CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Universidade Federal
de Campina Grande

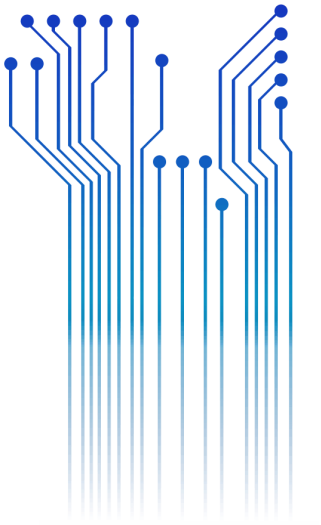
RAYMUNDO DE AMORIM JUNIOR

Centro de Engenharia
Elétrica e Informática

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA DE MEDIÇÃO AUTOMÁTICO DE PADRÃO DE IRRADIAÇÃO DE ANTENAS

Departamento de
Engenharia Elétrica



Campina Grande
2017

RAYMUNDO DE AMORIM JUNIOR

SISTEMA DE MEDIÇÃO AUTOMÁTICO DE PADRÃO DE RADIAÇÃO DE ANTENAS

Trabalho de Conclusão de Curso 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 Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Telecomunicações

Orientador:
Professor Dr. Glauco Fontgalland

Campina Grande

2017

RAYMUNDO DE AMORIM JUNIOR

SISTEMA DE MEDIÇÃO AUTOMÁTICO DE PADRÃO DE IRRADIAÇÃO DE ANTENAS

Trabalho de Conclusão de Curso 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 Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Telecomunicações

Aprovado em ____ / ____ / _____

Professor Avaliador
Universidade Federal de Campina Grande
Rômulo Raimundo Maranhão do Valle

Professor Dr. Glauco Fontgalland
Universidade Federal de Campina Grande
Orientador, UFCG

AGRADECIMENTOS

Agradeço a Deus, em primeiro lugar, por poder me proporcionar estes momentos com saúde.

Gostaria de agradecer também aos meus familiares, minha mãe Rita de Cássia Amorim, meu pai Raimundo de Amorim e meus irmãos e em especial a Rayan e Luana que tanto me apoiaram e me dão força para seguir em frente.

Aos meus colegas com quem tanto vivenciei momentos difíceis mas também os raros momentos de diversão, gostaria em especial citar o nome de meus amigos Frank Wesley e Suelson Lopes que participaram de forma direta e indireta nesta longa jornada nos momentos de stress e alegria.

Além do companheiro de LEMA Galba o qual me ajudou bastante neste TCC dedicando alguns domingos a este projeto.

Dentre os diversos ótimos professores os quais tive oportunidade de vivenciar nesta graduação, gostaria de citar o nome de uma das pessoas que mais participaram da minha formação na graduação com seus ensinamentos e sempre com paciência e humildade no ato de ensinar e exemplificar suas experiências agradeço ao Prof. Glauco por toda sua compreensão e coleguismo com todos que o cercam.

RESUMO

Este trabalho visa a integração de diferentes partes de um sistema de medição por meio da conexão lógica através do Matlab. Tendo em vista que no Laboratório de Antenas e Sensoriamento - LASen possuímos equipamentos que são pouco utilizados ou sub-utilizados buscamos com este trabalho a operabilidade desses dispositivos que apesar de possuírem um tempo de uso elevado se encontram em boa condição de utilização e robustez. Além do que são de aplicação importante na medição de diagramas de irradiação de antenas.

Visando a integração foram buscados equipamentos e padrões de comunicações o qual pudessem ser utilizados com o Matlab. O Arduíno possui uma integração facilitada com o Matlab visto que existem diversas bibliotecas disponíveis, facilitando a integração. O GPIB já é largamente utilizado como talvez o principal padrão de comunicação entre equipamentos.

Foram buscadas comparações com preços de sistemas no mercado com essas características como demonstrado valores na ordem de dezenas de euros como visto em [5] e com este projeto visaremos a redução deste valor.

Palavras-chave: GPIB, Arduíno, VNA, Medição antenas.

ABSTRACT

This work aims to integrate different parts of a measurement system through the logical connection with Matlab. Considering that in LabSen we have equipment that is underused, we seek with this work the operability of these devices that despite having a high usage time are in good condition of use and robustness.

In order, Arduino has an easy integration with Matlab, since there are several libraries available, making integration easier. The GPIB is already widely used as perhaps the main standard of communication between equipment. Comparisons with prices of systems in the market with these characteristics were sought, the price in general arrives to dozens of euros as seen in [5], and with this project we intend to reduce this amount.

Keywords: GPIB, VNA, Arduíno, Antennas.

LISTA DE ILUSTRAÇÕES

Figura 1: Conector USB-GPIB. 15
Figura 2: Tabela comparativa entre diferentes padrões de conexão com equipamentos....18
Figura 3: Placa Arduíno UNO 20
Figura 4: Diagrama de iRradiação para diversos tipos de antenas.....22
Figura 5: Exemplo do ganho de uma antena.....22
Figura 6: Tipos de polarizações em antenas. 23
Figura 7: Interconexão entre os dispositivos do sistema de medição.....23
Figura 8: Rotina de execução do sistema de medição. 24
Figura 9: Conexão do VNA e Arduíno ao computador.....24
Figura 10: Rotação da antena em relação ao seu eixo, (a) Eixo vertical, (b) Eixo horizontal. 25
Figura 11: Conexão do modulo relé com arduíno.....25
Figura 12: Posicionamento vertical proporcionado pelo motor 1.....26
Figura 13: VNA 8753/ET disponível no LabSen.....27
Figura 14: VNA com DST.....28
Figura 15: Coeficiente de trasmissão e reflexão adquiridos por meio da interface GPIB.....28
Figura 16: Fase dos coeficientes S11 e S21.....29
Figura 17: Carta de Smith do coeficiente S11. 29
Figura 18: Circuito de comando para acionamento dos motores. 30
Figura 19: Interface para configuração dos deslocamentos da antena 30
Figura 20: Configuração final do sistema de medição automático. 32

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ASCII	American Standard Code for Information and Interchange
dBi	Decibel em relação antena isotrópica
DST	Dispositivo sobre teste
GPIB	General Purpose Inteface Bus
HP-IB	Hewlett-Packard-Interface Bus
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
LAN	Local Area Network
LASen	Laboratório de Sensoriamento
MB	Mega Bytes
PC	Personal Computer
PWM	Pulse Width Modulation
SCPI	Standard Commands for Programmable Instruments
TTL	Transistor-to-Transistor Logic
USB	Universal Serial Bus
VISA	Virtual Instrument Software Architecture
VNA	Vector Network Analyser

SUMÁRIO

Agradecimentos vi
Resumo vii
Abstract viii
Lista de Ilustrações ix
Lista de Abreviaturas e Siglas x
Sumário xi
1 Introdução 12
2 Objetivos 13
3 Motivação 14
4 Desenvolvimento 14
4.1 GPIB 14
4.1.1 Largura de banda 15
4.1.2 Latência 16
4.1.3 Comunicação Baseada em Mensagem versus Comunicação Baseada em Registro 16
4.1.4 Configuração do Instrumento e Desempenho do Software 17
4.1.5 Robustez do Conector 17
4.1.6 Utilização GPIB 17
4.2 SCPI 18
4.2.1 História SCPI 18
4.2.2 Descrição SCPI 19
4.3 Arduíno 19
4.3.1 Hardware 20
4.4 Fundamentos de Antenas 21
4.4.1 Padrão de irradiação 21
4.4.2 Ganho 22
4.4.3 Polarização da antena 22
5 Resultados e discussões 23
5.1 Conexão, Computador – VNA 26
5.2 Conexão Computador – Arduíno 29
5.3 Interconexão do Sistema 31
6 Conclusão 33
Referências 34
Apêndice A 35

1 INTRODUÇÃO

Qualquer equipamento utilizado em tecnologias de comunicações sem fios (Wireless) necessitam de antenas que consigam emitir e receber a informações que são transmitidas pela propagação dos respectivos campos eletromagnéticas. Para isso é necessário dispor de sistemas capazes de caracterizar os mais diversos tipos de dispositivos irradiantes que venham a ser utilizados em qualquer sistema de comunicações, sejam eles para radiodifusão, telefonia móvel, sensoriamento, rastreio ou conectividade entre equipamentos.

