UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE COMPUTAÇÃO

Julyan Figueredo Martins

MATRIZ DE INTERCONEXÃO ELETRÔNICA PROGRAMÁVEL PARA PROTOTIPAGEM RÁPIDA E EXPERIMENTAÇÃO REMOTA

Araranguá

2018

Julyan Figueredo Martins

MATRIZ DE INTERCONEXÃO ELETRÔNICA PROGRAMÁVEL PARA PROTOTIPAGEM RÁPIDA E EXPERIMENTAÇÃO REMOTA

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Engenharia de Computação.
Orientador: Prof. Tiago Oliveira Weber, Dr.

Araranguá, novembro de 2018.

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Martins, Julyan
MATRIZ DE INTERCONEXÃO ELETRÔNICA PROGRAMÁVEL PARA PROTOTIPAGEM RÁPIDA E EXPERIMENTAÇÃO REMOTA / Julyan Martins ; orientador, Prof. Dr. Tiago Oliveira Weber, 2018.
84 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2018.

Inclui referências.

1. Engenharia de Computação. 2. Eletrônica analógica, laboratórios remotos. I. Oliveira Weber, Prof. Dr. Tiago . II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Julyan Figueredo Martins

MATRIZ DE INTERCONEXÃO ELETRÔNICA PROGRAMÁVEL PARA PROTOTIPAGEM RÁPIDA E EXPERIMENTAÇÃO REMOTA

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obteno do Ttulo de "Bacharel em Engenharia de Computação", e aprovado em sua forma final pela Universidade Federal de Santa Catarina.

Araranguá, 28 de novembro 2018.

Eliane Pozzebon Assinado de forma digital por Eliane Pozzebon Dados: 2018.12.03 09:51:02 -02'00'

Prof. Dr. Eliane Pozzebon
Coordenadora
Universidade Federal de Santa Catarina

Banca Examinadora:

Prof. Dr. Tiago Oliveira Weber
Universidade Federal de Santa Catarina

Prof. Dr. Marcelo Daniel Berejuck
Universidade Federal de Santa Catarina

Prof. Dr. Juarez Bento Da Silva
Universidade Federal de Santa Catarina

Dedico este trabalho à minha família.

AGRADECIMENTOS

Agradeço primeiramente ao professor Tiago, que além de ser um grande e empolgado instrutor, ajudou-me imensamente nestes últimos semestres do curso.

Agradeço aos meus colegas e amigos, tantos que conheci no decorrer do curso, e que estarão no meu coração e orações pro resto da vida. Que Deus nos guie nessa jornada que está só começando.

Enfim, agradeço àqueles que acreditam mais em mim do que eu mesmo. Aos meus pais, Sérgio João Martins e Maria de Fátima Figueredo Martins, e à minha irmã Stephani.

RESUMO

Laboratórios de eletrônica estão presentes em universidades, escolas e institutos de pesquisa. Os recursos disponíveis nestes laboratórios são, em muitos casos, escassos, ou disponíveis durante horários limitados. Para solucionar esse problema, a utilização de laboratórios remotos é uma alternativa que permite uso compartilhado de recursos. Ainda no âmbito de pesquisa, a realização de experimentos eletrônicos com interconexões programáveis pode facilitar implementações e reduzir o custo de protótipos. A proposta desse trabalho foi projetar e implementar uma matriz de interconexão eletrônica programável utilizando tecnologia MOS. Ela foi desenvolvida utilizando a plataforma *Arduino* e um circuito integrado de matriz de chaveamento analógico (MT8804). O sistema permitiu a programação da matriz e realização de medições através de um computador conectado ao *Arduino* e às placas desenvolvidas. Para testar o sistema, foram realizados três circuitos, dois circuitos resistivos divisores de tensão e o terceiro usando um amplificador operacional para criar um amplificador na configuração não inversora. Os resultados indicam o funcionamento correto do sistema e a interconexão correta dos componentes eletrônicos. O protótipo no estado atual permite a realização de circuitos com 12 nós e 24 terminais de componentes eletrônicos, com possibilidade de aumentar esses números. Concluiu-se que o protótipo obteve sucesso na implementação local. Enfim, a matriz de interconexões já pode ser incorporada em um ambiente de laboratório remoto para testes.

Palavras-chave: laboratório, remoto, eletrônica, analógica, matriz, interconexão

ABSTRACT

Analog electronic laboratiories are present at universities, schools and research intitutions. The resources available in these laboratories are, in most cases, scarse, or available in limited time periods. To solve this problem, the utilization of remote laboratories is an alternative that allows the sharing of resources. In research, the realization of eletronic experiments with programable interconections can improve the implementation and reduce prototype costs. The objective of this paper was to project and implement a programable electronic analog switching matrix using MOS technology. It was developed using the Arduino platform and the integrated circuit of an analog switching matrix (MT8804). The system allows the configuration of the matrix and measurements using a computer connected to the developed boards through the Arduino. The system was tested using three electronic circuits, two of them were voltage dividers with resistors and the third was a non inverter amplifier with an operational amplifier(opamp). The results show the correct operation of the systema and the correct switching and internconetion of components. The prototype at this stage allows electronic experiments with circuits with up to 12 nodes and 24 component terminals, with the possibility of increasing those numbers with additional boards. The system was a sucess at the local level. Now the programable electronic analog switching matrix can be included in a remote laboratory enviornment for testing.

Keywords: remote, laboratory, electronics, switching, matrix

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O pioneiro no ensino a distância foi o inventor inglês, Sir Issac Pitman(MATTHEWS, 1999). Ele teve a ideia de criar um curso por correspondência, que disponibilizasse instrução para um número ilimitado de pessoas. A partir de então, o ensino a distância por correspondência cresceu muito, com a criação de cursos e programas intensivos em vários países. Um marco no desenvolvimento deste tipo de ensino foi a criação da *British Open University* em 1969. Os materiais de estudo enviados por e-mail pela universidade incluíam vídeos, textos e áudios. Nos anos seguintes, várias outras universidades a distância foram criadas. Em 1980, nos EUA, 300.000 estudantes já participavam de cursos universitários a distância. Hoje em dia, as maiores universidades do mundo possuem disciplinas inteiras *online*, as vezes até cursos inteiros.

Segundo dados do eMEC (EMEC, 2016), de 2009 à 2014 houve um aumento de 60%, com destaque para os cursos de engenharia, que totalizaram 168 cursos, ofertando 351.034 vagas. Na engenharia de computação, 42.269 vagas foram ofertadas. No Brasil já existem 25.166 cursos superiores a distância segundo ((ABED), 2017).

Analisando o começo da história da educação por correspondência, percebe-se que há um esforço humano para romper as barreiras típicas do ensino tradicional, principalmente a falta de recursos, de vagas nas universidades e estruturas precárias. A educação a distância possibilita acabar com a exclusividade do ensino. Um grande professor, renomado por seus grandes feitos científicos, agora pode alcançar um número maior de estudantes, usando a combinação entre ensino a distância com ensino tradicional. Não é o intuito do ensino a distância substituir o ensino presencial, mas sim potencializá-lo.

