}


void ZerarPosicionador::on_pushButton_7_clicked()
{
    posicao = -10;

    Conversao conv;
    conv.~Conversao();

    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}

void ZerarPosicionador::on_pushButton_8_clicked()
{
    posicao = -1;

    Conversao conv;
    conv.~Conversao();

    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}
```

```
void ZerarPosicionador::on_pushButton_9_clicked()
{
    posicao = -0.1;

    Conversao conv;
    conv.~Conversao();

    posfinal = passo_man;

    AvancoAtrasoMan x;
    x.AvanAtraMan();

    posicao = 0;
}
```