De todas as medições de antena consideradas, o padrão de irradiação é o mais exigente nos passos de medição e mais difícil de interpretar. Uma antena pode irradiar até certo ponto em todas as direções do espaço que a rodeia ou não. Portanto, o diagrama de irradiação de uma antena é uma representação tridimensional da magnitude, fase e polarização do campo eletromagnético. Na maioria dos casos, a distribuição espacial da radiação eletromagnética é determinada em apenas dois planos principais de interesse.

Sem um sistema de posicionamento automático, torna-se por vez, uma tarefa árdua verificar todas as condições de irradiação de uma antena. Como diversos passos repetitivos para cobrir a revolução nos dois planos são necessários na execução manual desta tarefa, isto facilmente conduz a erros de leitura das escalas, de posicionamento espacial, de regularidade dos passos, na ortogonalidade dos planos e altura. Deve-se considerar ainda que sistemas de medição tridimensionais são extremamente especializados e disponíveis apenas em centros avançados. A grande maioria das câmaras anecóicas existentes, e principalmente no Brasil, não dispõem de um sistema automatizado como descrito. Algumas câmaras anecóicas são equipadas apenas com mesas giratórias, ou seja, possuem um sistema para realização apenas a rotação em um plano (horizontal).

Para obter resultados mais precisos e ao mesmo tempo reduzir o esforço, vislumbramos realizar neste projeto uma interface gráfica para automatização e visualização do diagrama de irradiação ou outra medida disponível em tempo real.

2 OBJETIVOS

O objetivo geral deste trabalho é implementar um sistema de posicionamento automatizado para estimação de parâmetros relativos a irradiação de antenas, apresentado ainda os seguintes objetivos específicos.

• Verificar os diagramas de irradiação em diferentes planos e polarizações.

• Implementar o controle automático dos passos de elevação e giro da antena no posicionador.

• Utilizar a ferramenta Matlab para criação de uma interface gráfica e controle do movimento das antenas bem como visualização gráfica dos resultados obtidos.

• Realizar a programação da porta GPIB a fim de interligar o VNA (Vector Network Analyser), com a interface gráfica no Matlab.

• Realizar a análise comparativa entre diferentes tipos de diagramas de irradiação de diferentes antenas em diferentes planos de medida.

A Metodologia para desenvolver este projeto é baseada no estudo dos principais conceitos envolvendo teoria de antenas e diferentes linguagens de programação com o objetivo de realizar a interconexão com o Matlab e a automatização do processo. Em seguida, será realizada uma revisão bibliográfica sobre o tema de antenas, funcionamento do VNA, programação da porta GPIB e programação no ambiente Matlab. O projeto terá duração de 4 meses e compreenderá a execução das seguintes atividades:

1. Com o intuito de prover um embasamento teórico para confirmação dos padrões apresentados pelas antenas, estudo da fonte e emissão e sistema de transmissão Como também será necessária a abordagem de técnicas de integração de diferentes dispositivos.
2. Sistema de recepção e validação dos dados.
3. Sistema de posicionamento, por meio da programação de um Arduíno a fim de realizar comandos automáticos orientando a posição da antena por meio do acionamento de motores.
4. Sistema de aquisição dos dados por meio do VNA e visualização dos resultados obtidos na interface gráfica.
5. Por fim, será elaborado e apresentado o relatório final deste projeto.

3 MOTIVAÇÃO

Diante da complexidade do processo de medição, elevado custo e considerável tempo para aquisição de dados referentes ao completo diagrama de irradiação de uma antena ou dispositivo sob teste (DST) é proposto a automatização e inclusão de novas funções ao sistema de medição disponível no LASen (Laboratório de Antenas e Sensoriamento) da UFCG.

Um sistema de medição como o descrito anteriormente possui um valor que pode atingir algumas dezenas de milhares de reais e, como mencionado, apenas centros altamente especializados dispõem deste sistema. As atividades de pesquisa, ensino e extensão desenvolvidas no LASen serão as principais beneficiadas com a conclusão deste projeto. Dessa forma, além de revitalizar o sistema existente, este projeto irá contribuir diretamente no orçamento das instalações do sistema de medição do LASen proporcionando a redução dos custos. Este valor será estimado no final do projeto.

Nesse contexto, este trabalho tem como objetivo diminuir o tempo de medição, aumentar a precisão das medições, automatizar o processo e introduzir novas funcionalidades ao sistema, que será utilizado nas atividades de pesquisa, ensino e extensão do LASen para caracterização de antenas comercialmente.

4 DESENVOLVIMENTO

No decorrer deste Capítulo será dado enfoque a uma abordagem das tecnologias utilizadas para o desenvolvimento do projeto.

4.1 GPIB

Nos anos 1960 a Hewlett-Packard (HP) fabricava vários instrumentos de testes e medidas automatizados, como multímetros digitais e analisadores lógicos. Eles desenvolveram o HP Interface Bus (HP-IB) para permitir interconexões mais fáceis entre controladores e instrumentos.

O barramento era relativamente fácil de implementar usando a tecnologia da época, usando um simples barramento paralelo e várias linhas individuais de controle. Eram simples periféricos HP-IB implementados em uma lógica TTL, sem usar microprocessadores.

A HP licenciou as patentes do HP-IB por uma taxa nominal para outros fabricantes. Ficou conhecido como General Purpose Interface Bus (GPIB), e se tornou um padrão para controle de instrumentos automatizados e industriais. Enquanto GPIB se tornou popular, foi formalizado por várias organizações de padronização.

Atualmente, o USB, PCI Express e Ethernet/LAN ganharam atenção como atrativas opções de comunicação para controle de instrumento. Alguns fabricantes de equipamentos para teste e medição assim como alguns especialistas têm dito que um destes barramentos, por si só, representa uma solução para todos os requisitos de instrumentação. Na realidade, é mais provável que duas ou mais tecnologias de barramento continuem a coexistir nos futuros sistemas de teste e medição porque cada barramento possui suas próprias vantagens. O desafio atual para engenheiros de teste não é escolher um único barramento ou plataforma para padronizar todas as aplicações,

Mas escolher um barramento ou plataforma apropriada para uma aplicação específica ou mesmo uma parte específica da aplicação.

FIGURA 1: CONECTOR USB-GPIB.

4.1.1 LARGURA DE BANDA

Na avaliação dos parâmetros técnicos dos barramentos, a largura de banda e a latência são duas das mais importantes características. A largura de banda mede a taxa que os dados são enviados através do barramento, tipicamente em MB/s (106 bytes por

segundo). Um barramento com grande largura de banda pode transmitir mais dados em um determinado período do que um barramento com baixa largura de banda. A maior parte dos usuários reconhece a importância da largura de banda porque esta afeta diretamente se os dados podem ser enviados através do barramento para um processador compartilhado tão rápido quanto os dados são adquiridos ou gerados, e o quanto de memória interna os instrumentos irão necessitar. A largura de banda é importante em aplicações como uma aquisição ou geração de uma forma de onda complexa como RF e aplicações de comunicação. Alta taxa de transferência de dados é particularmente importante para arquiteturas de instrumentação virtual e sintética. A funcionalidade e personalidade da instrumentação virtual e sintética são definidas por software; na maioria dos casos, isto significa que os dados devem ser transferidos para um PC host para processamento e análise.

4.1.2 LATÊNCIA

A latência mede o atraso na transmissão dos dados através do barramento. Por analogia, se compararmos um barramento para instrumentação como uma rodovia, a largura de banda corresponderia ao número de pistas e a velocidade permitida, enquanto a latência corresponderia ao atraso introduzido com os cruzamentos existentes na rodovia. Um barramento com baixa (boa) latência introduziria menos atraso entre o tempo que os dados são enviados em uma "ponta" e processados na outra "ponta" do barramento. A latência, ainda que menos notável que a banda, possui um impacto direto em aplicações onde uma sucessão rápida de pequenos comandos é enviada através do barramento.

4.1.3 COMUNICAÇÃO BASEADA EM MENSAGEM VERSUS COMUNICAÇÃO BASEADA EM REGISTRO

Barramentos que utilizam comunicação baseada em mensagem são geralmente mais lentos porque este modo de comunicação adicionar um atraso na forma de interpretação do comando e interpretação dos dados. Com uma comunicação baseada em registros, a transferência dos dados ocorre com a escrita e leitura direta dos dados binários nos registradores do hardware no dispositivo, resultando em uma transferência mais rápida. Protocolos com comunicação baseada em registro são mais comuns em

barramentos internos de PCs, onde as interconexões são fisicamente menores e são necessárias maiores taxas de transferências. Protocolos de comunicação baseados em mensagem são úteis para transmissão de dados através de longas distâncias onde um maior atraso é aceitável.