Uma parte importante na educação a distância são os experimentos práticos. Muitos cursos, principalmente aqueles que envolvem tecnologia, têm a experimentação como parte integrante e essencial do processo de aprendizagem.

Nos cursos de engenharia e tecnologia, o treinamento teórico e prático são indispensáveis. Nas disciplinas de circuitos elétricos e eletrônicos, além das simulações computacionais dos circuitos, a prática nos laboratórios é um componente essencial no aprendizado. Num modelo de laboratório presencial, prática laboratorial fica limitada ao espaço e aos horários de disponibilidade da instituição de ensino. Os laboratório-

rios remotos trazem maior flexibilidade aos usuários, pois é comum que os laboratórios remotos funcionem em todos os momentos, excluindo eventuais manutenções de sistema.

Os testes com simuladores elétricos são rápidos e versáteis, possibilitando testes sem risco e com poucos recursos. No entanto, as simulações não reproduzem com facilidade todas as possíveis variações e problemas inesperados que aparecem em testes físicos. Nas práticas laboratoriais, pode-se fazer experimentos reais que mostrarão as variações no sistema que se está projetando, como: diferenças entre os valores nominais e reais dos componentes; efeitos de alta ordem em dispositivos não lineares; efeito de ruídos de fontes diversas e problemas na aquisição dos valores medidos. No entanto, criar e manter um laboratório de eletrônica experimental demanda muitos recursos, além de exigir que o usuário esteja presente no local do experimento.

Na universidade e na educação básica, a maior restrição à implementação dos laboratórios tradicionais de eletrônica é a disponibilidade de equipamentos. Muitas escolas de educação básica possuem estruturas precárias, o que torna difícil a implementação dos laboratórios, tendo em vista outras necessidades estruturais mais urgentes. Segundo dados do (INEP, 2017), nas escolas básicas, somente 10% possui laboratórios de ciência.

Em face das potencialidades e dificuldades das simulações e dos laboratórios físicos, surge o conceito dos laboratórios remotos e experimentação remota. Experimentos remotos são experimentos físicos reais, realizados a distância e sem o contato direto do usuário com os equipamentos do laboratório. Os laboratórios remotos estão disseminados em várias universidades do mundo e abrangem as mais variadas áreas científicas. Várias iniciativas já existem para democratizar o acesso de alunos e usuários em geral aos experimentos remotos(KAZMIERKOWSKI; LISERRE, 2008).

Para pesquisa na área de eletrônica, o tempo e custo de confecção de placas de circuito impresso restringe a agilidade em testar novas ideias e projetos. Criar placas de circuito impresso específicas para uma aplicação envolve fazer o projeto das conexões do circuito (esquemático), desenhar a placa com suas interconexões metálicas (leiaute), confeccionar a placa e soldar os componentes. Estes dois últimos passos podem ser feitos manualmente, mas muitas vezes são realizados através de empresas que cobram um lote mínimo de placas, incrementando o custo de prototipagem. Para circuitos puramente digitais, a utilização de dispositivos FPGAs (Field-Programmable Gate Arrays) resolve este problema, permitindo a prototipagem rápida de circuitos

através do rearranjo programável de conexões entre portas lógicas, memórias e outros componentes. Por outro lado, para circuitos analógicos ou de sinal misto (analógico e digital), os dispositivos FPAAs (Field-Programmable Analog Array) não permitem tanta flexibilidade quanto às FPGAs, sendo normalmente direcionadas para aplicações específicas.

Alguns laboratórios remotos para experimentação de circuitos eletrônicos já foram desenvolvidos, e vários deles atendem as funcionalidades essenciais que o usuário de um laboratório real esperaria, como o projeto VISIR (GARCIA-ZUBIA et al., 2011). No entanto, estes laboratórios usualmente não utilizam hardware de baixo custo.

O potencial dos laboratórios remotos para aumentar o aprendizado dos alunos é cada vez mais visível. Além de auxiliar no aprendizado, alguns laboratórios mostram uma eficácia no aprendizado tão grande ou até maior que laboratórios presenciais(GARCIA-ZUBIA et al., 2017a).

1.2 METODOLOGIA

O sistema que foi implementado começou com a ideia da prototipação rápida de circuitos eletrônicos. Fazer um sistema onde fosse possível conectar componentes eletrônicos de maneira programável e sem o manuseio dos componentes do circuito. Além disso, essa conexão programável de componentes deveria ser feita de preferência com um multiplexador matricial analógico, que já estava disponível na UFSC para testes.

Fazer o projeto de um circuito e depois ter de fazer a placa *pcb* pode ser laborioso, então a ideia inicial era dar essa alternativa para os projetistas. Essa implementação seria usada para algum tipo de sistema de prototipagem rápida, como em uma *protoboard*, mas sem que o projetista tivesse que mexer nos componentes ou trocá-los. Nesse instante, e pesquisando sobre o assunto, ficou claro que essa ideia inicial era muito similar aos laboratórios remotos de eletrônica analógica. Estudando esses laboratórios, identificou-se que esse sistema de prototipação rápida se assemelhava muito às matrizes de interconexão que existem nas variadas implementações dos laboratórios.

Nas implementações das matrizes de interconexões foi detectada uma tendência. Todas essas matrizes usavam relês eletromecânicos, diferente da proposta feita no início das atividades do presente trabalho. Isso serviu como incentivo, pois esse trabalho poderia preencher essa lacuna, ou melhor, essa falta de opções no que se refere as tecnologias

usadas para implementar matrizes de interconexões.

Depois disso, testes com circuitos muito simples foram realizados usando uma *protoboard*. Com o sucesso destes testes na interconexão dos componentes, deu-se início ao primeiro protótipo. Foram feitos dois projetos de protótipo. No primeiro já era possível fazer a configuração da matriz, usando a placa do *Arduino UNO* para mandar dados de configuração e receber as dados de leitura de tensão. Por meio do *GTKterm*, um terminal usado especificamente para comunicação serial, foi possível mandar os comandos do computador, e assim controlar a matriz de interconexões e fazer circuitos de divisor de tensão.

Porém, alguns melhoramentos eram necessários, principalmente no *layout* da placa e na posição de componentes. Por isso, um segundo protótipo foi desenvolvido. Seria necessário criar um matriz de interconexões maior, onde houvesse maior número de terminais para componentes, mais do que os 8 terminais da placa do primeiro protótipo. O *layout* deste segundo protótipo possibilita que placas sejam empilhadas, formando uma matriz cada vez maior.

Para validar essas placas criadas no segundo protótipo, foi criado um sistema de validação das placas. Usando um programa no *Arduino* e um registrador de deslocamento auxiliar, todas as possíveis conexões entre linhas e colunas foram testadas. Este teste também garantiu que as trilhas das placas desenvolvidas estavam em boas condições. Aquelas placas que não passavam nos testes foram consertadas ou substituídas.