4.1.4 CONFIGURAÇÃO DO INSTRUMENTO E DESEMPENHO DO SOFTWARE

A facilidade de uso em termos de configuração do instrumento e desempenho do software é o critério mais subjetivo examinado neste artigo. Mesmo assim é importante ser discutido. A configuração do instrumento descreve um experimento "out of the box" (inicialização do dispositivo a partir da retirada de sua embalagem) e tempo de configuração. O desempenho do software relata o quão facilmente os usuários podem localizar utilitários interativos ou APIs (Interfaces para programação da aplicação) padrão como VISA para comunicar ou controlar um instrumento.

4.1.5 ROBUSTEZ DO CONECTOR

O conector físico dos barramentos implica na maneira que este é apropriado para aplicações industriais e se existe a necessidade de esforços adicionais para "reforçar" a conexão entre o instrumento e o sistema controlador. A Figura 2 apresenta fotos de diversos conectores de barramentos para instrumentação.

4.1.6 UTILIZAÇÃO GPIB

O GPIB (Barramento com Interface para Propósitos Gerais) também conhecido como IEEE 488. O GPIB é um barramento consolidado e projetado especificamente para aplicações de controle de instrumentos. O GPIB tem sido um barramento robusto e confiável por 30 anos e ainda é a escolha mais popular para controle de instrumentos devido a sua baixa latência e largura de banda aceitável. Atualmente é o barramento de maior adoção na indústria, com uma base de mais de 10.000 modelos de instrumentos com conectividade GPIB.

Com uma largura de banda máxima de 1.8 MB/s, é melhor utilizado para comunicação e controle de instrumentos autônomos. O mais recente, revisão de alta

velocidade, HS488, teve um aumento na largura de banda para 8 MB/s. A transferência é baseada em mensagem, frequentemente na forma de caracteres ASCII. Múltiplos instrumentos GPIB podem ser conectados ao mesmo cabo com uma distância total de 20 m e a largura de banda é compartilhada entre todos os instrumentos conectados ao barramento. Desconsiderando a relativamente pequena largura de banda, a latência do GPIB é significantemente menor (melhor) do que o USB e especialmente o Ethernet. Os instrumentos GPIB não são auto-detectados ou auto-configurados quando conectados ao sistema, mas o software GPIB está entre os mais bem avaliados, a robustez dos cabos e conectores são apropriados para a maior parte dos ambientes físicos. O GPIB é ideal para automatizar equipamentos existentes ou sistemas que requerem instrumentos altamente especializados além de sua interoperalidade com o Matlab uma dos principais motivos para utilização deste padrão neste projeto.

Desconsiderando a conveniência conceitual de designar um único barramento ou padrão de comunicação como a tecnologia "final" e "ideal". A história mostra que diversos padrões continuarão a coexistir como alternativa, pois cada tecnologia de barramento possui suas vantagens e desvantagens.

FIGURA 2: TABELA COMPARATIVA ENTRE DIFERENTES PADRÕES DE CONEXÃO COM EQUIPAMENTOS

	Largura de Banda (MB/s)	Latência (µs)	Distância (m) (sem extensores)	Instalação e Configuração	Robustez do Conector
GPIB	1.8 (488.1) 8 (HS488)	30	20	Bom	Melhor
USB	60 (Hi-Speed)	1000 (USB) 125 (Hi-Speed)	5	Melhor	Bom
PCI	132	0.7	Barramento Interno no PC	Ótimo	Melhor Ótimo (para PXI)
PCI Express	250 (x1) 4000 (x16)	0.7 (x1) 0.7 (x4)	Barramento Interno no PC	Melhor	Melhor Ótimo (para PXI)
Ethernet/LAN/LXI	12.5 (Fast) 125 (Gigabit)	1000 (Fast) 1000 (Gigabit)	100 m	Bom	Bom

4.2 SCPI

4.2.1 HISTÓRIA SCPI

SCPI (Comandos padrões para instrumentos programáveis) é uma linguagem de programação padrão projetada especificamente para controlar instrumentos. Ela define como você se comunica com estes instrumentos de um computador externo.

Hewlett-Packard desenvolveu o HP-IB como padrão interno para a transmissão de bytes individuais de dados e comandos entre instrumentos e computadores, definindo handshaking, addressing e protocolo geral. HP-IB ganhou aceitação generalizada indústria e migrou para GPIB (finalidade geral Interface Bus), que é construído sobre o IEEE 488.1 e 488,2 padrões. Emulação de GPIB também é implementada em interfaces Ethernet e USB. SCPI é o padrão mais comum usado para instrumentos de controle sobre as interfaces.

4.2.2 DESCRIÇÃO SCPI

SCPI é definindo como um conjunto padrão de comandos de programação. Para uma função de medição determinado (como frequência), o SCPI define os comandos específicos usados para acessar essa função através das interfaces USB, GPIB ou LAN. Se dois analisadores de ambos em conformidade com o padrão SCPI, você usaria o mesmo comando para ajustar a frequência central de cada analisador. Comandos padrão fornecem duas vantagens:

- Se você sabe como controlar funções em um instrumento SCPI, você sabe como controlar as mesmas funções em qualquer instrumento SCPI
- Programas escritos para um determinado instrumento SCPI adaptam-se facilmente para trabalhar com um instrumento SCPI semelhante.

4.3 ARDUÍNO

O Arduíno é uma plataforma de prototipagem eletrônica *open-source* que se baseia em hardware e software flexíveis e fáceis de usar. É destinado a artistas, designers, *hobbistas* e qualquer pessoa interessada em criar objetos ou ambientes interativos.

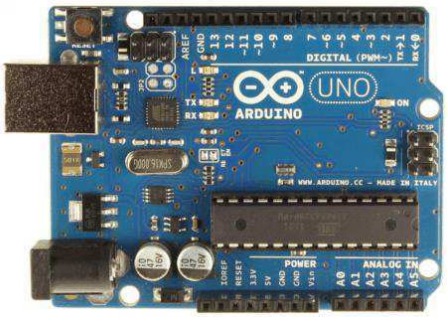
O Arduíno pode sentir o estado do ambiente que o cerca por meio da recepção de sinais de sensores e pode interagir com os seus arredores, controlando luzes, motores e outros atuadores. O microcontrolador na placa é programado com a linguagem de programação Arduíno, baseada na linguagem Wiring, e o ambiente de desenvolvimento Arduíno, baseado no ambiente Processing. Os projetos desenvolvidos com o Arduíno podem ser autônomos ou podem comunicar-se com um computador para a realização da tarefa, com uso de software específico (ex: Flash, Processing, MaxMSP).

As placas podem ser construídas de forma caseira (manualmente) ou adquiridas já montadas e o software pode ser baixado gratuitamente. O projeto do hardware (arquivos de CAD) está disponível sob licença *open-source* e você é livre para adaptá-lo para as suas necessidades.

4.3.1 HARDWARE

Existem diversas placas oficiais de Arduíno e muitas outras não oficiais. Vamos abordar a placa Arduíno Uno o qual foi utilizada neste projeto. A seguir é exibida a placa Arduíno Uno REV3:

FIGURA 3: PLACA ARDUÍNO UNO



Conforme visto na imagem acima a placa Arduíno UNO possui diversos conectores que servem para interface com o mundo externo. Vejamos como estão organizados os pinos na placa:

- 14 pinos de entra e saída digital (pinos 0-13):

    - Esses pinos podem ser utilizados como entradas ou saídas digitais de acordo com a necessidade do projeto e conforme foi definido no sketch criado na IDE.
- 6 pinos de entradas analógicas (pinos A0 - A5):
    - Esses pinos são dedicados a receber valores analógicos, por exemplo, a tensão de um sensor. O valor a ser lido deve estar na faixa de 0 a 5 V onde serão convertidos para valores entre 0 e 1023.
- 6 pinos de saídas analógicas (pinos 3, 5, 6, 9, 10 e 11):
    - São pinos digitais que podem ser programados para ser utilizados como saídas analógicas, utilizando modulação PWM.

A alimentação da placa pode ser feita a partir da porta USB do computador ou através de um adaptador AC.

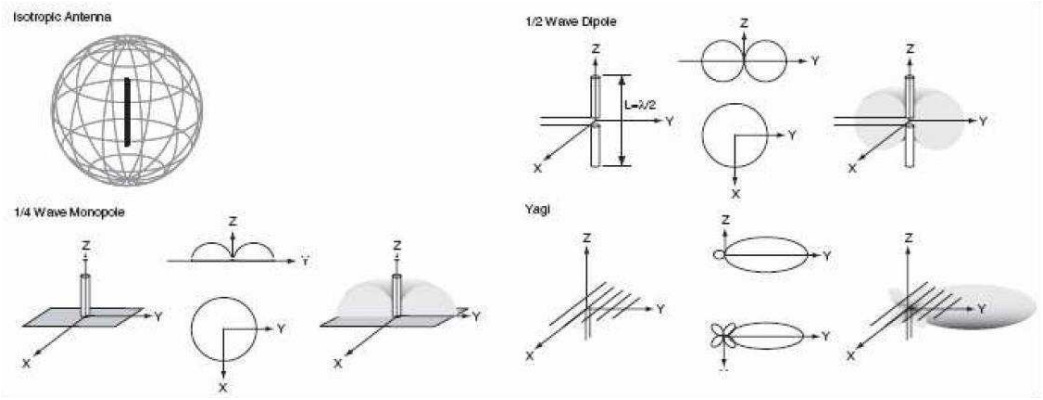
4.4 FUNDAMENTOS DE ANTENAS

De forma simplista uma antena é o componente elétrico responsável pela transmissão das ondas eletromagnéticas pelo espaço e também pela recepção destas ondas eletromagnéticas. A antena é essencialmente um transdutor que converte corrente alternada em ondas eletromagnéticas.

4.4.1 PADRÃO DE IRRADIAÇÃO

O padrão de irradiação mostra na forma de um desenho como a energia é distribuída no espaço. O termo antena isotópica é usado para definir uma antena com um padrão de irradiação perfeito em todas as direções, esta é uma antena teórica e servirá de referência para os outros tipos de antenas. Na prática cada tipo de antena possui uma característica de irradiação. A figura abaixo mostra a característica de irradiação das principais antenas. Existem antenas que irradiam igualmente em todas as direções, são chamadas omnidirecionais e antenas que irradiam em uma direção preferencial, chamadas de antenas diretivas.

FIGURA 4: DIAGRAMA DE IRRADIAÇÃO PARA DIVERSOS TIPOS DE ANTENAS.

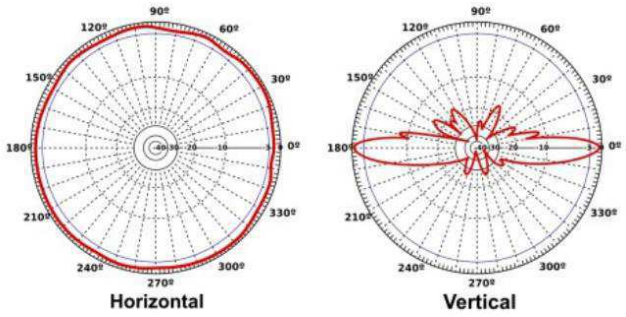


4.4.2 GANHO

O ganho de uma antena é a potência efetivamente irradiada em comparação com uma antena de referência. Normalmente a antena de referência é uma antena isotópica, neste caso a unidade é dBi que significa: Ganho em dB sobre uma antena isotópica.

Outra forma é comparar a antena a uma antena padrão do tipo dipolo, a variação aceitável entre uma antena isotópica e uma antena dipolo é de 2,2 dBi.

FIGURA 5: EXEMPLO DO GANHO DE UMA ANTENA.

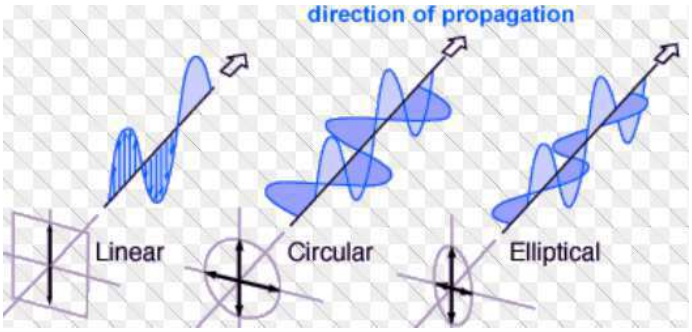


4.4.3 POLARIZAÇÃO DA ANTENA

A polarização é uma característica importante para uma antena e diz respeito à orientação dos fluxos dos campos elétrico gerado pela antena. Uma antena com campo elétrico orientado na posição vertical é dita polarizada verticalmente, da mesma forma, uma antena cujo campo elétrico esta orientado horizontalmente à antena é dita com polarização horizontal.

A recepção ocorre de maneira satisfatória quando a antena receptora e a antena transmissora possuem a mesma polarização, em outro caso parte da energia transmitida é perdida.

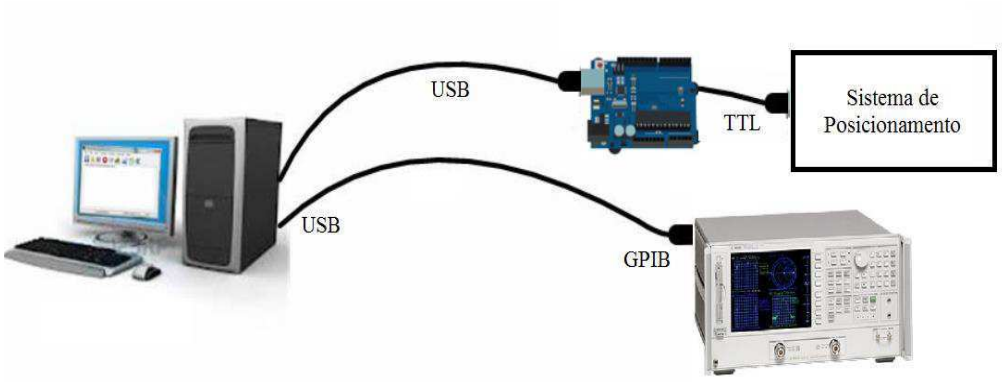
FIGURA 6: TIPOS DE POLARIZAÇÕES EM ANTENAS.



5 RESULTADOS E DISCUSSÕES

Todo o sistema foi pensado visando a automatização do processo de medições do padrão de radiação de antenas, para isto foi idealizado um sistema com as seguintes características, resumidas na Figura 7 abaixo.

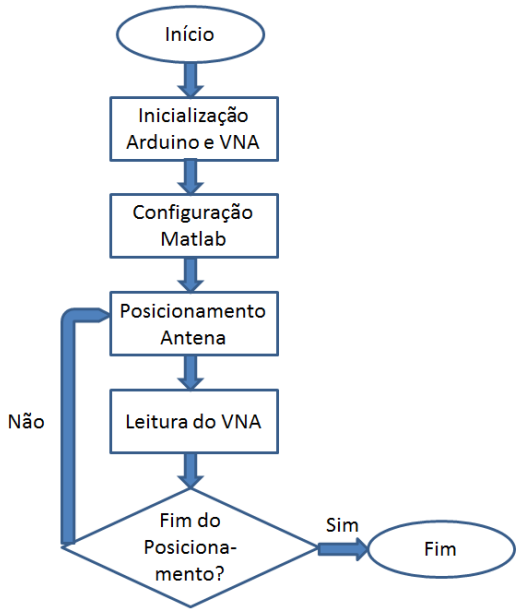
FIGURA 7: INTERCONEXÃO ENTRE OS DISPOSITIVOS DO SISTEMA DE MEDIÇÃO.