1.3 OBJETIVOS

O objetivo do presente trabalho é mostrar o desenvolvimento de uma implementação de um laboratório remoto de eletrônica analógica, com especial atenção à matriz de interconexões, peça fundamental desse tipo de laboratório. A implementação aqui mostrada tentará trazer uma alternativa com tecnologia diferente das matrizes de interconexões já utilizadas nos laboratórios remotos de eletrônica analógica existentes.

1.4 JUSTIFICATIVA

As implementações dos laboratórios remotos de eletrônica analógica agregam um custo alto devido às escolhas de equipamentos de instrumentalização e medição eletrônicos caros. Levanta-se a hipótese de realizar os mesmos efeitos com equipamentos mais baratos. As ma-

trizes de interconexão usadas nos laboratórios observados sempre usaram relês eletromecânicos. Este trabalho levanta a hipótese quanto viabilidade de construir uma matriz de interconexões com tecnologia de chave comutadora MOS.

1.5 TRABALHOS RELACIONADOS

Outros trabalhos foram desenvolvidos pelo autor nos últimos semestres, abordando o mesmo tema:

- MARTINS, Julyan Figueredo; WEBER, Tiago Oliveira. Matriz de Interconexão Eletrônica Programável de Baixo Custo e Aplicação em Laboratório Remoto e Prototipagem Rápida. II Simpósio Ibero-americano de Tecnologias Educacionais: SITED, Araranguá,SC, abr. 2018.
- MARTINS, Julyan Figueredo; WEBER, Tiago Oliveira. Sistema Matricial de Interconexão Programável Para Aplicação em Laboratórios Remotos de Eletrônica Analógica. 7 º Simpósio de Integração Científica e Tecnológica do Sul Catarinense: Sict-Sul, Araranguá,SC, out. 2018.

2 CONCEITOS TEÓRICOS

Essa parte do trabalho servirá para apresentar e esclarecer alguns conceitos necessários para o entendimento do assunto desenvolvido neste trabalho.

Inicialmente, será abordado o tema de laboratórios remotos de forma geral. Após estudar sua classificação, vantagens, componentes e comunicação, serão analisados os laboratórios remotos para eletrônica analógica.

Em (KAZMIERKOWSKI; LISERRE, 2008) são definidos conceitos importantes que serão apresentados a seguir.

2.1 CARACTERIZAÇÃO DOS LABORATÓRIOS REMOTOS

Segundo (KAZMIERKOWSKI; LISERRE, 2008), qualquer experimentação pode ser definida por três critérios:

1. tipo de interação do usuário com o experimento;
2. qual a natureza do experimento;
3. tipo de localização do usuário em relação ao experimento.

O primeiro critério, referente a como o usuário interage com o experimento, dois tipos de interações são possíveis:

- laboratório tradicional, onde o usuário controla instrumentos e dispositivos diretamente;
- o usuário controla os instrumentos e dispositivos por meio de uma interface computacional.

Juntando os outros dois critérios, temos a Figura 1 que mostra os tipos de experimentos divididos entre experimentos físicos reais, e experimentos virtuais ou simplesmente simulados.

Considerando os experimentos onde o usuário está distante do experimentos, também chamados de experimentos remotos, tem-se três distinções:

- Os Laboratórios Virtuais concebidos somente com simulações remotas são chamados de Laboratórios Simulados. Podem ser disponibilizados na internet de maneira aberta para os usuários usarem em seus computadores pessoais, ou podem ser implementados em servidores dedicados. Os Laboratórios Virtuais oferecem

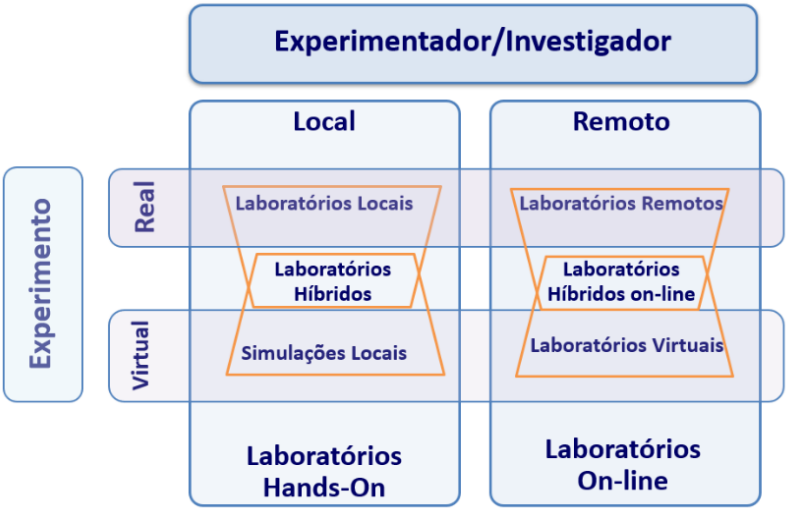


Figura 1: Caracterização dos laboratórios
Fonte: adaptado de (ZUTIN et al., 2010).

muita versatilidade ao usuário, pois os experimentos podem ser repetidos várias vezes, além de não oferecer risco algum. Podem ser poderosas ferramentas de ensino.

- Laboratório remotos reais proveem ao usuário acesso remoto, por meio da internet, aos instrumentos e dispositivos reais encontrados em laboratórios. Existem as mais diversas implementações de laboratórios desse tipo. Esse tipo de laboratório é o foco do presente trabalho.

- Laboratórios Híbridos são aqueles que combinam recursos laboratoriais reais, físicos, com softwares de simulação.

2.2 BENEFÍCIOS DOS LABORATÓRIOS REMOTOS

Segundo (KAZMIERKOWSKI; LISERRE, 2008), os principais benefícios dos laboratórios remotos são os seguintes:

- Os horários que o estudante pode usar o laboratório são flexíveis;

- Estudantes podem usar o laboratório remoto como um substituto na ausência do laboratório presencial;
- Mais fácil de agendar experimentos;
- Há maior aproveitamento dos recursos das instituições que implementam os laboratórios remoto, pois mais pessoas podem usar o laboratório pela internet, enquanto que o laboratório presencial tem acesso mais restrito;
- Aumenta a contribuição e troca de conhecimentos e recursos tanto de alunos como de instituições de várias partes do mundo;
- Dá mais liberdade e autonomia ao estudante para fazer seus experimentos;
- Ajuda os estudante que possuem necessidades especiais;
- Diminui a incidência de danos aos equipamentos laboratoriais, devido aos mecanismos presentes nos laboratórios remotos que impedem que o usuário faça experimentos que danifiquem os equipamentos;
- É uma tecnologia muito bem vinda para os cursos a distância(DíAZ et al., 2013).

2.3 COMPONENTES DE UM LABORATÓRIO REMOTO

Existem diversas implementações de laboratórios remotos pelo mundo. Mesmo com pouca ou nenhuma compatibilidade entre as implementações, algumas partes são identificadas em qualquer laboratório remoto(GARCIA-ZUBIA; DIEGO; PABLO, 2005). Os requisitos mostrados abaixo são os mais comuns, o que não constitui uma regra para as implementações.