O sistema de posicionamento localiza a antena em posições pré-definidas pelo usuário em uma interface gráfica realizada no Matlab. No computador que é responsável pelo gerenciamento entre as diferentes interfaces do sistema foi instalado o Matlab com licença estudantil o qual dispõe de bibliotecas para a comunicação com o Arduíno e com o VNA. O Matlab é responsável pelo intefaceamento entre VNA e

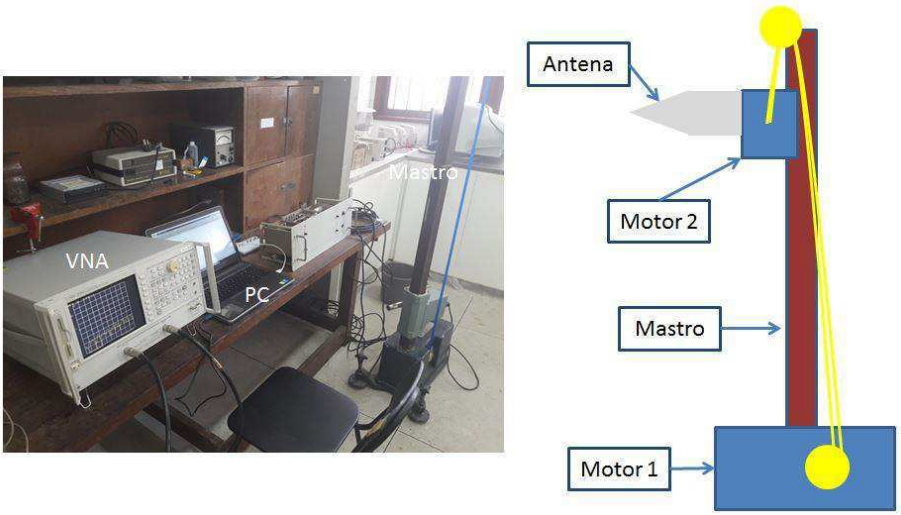
Arduíno, ou seja, a ordem de movimentação e leitura feita pelo VNA, à Figura 8 abaixo ilustra tal processo.

FIGURA 8: ROTINA DE EXECUÇÃO DO SISTEMA DE MEDIÇÃO.



A inicialização do sistema se dá pela conexão do Arduíno com o computador e a conexão do VNA com o PC por meio do cabo GPIB mostrado na Figura 9**.**

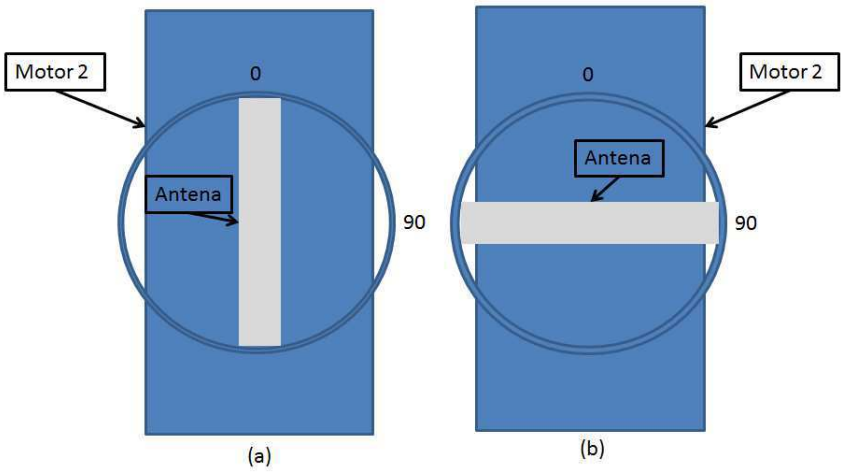
FIGURA 9: CONEXÃO DO VNA E ARDUÍNO AO COMPUTADOR.



A etapa de configuração do Matlab diz respeito aos limites de movimentação da antena, por exemplo, limites de inicio e fim do movimento vertical proporcionado pelo

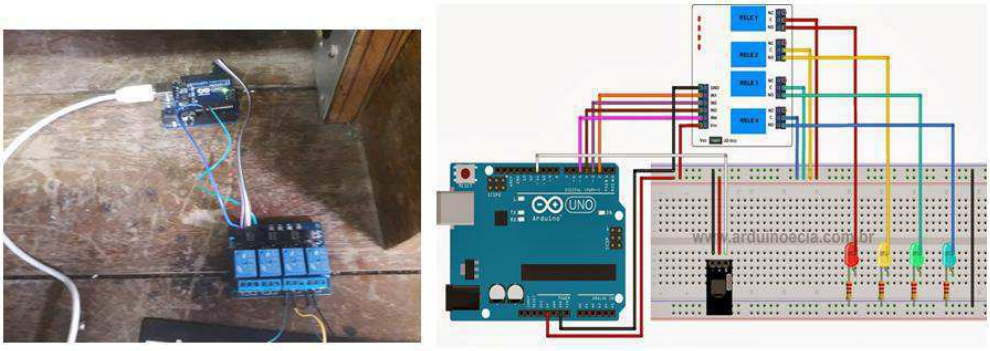
motor 1 e ângulo de rotação da antena em torno de seu eixo proporcionado pelo motor 2, a ilustração do sistema de movimento da antena é resumido na Figura 10.

FIGURA 10: ROTAÇÃO DA ANTENA EM RELAÇÃO AO SEU EIXO, (A) EIXO VERTICAL, (B) EIXO HORIZONTAL.



O movimento da antena é comandado pelo o Arduíno e um modulo relé de quatro canais vistos na Figura 11. Cada relé desse módulo suporta cargas de até 10 A, em 125 VAC, 250 VAC ou 30 VDC. Os *leds* indicadores mostram o estado do relé (ligado/desligado) em cada canal. O módulo já contém todo o circuito de proteção para evitar danos ao micro-controlador, para ativar cada relé basta impor nível lógico alto a qualquer porta do Arduíno em que o módulo relé esteja conectado, como mostrado na figura abaixo onde cada *led* representa sentidos de rotação diferentes de cada motor.

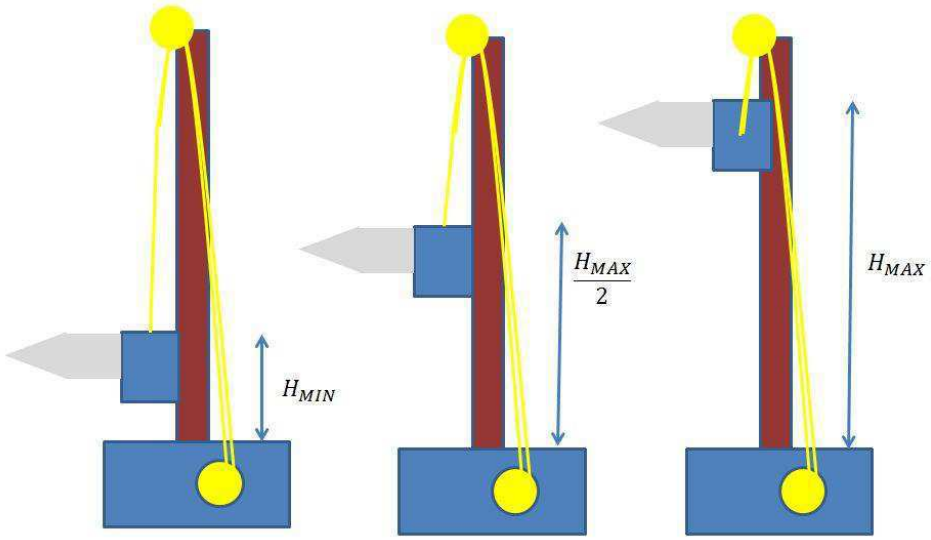
FIGURA 11: CONEXÃO DO MODULO RELÉ COM ARDUÍNO.



A antena é utilizada montada sobre o motor 2 que pode rotacionar a antena em torno de seu eixo para qualquer ângulo desejado, na Figura 10 é mostrado o processo.

O motor 1 realizar a tarefa de deslocar o conjunto antena motor por diferentes posições relacionado a vertical como é mostrado na Figura 12.

FIGURA 12: POSICIONAMENTO VERTICAL PROPORCIONADO PELO MOTOR 1.



Em seguida é realizada a leitura pelo VNA e os parâmetros são armazenados no computador o qual esta se realizando a tarefa, o Matlab indique que o posicionamento chegou ao final o programa é encerrado.

O motor responsável pelo movimento da antena no mastro possuía um problema relativo ao sistema de partida, foi necessária a troca do capacitor de partida do conjunto motor.

5.1 CONEXÃO, COMPUTADOR – VNA

Todas conexão feita neste projeto visa a integração feita com o Matlab, ou seja, o Matlab será o responsável por interconectar logicamente as partes constituintes destes sistema. Vale comentar que o equipamento em questão não estava sendo utilizado para realização de testes uma vez que era árdua a tarefa de recuperar os dados registrados no VNA, pelo motivo da captação dos dados serem possíveis apenas com a utilização de um disquete. Para isso, foi desenvolvida uma rotina no Matlab por meio da linguagem SCPI utilizando o padrão de comunicação GPIB.

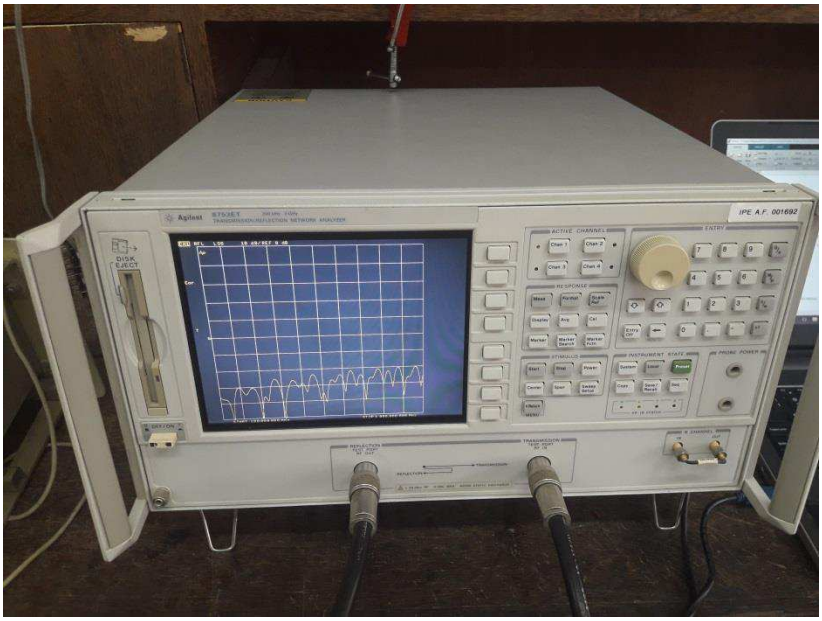
A rotina Matlab pode ser vista no anexo, que trata inicialmente a da conexão do computador com o VNA, previamente foram instalados os drivers necessários para correta execução e comunicação dos dispositivos.

O VNA 8753/ET disponível no LASen possui duas portas e é capaz de varrer uma frequência de 100 kHz à 3GHz, o dispositivo pode ser visto na Figura 13.

FIGURA 13: VNA 8753/ET DISPONÍVEL NO LABSEN.

Posteriormente, diversos parâmetros foram captados por meio a rotina Matlab e por diante os dados foram salvos no computador por meio do Matlab. Inicialmente um cabo foi conectado as portas do VNA a fim de verificar a eficiência do cabo como pode ser visto na Figura 14**,** uma medição de parâmetros primordiais foi realizada como pode ser visto nas Figura 15**,** Figura 16**,** Figura 17**.**

FIGURA 14: VNA COM DST.



Vale salientar que toda configuração feita por meio de comando por meio da interface do VNA é possível via Matlab.

FIGURA 15: COEFICIENTE DE TRASMISSÃO E REFLEXÃO ADQUIRIDOS POR MEIO DA INTERFACE GPIB.

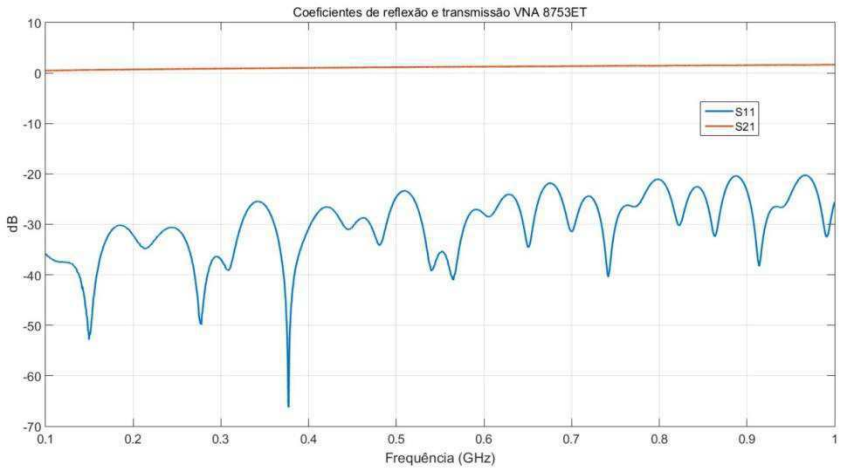


FIGURA 16: FASE DOS COEFICIENTES S11 E S21.

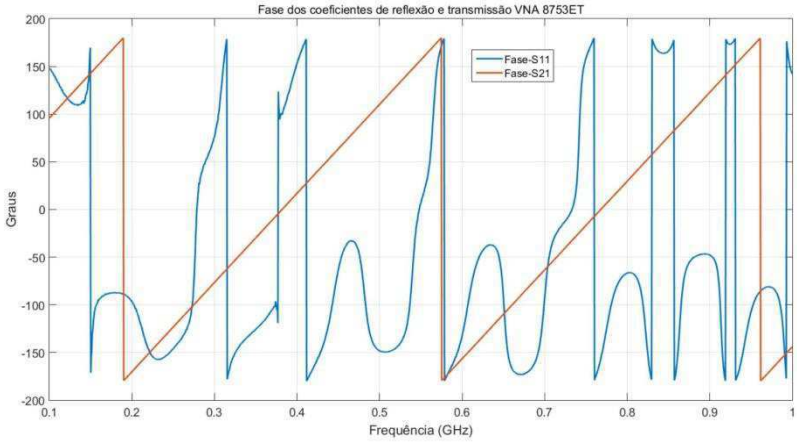
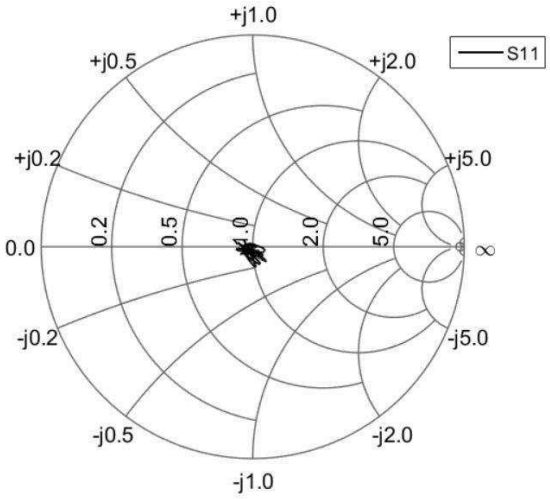


FIGURA 17: CARTA DE SMITH DO COEFICIENTE S11.



5.2 CONEXÃO COMPUTADOR – ARDUÍNO

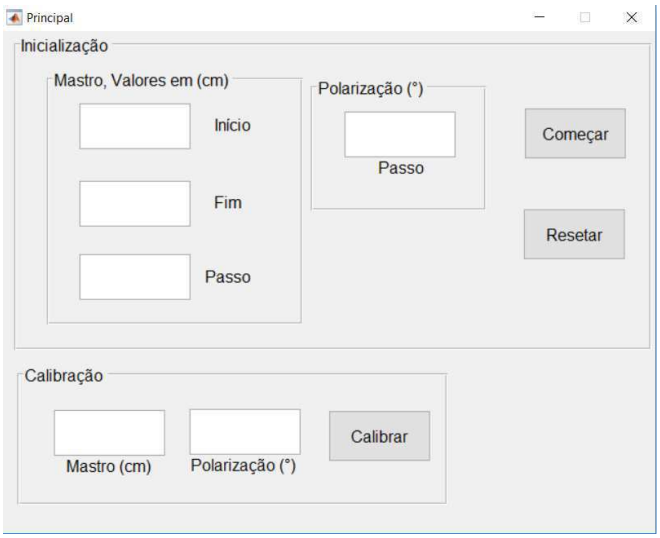
Uma vez estabelecida com sucesso a comunicação entre computador – VNA, foi iniciado a etapa de controle dos motores através do Arduíno, Para esta tarefa foi vistas as características do circuito de comando e força, ou seja, o comando como é visto na Figura 18, possui uma correte muito inferior a correte utilizada para a partida do motor.

FIGURA 18: CIRCUITO DE COMANDO PARA ACIONAMENTO DOS MOTORES.

No motor 1 para o circuito de comando temos cerca de 70mA no momento do acionamento da chave de partida, já no motor 2 temos em torno de 50mA no momento em que acionamos a chave para acionamento. Tendo em vista estas características buscamos retirar a dependência da chave manual para partida dos motores por meio um modulo relé o qual é conectado ao Arduíno, este modulo vem a ditar o movimento dos motores.

Para o acionamento automático e melhor configuração dos parâmetros de rotação e deslocamento da antena, foi implementado uma interface gráfica por meio do Matlab, que pode ser vista na Figura 19 e seu código será disponibilizado em anexo.