- Deve ser possível realizar experimentos;
- Equipamentos de instrumentação para controlar e medir resultados do sistema;
- Servidor que faça o controle dos equipamentos de medição e atuadores do laboratório remoto;
- Servidor que, por meio da internet e conexão cliente-servidor, receba os parâmetros e requisições do usuário;

- Na parte do cliente é comum haver alguma interface, o mais realista possível, para o controle dos experimentos.

O acesso do usuário na parte cliente da aplicação cliente-servidor pode ser feita com aplicações Desktop. Com a evolução das tecnologias web, é notada uma tendência cada vez maior das mesmas serem adotadas como padrão para criação das interface do usuário(GARCIA-ZUBIA et al., 2017b).

2.4 COMUNICAÇÃO ENTRE COMPONENTES DO LABORATÓRIO REMOTO

A comunicação entre o servidor do laboratório e os equipamentos de instrumentação podem usar alguns padrões como TCP-IP, RS-232, IEEE-488. Na comunicação entre software do servidor do laboratório remoto e os equipamentos de instrumentalização, a tecnologia *LabView*, tem grande destaque. Os equipamentos de instrumentalização usados também influenciam na escolha da tecnologia para a criação do software de controle dos equipamentos.

2.5 LABORATÓRIOS REMOTOS DE ELETRÔNICA ANALÓGICA

Dentro do escopo dos laboratórios remotos, existem aqueles específicos para experimentação de eletrônica analógica. Nestes laboratórios a configuração cliente-servidor é comum, como nos outros tipos de laboratórios remotos.

A mediação entre o servidor de um lado e os instrumentos de medição e componentes eletrônicos de outro fica a cargo da matriz de interconexão eletrônica. A matriz de interconexões eletrônica merece especial atenção, pois é o foco do presente trabalho. Como mencionado anteriormente, aqui será mostrada a implementação de uma matriz com tecnologia pouco convencional.

2.6 RELÊS ELETROMECÂNICOS

Os relês eletromecânicos são os componentes fundamentais de muitas das matrizes de interconexões usadas nos laboratórios remotos de eletrônica analógica que serão mostrados no próximo capítulo. Um relê eletromecânico funciona como chave eletrônica. São cinco os pinos

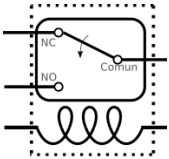


Figura 2: Relê Eletromecânico
Fonte: elaborado do autor.

do relês. Dois pinos são da bobina ou indutor. Os outros três são: comum, normalmente fechado(NO) e normalmente aberto(NC). Quando não há corrente na bobina, não há acionamento e o pino comum permanece conectado ao pino normalmente aberto. Quando há corrente na bobina, um campo elétrico é gerado, atraindo o contato interno em direção à bobina, fechando o contato com o pino normalmente fechado. Ocorre então o acionamento do sistema interfaceado pelo relê. A Figura 2 mostra o esquema de um relê genérico.

2.7 CHAVES COM TRANSISTORES MOSFET

Transistor MOSFET é, dos transistores, o mais difundido, sendo usado principalmente na fabricação de CIs. Comparado ao transistor de junção bipolar, por exemplo, o MOSFET é muito pequeno e de fabricação mais simples(SEDRA; SMITH, 2004). Dessa forma é possível colocar números imensos deste componente em áreas muito pequenas. A Figura 3 mostra detalhes da estrutura física do MOSFET.

O transistor é fabricado sob um substrato de tipo-p, que consiste em uma placa de cristal de silício que provê sustentação física para o dispositivo. Duas regiões altamente dopada do tipo-n, indicadas no desenho como *source* n+ e *drain* n+ (fonte e dreno) são cridas sobre o substrato. Uma fina camada de dióxido de silício(SiO_2), que serve como isolante, é posta sobre as áreas do *drain*, *source* e substrato. Uma camada de metal é posta sobre o dióxido de silício para formar o *gate* (porta). Contatos de metal são colocados sobre o *drain* e *source*. Esta é a estrutura do transistor NMOS, onde o substrato é do tipo-p, enquanto as regiões de dreno e fonte são do tipo-n. Invertendo o tipo dessas regiões, tem-se o transistor PMOS.

Na Figura 4, as regiões de dreno e fonte estão aterradas. É aplicada uma tensão positiva na porta. Como a fonte está aterrada, a tensão na porta é V_{gs}. A tensão positiva na porta causa, num primeiro

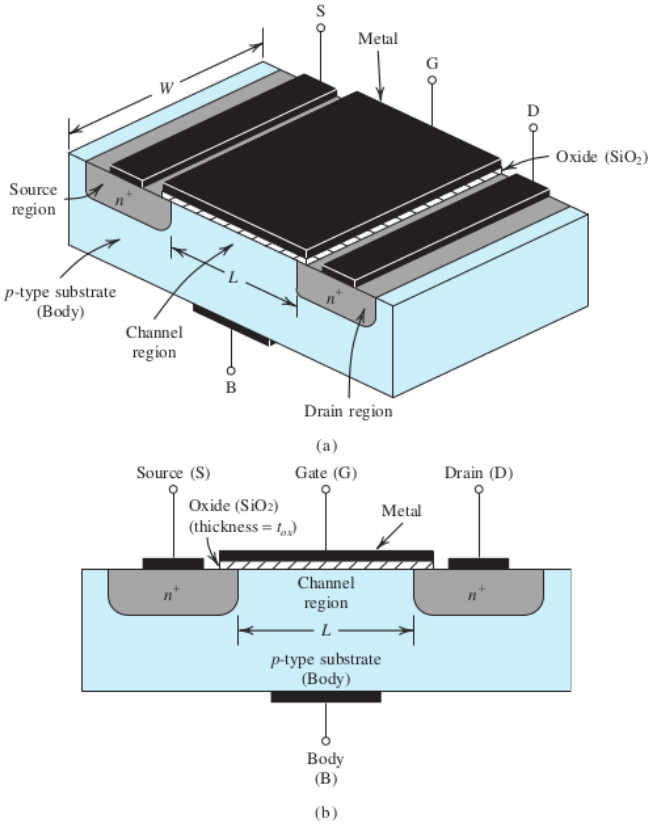


Figura 3: Estrutura física MOSFET
Fonte: adaptado de (SEDRA; SMITH, 2004).

momento, que os buracos livres, que são as cargas positivas, sejam repelidos da região do substrato sob a porta. Os buracos são empurrados para baixo, criando uma região com carga mais negativa, chamada região de depleção. Ao mesmo tempo que os elétrons das regiões de dreno e fonte são atraídos para a região do substrato logo abaixo da área da porta. Essa movimentação de cargas cria uma região de tipo-n sob a porta, conectando o dreno e a fonte, que também são do tipo-n. Forma-se assim um canal de condução elétrica entre os terminais de dreno e fonte. Neste estado, aplicando uma diferença de potencial entre dreno e fonte, corrente fluirá entre os dois terminais.

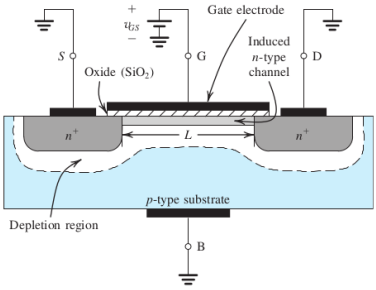


Figura 4: Canal de corrente no MOSFET
Fonte: adaptado de (SEDRA; SMITH, 2004).

A região da porta e o substrato abaixo dela formam um capacitor, tendo o dióxido de silício separando-as, servindo como dielétrico. A tensão positiva na porta causa um acumulo de carga positiva na placa metálica da porta. Isto induz uma carga negativa na área do substrato. Cria-se então um campo elétrico, e a intensidade desse campo controla a intensidade da carga negativa no substrato. Como essa carga negativa no substrato é o que define o tamanho da região do tipo-n induzida, consequentemente a tensão positiva na porta é o fator determinante que controla o tamanho do canal induzido entre dreno e fonte. O MOSFET se comporta como uma resistência variável controlada pela tensão da porta.

Uma chave eletrônica ideal possui resistência nula, quando acionada; resistência infinita quando aberta; e tempo de resposta nulo. O transistor MOSFET simples não possui essas características. Ele possui uma resistência variável e não linear. Porém, conectando transistores PMOS e NMOS em paralelo obtém-se o transistor CMOS, que possui resistências mais baixas e estáveis quando acionado. A Figura 5 mostra a estrutura do transistor CMOS. O transistor NMOS é fabricado diretamente no substrato tipo-p, enquanto que o transistor PMOS necessita de uma região tipo-n criada especialmente para ele.

2.8 MATRIZ DE INTERCONEXÃO ELETRÔNICA PROGRAMÁVEL

Para o desenvolvimento de laboratórios remotos de eletrônica analógica, o hardware fundamental é a implementação de chaves ana-

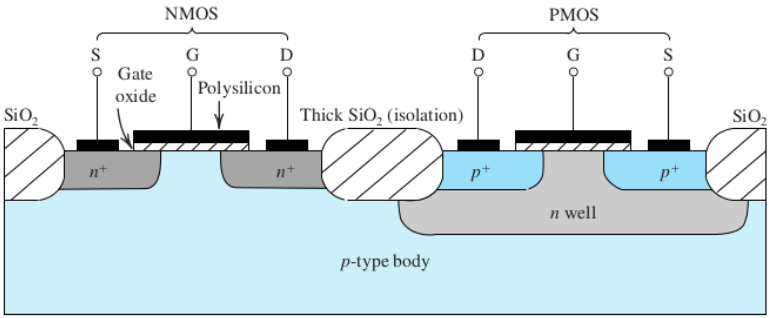


Figura 5: CMOS
Fonte: adaptado de (SEDRA; SMITH, 2004).

lógicas que possam interconectar os elementos de forma programável. Estas chaves permitem que determinados componentes do circuito es-

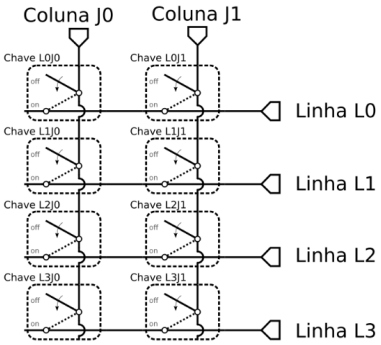


Figura 6: Chaves eletrônicas analógicas
Fonte: elaborado pelo autor.

tejam conectados ou não, de acordo com um valor digital salvo na memória do sistema. Neste trabalho, parte do objetivo é mostrar a viabilidade ou não de chaves eletrônicas com tecnologia MOS. A principal desvantagem destas chaves é que a resistências das chaves não é desprezível, quando acionadas. Segue na Figura 6 um esquema de matriz de interconexões de 2 colunas e 4 linhas.

3 ESTADO DA ARTE

Neste trabalho, é desenvolvida uma matriz de interconexões de componentes eletrônicos com tecnologia CMOS. Para avaliar se este tipo de implementação é viável para um laboratório remoto em comparação com as implementações já existentes, é necessário realizar um estudo sobre os trabalhos já desenvolvidos na área.

Neste capítulo, são estudados os principais projetos de laboratórios remotos de eletrônica analógica que servem de parâmetro e inspiração para este trabalho.

3.1 *NETLAB*

O *NetLab* é um laboratório remoto desenvolvido na *University of South Australia* em setembro de 2001(NEDIC; MACHOTKA; NAFALSKI, 2003). Foi desenvolvido com o intuito de auxiliar nos cursos de engenharia elétrica. O sistema é usado na própria universidade. Possui uma interface gráfica própria. Essa interface possui imagens de instrumentos laboratoriais que o laboratório possui, o que dá uma boa experiência ao aluno. Os alunos podem realizar os experimentos em grupos de até 3 pessoas, podendo se comunicar pelo *chat* do sistema.

Os usuários podem visualizar o experimento por meio de uma câmera de vídeo. O *NetLab* possui um *software* de interface chamado *CircuitBuilder*(NEDIC; MACHOTKA, 2007) que permite ao usuário fazer a conexão dos componentes eletrônicos disponíveis para o experimento, simulando a conexão em um laboratório real. Além disso, *NetLab* possui também componentes variáveis, como resistores variáveis, que podem ser manipulados pelo *software*. Para gerar essa resistência variável são usados relês controlados por microcontroladores que, quando comutados, adicionam ou retiram resistência dessa resistência variável. Também fazem parte do sistema um servidor que recebe as requisições, com *software* feito com a tecnologia *LabView*, da empresa *National Instruments*. O servidor processa os comandos do usuário e controla os instrumentos com a interface IEEE 4888.2, conhecida como *GPIB(General Purpose Interface Bus)*. Na figura 7 temos as as interfaces de usuário do *Netlab*.

Figura 7: Netlab

Fonte: (NETLAB, 2018)

3.2 *REMOTELECTLAB*

O *RemotElectLab*(SOUSA; ALVES; GERICOTA, 2010) nasceu com o intuito de ser um laboratório remoto que simulasse de maneira mais fiel possível o comportamento de um laboratório real. Deveria ser também reutilizável, ou seja, ter a flexibilidade de mudar os experimentos que serão realizados no laboratório se for preciso. A utilização desse laboratório remoto ficou muito atrelada aos experimentos realizados em sala de aula. O professor é quem define os experimentos que serão realizados pelo aluno em casa, após uma aula prática no laboratório

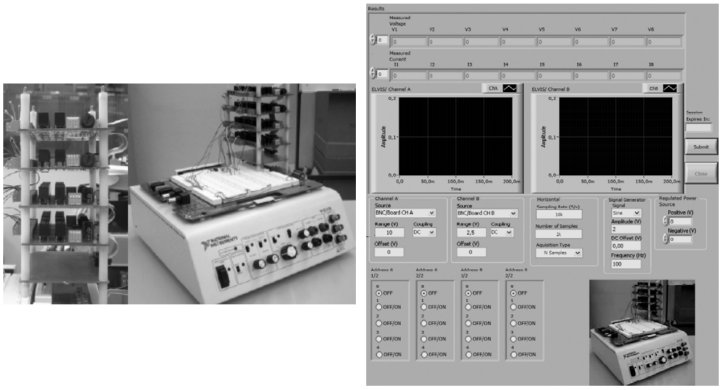
real.