FIGURA 19: INTERFACE PARA CONFIGURAÇÃO DOS DESLOCAMENTOS DA ANTENA



A interface foi dividida em duas seções a parte de calibração e a parte de inicialização. A calibração que tem por objetivo a estabelecer a posição inicial do motor e o ângulo inicial da antena.

Já na parte referente a inicialização temos os parâmetros inicio que se trata de onde desejamos começar a o movimento da antena à partir da calibração, o fim que nos diz o ponto de parada. Todos os campos da inicialização devem estar preenchidos para inicialização do programa.

O campo passo nos diz exatamente o deslocamento por medição. Foi verificado que 1 segundo com a relé pressionado corresponde a 10 cm de deslocamento, com esta relação foi obtido o deslocamento exato à partir de cada parâmetro posto nos campos da interface. Por exemplo, supondo que não rotacionaremos a antena, Polarização-Passo = 0, após o sistema ser calibrado, suponhamos que desejamos começar do ponto inicial ao da calibração, ou seja, INICIO = 0, e desejamos para 90 cm depois, temos FIM = 90, levando em consideração um passo de 10 cm, ou seja, PASSO = 10, desta forma teremos uma medição na vertical a cada 1 segundo.

Caso desejemos rotacionar a antena colocaremos diretamente o valor do ângulo em graus, inicialmente foi verificado que pressionando o botão por 1 segundo corresponde a uma rotação de 90º, desta forma poderemos realizar medições na horizontal e vertical.

5.3 INTERCONEXÃO DO SISTEMA

Após a realização de ambas as etapas, foi realizada a junção dos sistemas por meio da integração lógica com o Matlab. A parte física da montagem final pode ser visto na Figura 20.

FIGURA 20: CONFIGURAÇÃO FINAL DO SISTEMA DE MEDIÇÃO AUTOMÁTICO.

Esta parte foi prejudicada pela dificuldade de interconexão do Arduíno com o sistema de comando dos motores visto que o modulo relé após algumas comutações era desconfigurado e seus contatos permaneciam todos abertos ou todos fechados criando assim em algumas situações sobre correntes no circuito vindo a danificar alguns capacitores da parte de comando, uma vez que, como não possuíamos o diagrama de circuito da parte de comando não foi possível identificar rapidamente o defeito e contornar a situação. O sistema ficava funcional apenas por algumas comutações.

Para solucionar esse problema foram introduzidos circuitos de proteção para o modulo relé que ainda estão em fase de conclusão.

6 CONCLUSÃO

Neste documento foram vistos tópicos acerca da interconexão de partes de diferentes sistemas por meio de tecnologias altamente utilizadas no meio industrial como, por exemplo, o GPIB e Matlab.

Foi recuperado dois sistemas que estava em desuso no âmbito do LabSen, o sistema de posicionamento de antenas e o leitura por meio do analisador vetorial de redes 8573/ET. Possibilitando uma aplicação barata em comparação a sistemas disponibilizados no mercado com estas características, além de inovador pois normalmente os sistemas possuem o posicionamento na vertical como nosso sistema e posicionamento em torno do mastro, ou seja, leitura do ângulo levando em consideração rotação em torno do mastro. Para nosso sistema ser completo bastaria realizar rotações em torno do mastro, assim teríamos todas possibilidades possíveis de leitura de diagrama de radiações de forma automática. Como sugestão, resta realizar um levantamento no circuito da parte de comando do acionamento dos motores a fim de miniaturização do circuito sabendo que este parte constituinte do sistema possui cerca de 40 anos, assim a comutação dos motores poderia ser feita de maneira eficiente.

Os custos adicionais para realização deste projeto foram a compra do Arduíno (R$ 40,00) e o cabo GPIB (R$ 2.000,00), visando mais uma vez a diminuição no preço total do sistema. O custo total do projeto incluindo os motores e o mastro elevará o preço do sistema para cerca de R$ 4.000,00

REFERÊNCIAS

[1] BALANIS, Constantine A. **Antenna theory: analysis and design**. John Wiley & Sons, 2016.

[2] FORNETTI, Francesco. **Instrumentation control, data acquisition and processing with MATLAB**. Explore RF, 2013.

[3] MathWorks, Legacy **MATLAB and Simulink Support for Arduino**. Disponível em: http://www.mathworks.com/matlabcentral/fileexchange/32374-legacy-matlab-and-simulink-support-for-arduino. Acesso em: 10 jun. De 2016.

[4] TAYGUR, Mehmet Mert et al. Low-Cost FPGA Based Antenna Pattern Measurement System. In: **Applied Electronics (AE), 2013 International Conference on**. IEEE, 2013. p. 1-4.

[5] http://www.diamondeng.net/antenna-measurement-27/pricing/37-antenna-measurement/x000, visitado em 13/02/2017.

[6] ANDERSON, J. T. et al. A COST-EFFECTIVE ANTENNA POSITIONING SYSTEM FOR MODERN RADIO-FREQUENCY (RF) AND MICROWAVE ANTENNA MEASUREMENTS. In: **Proceeding of the ASEE-2010 North Midwest sectional conference**. 2010

[7] JONES, Eric D. et al. Catching the Right Bus V: A Beginners' Guide to Programming the IEEE-488 Bus. **Computers in Physics**, v. 9, n. 2, p. 140-147, 1995.

APÊNDICE A

Código para conexão do Computador com o VNA.

```
close all
clear
clc

% % % % % % % % % % rver pagina 426 Span, preset - pag 221, pag 227

    set(0, 'DefaultAxesFontWeight', 'normal', ...
     'DefaultAxesFontSize', 14, ...
     'DefaultAxesFontAngle', 'normal', ... % Not sure the difference here
     'DefaultAxesFontWeight', 'normal', ... % Not sure the difference here
     'DefaultAxesTitleFontWeight', 'normal', ...
     'DefaultAxesTitleFontSizeMultiplier', 1) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Find a VISA-GPIB object.
obj1 = instrfind('Type', 'visa-usb', 'RsrcName', 'GPIB0::16::INSTR', 'Tag', '');

% Create the VISA-GPIB object if it does not exist
% otherwise use the object that was found.
if isempty(obj1)
   obj1 = visa('NI', 'GPIB0::16::INSTR');
else
   fclose(obj1);
   obj1 = obj1(1);
end

set(obj1,'InputBufferSize', 80050);
pause(0.5); %% let the time to allocate the buffer memory

% Connect to instrument object, obj1.
fopen(obj1);

% Communicating with instrument object, obj1.

% fwrite(obj1,'OPC?;PRES');
fwrite(obj1, 'FORM4');
fwrite(obj1, 'CHAN1;LOGM;S11');
fwrite(obj1, 'CHAN1;10dB;');
fwrite(obj1, 'CHAN1;REFV;0dB;');
% fwrite(obj1, 'S11');
fwrite(obj1, 'AUXCON');
fwrite(obj1, 'CHAN3;SMIC');
```

```
fwrite(obj1, 'CHAN2;LOGM;S21');
fwrite(obj1, 'CHAN2;10dB;');
fwrite(obj1, 'CHAN2;REFV;0dB;');
% fwrite(obj1, 'S21');
fwrite(obj1, 'AUXCON');
fwrite(obj1, 'CHAN4;SMIC');

F_Start = query(obj1,'STAR?');
F_Stop = query(obj1,'STOP?');
points = query(obj1, 'POIN?');

data3 = query(obj1,'CHAN4;OUTPFORM');
data2 = query(obj1,'CHAN3;OUTPFORM');
data1 = query(obj1,'CHAN2;OUTPFORM');
data = query(obj1,'CHAN1;OUTPFORM');

fwrite(obj1, 'CHAN4;PHAS');
data5 = query(obj1,'CHAN4;OUTPFORM');

fwrite(obj1, 'CHAN3;PHAS');
data4 = query(obj1,'CHAN3;OUTPFORM');

%Conversao Char p/ int;

points = str2num(points);
F_Start = str2num(F_Start);
F_Stop = str2num(F_Stop);
logMChan1 = str2num(data);
logMChan2 = str2num(data1);
data2 = str2num(data2);
data3 = str2num(data3);
phasChan1 = str2num(data4);
phasChan2 = str2num(data5);
SmithChan1 = data2(:,1)+j*data2(:,2);
SmithChan2 = data3(:,1)+j*data3(:,2);

fwrite(obj1,'AUXCOFF;');
fwrite(obj1,'DUACOFF;');

figure;
[ss1,sss1] = smithchart(SmithChan1);
ss1.Color = 'black';
ss1.LineWidth = 1.5;
sss1.LabelSize = 15;
sss1.LineWidth = 1;
legend('S11');
set(gcf,'color','w');

figure;
[ss2,sss2] = smithchart(SmithChan2);
```

```
ss2.Color = 'red';
ss2.LineWidth = 1.5;
sss2.LabelSize = 15;
sss2.LineWidth = 1;
legend('S21');
set(gcf,'color','w');

Freq = linspace(F_Start,F_Stop,points)/1e9;

figure;
plot(Freq,phasChan1(:,1),Freq,phasChan2(:,1),'linewidth',2);
grid on;
set(gcf,'color','w');
title('Fase dos coeficientes de reflexão e transmissão VNA 8753ET');
legend('Fase-S11','Fase-S21');
xlabel('Frequência (GHz)');
ylabel('Graus');


%Plotagem dos valores obtidos
figure;
plot(Freq,logMChan1(:,1),Freq,logMChan2(:,1),'linewidth',2);
grid on;
set(gcf,'color','w');
title('Coeficientes de reflexão e transmissão VNA 8753ET');
legend('S11','S21');
xlabel('Frequência (GHz)');
ylabel('dB');

%
% % Connect to instrument object, obj1.
% fopen(obj1);

% Disconnect from instrument object, obj1.
fclose(obj1);

% Delete instrument object, obj1.
% delete(obj1);
```