O custo de implementação do laboratório fica igual ou mais caro do que o de um laboratório real, pois além de precisar de todos os equipamentos de instrumentalização dos laboratórios reais, ainda é necessário que esses equipamentos tenham as tecnologias necessárias ao acesso e controle remoto, interface e protocolos específicos.

O *RemotElectLab* usa o *NI-ELVIS(National Instruments Educational Laboratory Virtual Instruments Suite)*, que é uma plataforma que pode ser usada tanto no laboratório remoto como em um real. *NI-ELVIS* vem equipado com uma *protoboard* para o acoplamento dos componentes dos experimentos, 12 instrumentos integrados de uso de laboratórios de elétrica, entradas e saídas analógicas. Além disso, essa plataforma é controlada com a linguagem *LabView* que facilita a configuração do equipamento. O *NI-ELVIS* é conectado e controlado por um servidor que fará então a conexão com a internet.

Para fazer as interconexões dos componentes e dos instrumentos, foi usada uma matriz de interconexões com relês eletromecânicos ao invés de qualquer alternativa de chaveamento por transistores. A justificativa é que, para que o laboratório remoto fique mais fidedigno à um laboratório real, não seria possível usar multiplexadores feitos com transistores devido as resistências e capacitâncias internas, além das limitações de tensão. Essa é uma observação importante para o decorrer do presente trabalho. Na figura 8 temos a interface para usuário e parte do *hardware* que compõe o sistema.

Figura 8: *RemoteElectLab - hardware(esq.) e User interface(dir.)*



Fonte: Hernandez et al. (2010)

3.3 *ISILAB*

O *ISILab* é um laboratório remoto e compartilhado, desenvolvido por meio da centralização dos recursos de vários laboratórios em um único laboratório virtual, acessado através de uma página web. O *hardware* utilizado foi o mesmo do *RemotElectLab*. As funcionalidades principais dos laboratórios compartilhados são:

- possibilitar ao usuário desenvolver experimentos reais em vários laboratórios reais, acessados remotamente pela internet;
- dar acesso a qualquer laboratório conectado a internet;
- evitar que o usuário tenha que instalar ou adquirir qualquer *hardware* ou *software* especial;
- permitir o acesso ao sistema pelo maior número possível de pessoas;
- permitir aos professores criar e supervisionar experimentos;
- criar um sistema de controle de acesso ao sistema.

O VLS(*Virtual Laboratory Server*) faz o controle de acesso dos usuários à um laboratório virtual. Nesse laboratório virtual há uma breve descrição dos experimentos disponíveis dos laboratórios reais, quantas pessoas estão usando o laboratório e a localização geográfica do laboratório. O VLS verifica se o experimento escolhido pelo usuário está disponível e faz, então, a conexão do usuário com o laboratório real. O RLS(*Real Laboratory Server*) controla os instrumentos do laboratório real. Controla também as configurações do experimento escolhidas pelo usuário e faz a separação entre a concorrência dos vários usuários que podem usar o sistema ao mesmo tempo.

Esse laboratório remoto usa o *LabVIEW* da *National Instruments*, que segundo os autores(CHIRICO; SCAPOLLA; BAGNASCO, 2005) é um ambiente gráfico de programação utilizado na industria e em sistemas de aprendizado remoto. Na linguagem, temos blocos chamados VI(*Virtual Instruments*), que representam os instrumentos de laboratório, descrevendo o comportamento físico deste instrumento, além de dar ao programador uma interface gráfica para o instrumento laboratorial específico. O *LabVIEW* facilitou muito o desenvolvimento dos laboratórios remotos. Foram integradas ao *LabVIEW* algumas aplicações em Java, o que aumentou o realismo do laboratório *ISILab*.

3.4 VISIR

VISIR(*Virtual Instrument Systems in Reality*) é um projeto iniciado em 1999, na *Blekinge Institute of Technology*, conhecida como BTH. Na BTH, exitiam à época, estações de trabalho, contendo vários equipamentos comuns em laboratórios de eltrônica, que permitiam aos alunos fazer os experimentos das aulas dos cursos de engenharia e tecnologia(GUSTAVSSON; ZACKRISSON; OLSSON, 2018). Cada estação de trabalho podia ser usada por um aluno ou por grupos reduzidos, restrição de comum a qualquer laboratório real. Com o intuito de aumentar a capacidade dessas estações de trabalho, desenvolveu-se a ideia de laboratórios remotos, onde os recursos das estações de trabalho possam ser utilizados por mais de um aluno simultaneamente, por meio da internet.

Em sua implementação original, o VISIR contava com um computador conectado a internet como sendo o servidor. Esse computador por sua vez conecta-se ao PXI(*PCI Extensions for Instrumentation*), que contém placas eletrônicas da National Instruments. Essas placas que são conectadas à esse chassis PXI são os instrumentos laboratoriais de medição eletrônica, como gerador de funções, multímetro digital, osciloscópio e uma placa de entradas e saídas digitais. Os parâmetros e configurações são controlados pelo computador.

Para fazer a conexão dos componentes, temos uma matriz de comutação. Essa matriz é formada por placas com relês eletromecânicos para o chaveamento. Nessas placas são conectados os componentes que farão parte dos experimentos de circuitos eletrônicos. Essas placas são feitas de forma que a matriz seja expansível, ou seja, quanto mais placas desenvolvidas, maior o tamanho da matriz que interconectará os componentes. Nessa versão inicial do VISIR, com 6 placas, a matriz pode acomodar até 50 resistores, com 5 nós para o usuário utilizar. Também aqui foi usado a ferramenta *LabView* para programar os instrumentos da *National Instruments* na parte do servidor.

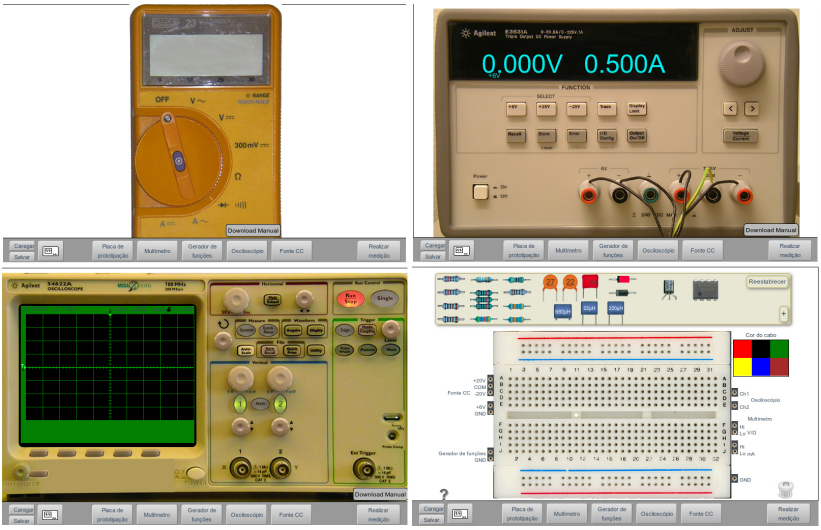
Por meio da internet e de um *software* cliente, que pode ser implementado com diferentes linguagens, o usuário é conectado ao servidor. O *software* cliente dispõe de uma *protoboard* virtual onde o usuário pode conectar os componentes eletrônicos que estão a sua disposição. A interface gráfica também possui osciloscópio, multímetro, amperímetro e gerador de funções para o usuário. Após fazer as conexões dos componentes na *protoboard*, mudar as configurações dos instrumentos de medição, o usuário deve mandar o experimento, clicando no botão "Realizar Medição". O servidor receberá uma *netlist* das conexões dos

componentes e valores de configuração dos instrumentos. Esses dados caem numa fila de espera, e serão analisados pelo *software* do servidor. Se não houver nenhum erro na *netlist* e configurações, a fonte de alimentação do sistema é ligada.

Por ser o laboratório remoto mais avançado na área de experimentação eletrônica (GARCIA-ZUBIA et al., 2017b), e pelo interesse que o trabalho aqui apresentado tem na matriz de interconexão, ao VISIR foi dada uma atenção maior do que ao demais laboratórios.

O VISIR trouxe uma inovação muito importante aos laboratórios remotos de eletrônica analógica. O VISIR nasce com a proposta de deixar que o usuário use componentes disponíveis da maneira que desejar, e forme os experimentos de forma livre, respeitando as limitações do sistema, enquanto que nos outros laboratórios remotos, a ideia de que o laboratório remoto é somente uma ferramenta para complementar o aprendizado feito no laboratório tradicional estava muito presente(GUSTAVSSON et al., 2005). Na figura 9 são apresentadas as interfaces gráficas do usuário.

Figura 9: Osciloscópio VISIR



Fonte: adaptado de Rexlab - https://rexlab.ufsc.br/

4 O SISTEMA PROPOSTO

Será apresentado nessa seção o sistema proposto, incluindo as placas com as matrizes de interconexão, *firmware*, *software* e configuração do sistema. Ressalta-se que, apesar deste trabalho ter foco em aplicações de laboratório remoto, o foco do desenvolvimento foi na implementação das matrizes e configuração através de um computador local.

A matriz serve basicamente para conectar os componentes de um circuito. Em um laboratório remoto, como em um laboratório presencial, o usuário tem ao seu dispor uma variedade limitada de componentes eletrônicos e de instrumentos laboratoriais. Para fazer a conexão desses componentes e instrumentos, é necessário um mecanismo que possa atender as mais variadas configurações de circuitos eletrônicos possíveis. A matriz de interconexões é configurada para fazer essas conexões de componentes.

4.1 VISÃO GERAL

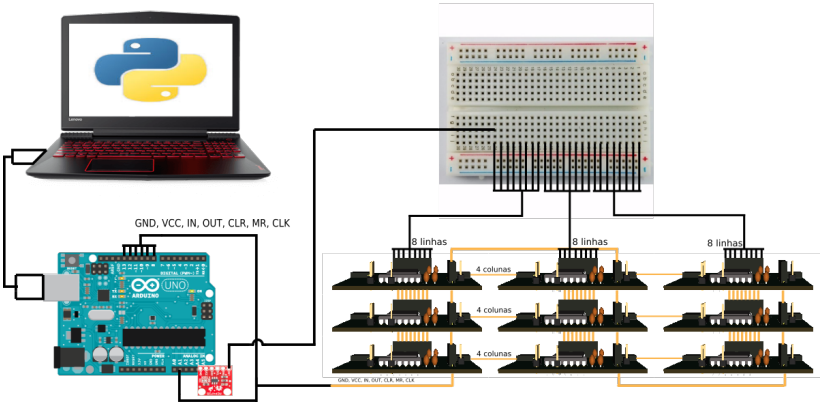
O sistema da matriz de interconexões eletrônicas para laboratório remoto e prototipagem rápida é formada por três blocos principais:

- ***software* de controle para o usuário**: responsável por receber as conexões desejadas pelo usuário e transmití-las para a placa, bem como por mostrar os resultados vindos da placa;
- **placa microcontrolada**: responsável por, a partir dos dados passados pelo computador, fazer a configuração do sistema serial;
- **sistema matricial de interconexão**: constituído de 9 placas eletrônicas que formam a matriz de interconexões propriamente ditas; Nela são conectados os componentes cujos terminais serão ligados entre si através da matriz.

Na Figura 10 estão postas todas as partes que compõe o sistema. Há um computador que se comunica com um *Arduino*. Esse *Arduino* manda sinais de configuração para a matriz de interconexões. As 9 placas da matriz estão dispostas no desenho e suas linhas, 24 no total, estão conectadas a uma pequena *protoboard*, onde são conectados os componentes eletrônicos disponíveis ao usuário. No canto direito da placa do *Arduino* está representado o DAC, conversor digital analógico, que

gerará as tensões de entrada dos circuitos. A saída desse DAC está conectada a uma linha qualquer da matriz de interconexão. Basta ao usuário, quando configurar a matriz, conectar essa saída do DAC no ponto específico do seu circuito experimental.

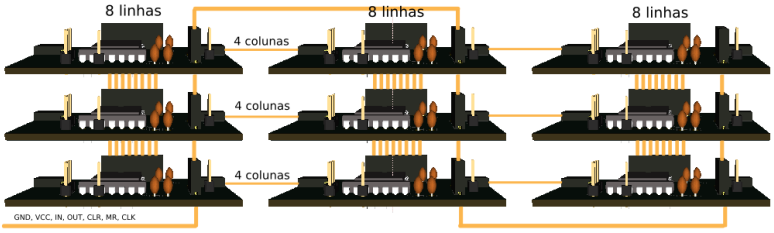
Figura 10: Visão geral



Fonte: elaborado pelo autor.

O *hardware* desenvolvido no trabalho está ilustrado com mais detalhes pela figura 11.

Figura 11: conexão das 9 placas



Fonte: elaborado pelo autor.

5 A MATRIZ DE INTERCONEXÃO

Analisando muitas das implementações de laboratórios remotos de eletrônica analógica, nota-se que parte essencial desses sistemas é a matriz de interconexão eletrônica. Essa matriz serve para conectar os componentes de maneira a formar os circuitos eletrônicos dos experimentos. Em todas as implementações vistas, usou-se relês eletromecânicos para fazer os chaveamentos e conectar os componentes dos experimentos. O chaveamento com tecnologia MOS parece pouco apropriado para esse tipo de aplicação, tendo em vista que chaves implementadas com transistores apresentam resistências e capacitâncias. Em contrapartida, os contatos dos relês eletromecânicos tem resistência virtualmente zero.