Código para conexão do Computador com o Arduíno.

```
function varargout = Gui_Mastro(varargin)
% GUI_MASTRO MATLAB code for Gui_Mastro.fig
%      GUI_MASTRO, by itself, creates a new GUI_MASTRO or raises the existing
%      singleton*.
%
%      H = GUI_MASTRO returns the handle to a new GUI_MASTRO or the handle to
%      the existing singleton*.
```

```
%
%      GUI_MASTRO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_MASTRO.M with the given input
arguments.
%
%      GUI_MASTRO('Property','Value',...) creates a new GUI_MASTRO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Gui_Mastro_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Gui_Mastro_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Gui_Mastro

% Last Modified by GUIDE v2.5 27-Jan-2017 17:32:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Gui_Mastro_OpeningFcn, ...
                   'gui_OutputFcn',  @Gui_Mastro_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Gui_Mastro is made visible.
function Gui_Mastro_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Gui_Mastro (see VARARGIN)

% Choose default command line output for Gui_Mastro
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

delete(instrfindall);
clear;
global ard cte cte2;
cte = 1;
cte2 = 1;
ard = arduino();
% configurePin(ard,'A0','AnalogInput');


% Configuration of digital pins
configurePin(ard,'D2','DigitalOutput');
configurePin(ard,'D3','DigitalOutput');
configurePin(ard,'D4','DigitalOutput');
configurePin(ard,'D5','DigitalOutput');

% Set initial pins
writeDigitalPin(ard,'D2',1);
writeDigitalPin(ard,'D3',1);
writeDigitalPin(ard,'D4',1);
writeDigitalPin(ard,'D5',1);



% UIWAIT makes Gui_Mastro wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Gui_Mastro_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in calib.
function calib_Callback(hObject, eventdata, handles)
% hObject    handle to calib (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ard cte cte2;

if handles.cal_mastro > 0
   writeDigitalPin(ard,'D4',0)
```

```
   pause(cte*handles.cal_mastro)
   writeDigitalPin(ard,'D4',1)
    disp('Mastro+')
elseif isnan(handles.cal_mastro) | handles.cal_mastro == 0
    set(handles.cal_mastro_edttext,'String','');
    disp('nadaM')
else
    writeDigitalPin(ard,'D5',0)
    pause(cte*abs(handles.cal_mastro))
   writeDigitalPin(ard,'D5',1)
   disp('Mastro-')
end

pause(2)

if handles.cal_pol > 0 && handles.cal_pol < 360
   writeDigitalPin(ard,'D2',0)
   pause(cte2*handles.cal_pol)
   writeDigitalPin(ard,'D2',1)
   disp('pol+')
elseif   handles.cal_pol > 360 | isnan(handles.cal_pol) | handles.cal_pol == 0
    set(handles.cal_pol_edttext,'String','');
    disp('nadaP')
else
    writeDigitalPin(ard,'D3',0)
    pause(cte2*abs(handles.cal_pol))
   writeDigitalPin(ard,'D3',1)
   disp('pol-')
end



% --- Executes on button press in comecar.
function comecar_Callback(hObject, eventdata, handles)
% hObject    handle to comecar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global ard i u cte cte2 temp;

% % % % Saber onde comeca
% writeDigitalPin(ard,'D4',0);
% pause(abs(handles.inicio)*cte);
% writeDigitalPin(ard,'D4',1);

pause(1);
temp = handles.inicio;
for i = handles.inicio:abs(handles.passo):abs(handles.fim)
%     Mastro
   if handles.fim <  0 & ~(temp == 0)
```

```
        writeDigitalPin(ard,'D5',0);
        pause(cte*abs(handles.passo));
        writeDigitalPin(ard,'D5',1);
        disp(sprintf('Mastro passo Neg %d',i));
    elseif temp == 0
        disp('Mastro passo 0');
        temp = temp + 1;
    else
        writeDigitalPin(ard,'D4',0);
        pause(cte*abs(handles.passo));
        writeDigitalPin(ard,'D4',1);
        disp(sprintf('Mastro passo Pos %d',i));
    end
    pause(2);

    for u = 0:handles.pol_passo:180

% % % % %     Funcao VNA
% % % % %
%         Giro
    writeDigitalPin(ard,'D2',0);
    pause(cte2/30*handles.pol_passo)
    writeDigitalPin(ard,'D2',1);
    disp(sprintf('Pol passo %d',u));
    pause(2);
% % % %     Simula a leitura do VNA


% % % %     Talvez necessite de um PAUSE por causa da leitura do VNA,
% procurar um meio de verficar se a medicao foi feita corretamente WAIT ->
% VNA.
    end
end

function cal_mastro_edttext_Callback(hObject, eventdata, handles)
% hObject    handle to cal_mastro_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cal_mastro_edttext as text
%        str2double(get(hObject,'String')) returns contents of cal_mastro_edttext as a
double
handles.data5=get(hObject,'String');
handles.cal_mastro=str2double(handles.data5);
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function cal_mastro_edttext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cal_mastro_edttext (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function cal_pol_edttext_Callback(hObject, eventdata, handles)
% hObject    handle to cal_pol_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cal_pol_edttext as text
%        str2double(get(hObject,'String')) returns contents of cal_pol_edttext as a double
handles.data6=get(hObject,'String');
handles.cal_pol=str2double(handles.data6);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function cal_pol_edttext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cal_pol_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function pol_passo_edttext_Callback(hObject, eventdata, handles)
% hObject    handle to pol_passo_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pol_passo_edttext as text
%        str2double(get(hObject,'String')) returns contents of pol_passo_edttext as a
double
handles.data4=get(hObject,'String');
handles.pol_passo=str2double(handles.data4);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
```

```
function pol_passo_edttext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pol_passo_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function inicio_edttext_Callback(hObject, eventdata, handles)
% hObject    handle to inicio_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inicio_edttext as text
%        str2double(get(hObject,'String')) returns contents of inicio_edttext as a double
handles.data1=get(hObject,'String');
handles.inicio=str2double(handles.data1);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function inicio_edttext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inicio_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function fim_edttext_Callback(hObject, eventdata, handles)
% hObject    handle to fim_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fim_edttext as text
%        str2double(get(hObject,'String')) returns contents of fim_edttext as a double
handles.data2=get(hObject,'String');
handles.fim=str2double(handles.data2);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
```

```
function fim_edttext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fim_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function passo_edttext_Callback(hObject, eventdata, handles)
% hObject    handle to passo_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of passo_edttext as text
%        str2double(get(hObject,'String')) returns contents of passo_edttext as a double
handles.data3=get(hObject,'String');
handles.passo=str2double(handles.data3);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function passo_edttext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to passo_edttext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.inicio_edttext,'String','');
set(handles.passo_edttext,'String','');
set(handles.fim_edttext,'String','');
set(handles.pol_passo_edttext,'String','');
set(handles.cal_mastro_edttext,'String','');
set(handles.cal_pol_edttext,'String','');
```