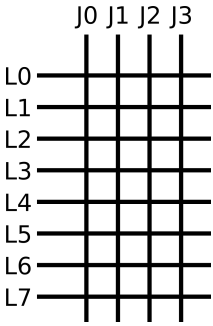
Em (SOUSA; ALVES; GERICOTA, 2010), a hipótese de usar tecnologia MOS é descartada em virtude das limitações de tensão e corrente adicionadas pelas chaves no experimento. Um dos objetivos deste trabalho é averiguar com mais rigor, por meio de uma implementação, se é possível montar a matriz de interconexão usando chaves com tecnologia MOS, levando em conta as limitações intrínsecas a este tipo de chaveamento na hora de desenvolver os experimentos.

5.1 PLACA DA MATRIZ DE INTERCONEXÕES

A placa matricial, peça chave do sistema, é constituída de dois componentes básicos, um multiplexador matricial e um registrador de deslocamento. O mais importante deles é o multiplexador matricial, que faz as conexões entre os terminais dos componentes eletrônicos usados nos experimentos. Foi usado o CI MT8804A, que contém internamente uma matriz de interconexões de 8 linhas por 4 colunas, como mostra a Figura 12. Todas as linhas podem ser conectadas à todas as colunas, inclusive ao mesmo tempo, totalizando 32 possíveis conexões. Existem portanto 32 chaves eletrônicas internas ao CI. Cada linha é controlada por um registrador de 4 bits, representando as 4 colunas. Desse modo, se, no registrador da linha 0, o bit da coluna 3 estiver em nível alto, a linha 0 será conectada à coluna 3.

A escalabilidade é uma propriedade desejável para as matrizes de interconexões. É possível conectar vários CIs MT8804, aumentando a matriz. Na Figura 13 é apresentada a conexão entre dois MT8804 que aparece no *datasheet* do componente, formando uma matriz de 8x8.

Figura 12: Representação da matriz interna do MT8804A



Fonte: elaborado pelo autor.

As linhas dos dois CIs são interconectadas. Então, para conectar as linhas com as colunas do CI da esquerda é necessário que somente este CI tenha seus registradores das linhas configurados. Da mesma forma acontece com as linhas da direita. Para fazer a conexão entre duas colunas basta que essas colunas sejam conectadas na mesma linha. No sistema implementado, 9 placas são conectadas usando o mesmo raciocínio. No caso deste trabalho, são as linhas que são interconectadas, e as colunas fazem o intermédio.

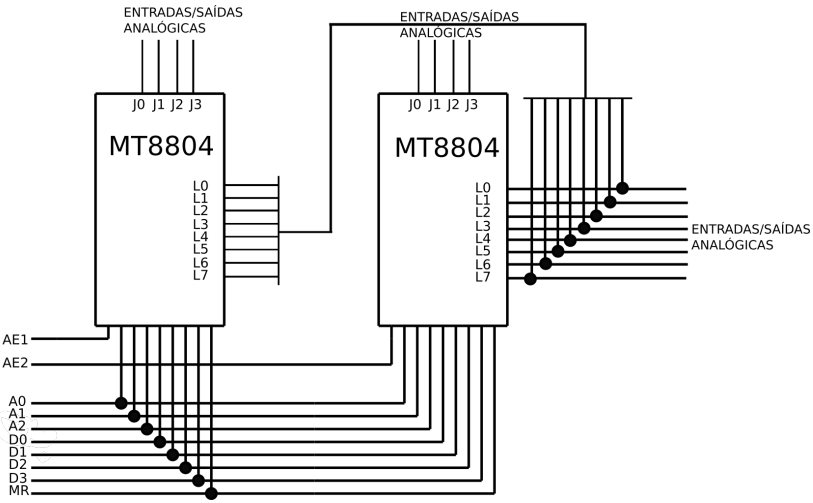
O segundo componente que faz parte da placa matricial é um registrador de deslocamento, ou *shift-register*, e o CI usado foi o SN74HC164. Esse CI é um conversor serial-paralelo de 8 bits.

Usando dois pinos, *clock* e dados, é possível gerar uma saída paralela de 8 bits, dos quais 7 serão os bits de configuração das linhas do MT8804, e o bit excedente será o sinal de saída da placa. Esse bit de saída é conectado à entrada da placa seguinte, dando ao sistema a característica de escalabilidade. Na figura 14 são mostradas as conexões internas de uma placa do sistema. Os pinos A e B do 74HC164 formam a entrada de dados e os pinos Q são as 8 saídas. Também tem o pinos de *clock* e *clear* que coloca as saídas em nível baixo. No MT8804, os pinos D são entrada de dados, os pinos A são para sinais de endereço dos registradores internos das linhas da matriz.

5.2 CONFIGURAÇÃO DE UMA PLACA MATRICIAL

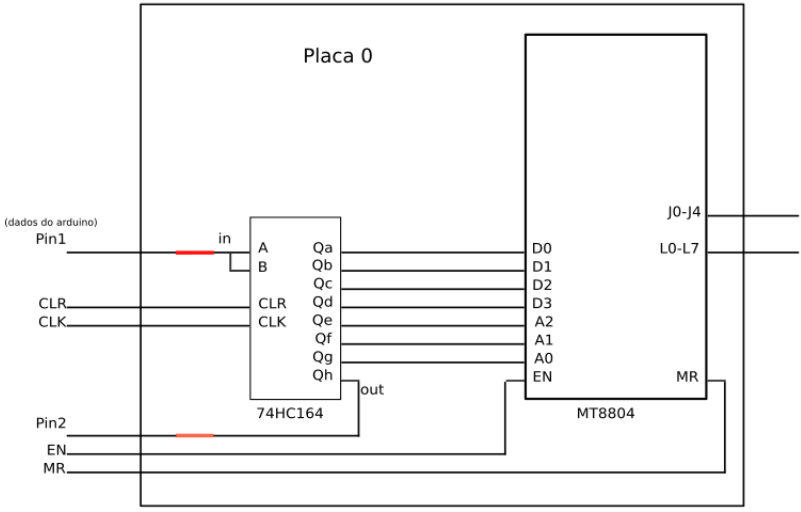
São necessários 7 bits para configurar qualquer uma das 8 linhas, sendo 3 bits para controlar qual a linha será configurada, enquanto os

Figura 13: Conexões entre dois MT8804A)



Fonte: adaptado do *datasheet*.

Figura 14: Esquema lógico de uma placa



Fonte: elaborado pelo autor.

4 bits restantes são os dados a serem guardados no registrador mencionado anteriormente. Cada MT8804 tem 8 registradores, um para cada linha da matriz interna.

Para mudar o valor do registrador da linha é necessário um sinal de *enable*, fornecido por meio do pino AE do MT8804A. O pino MR, que recebe o sinal de *master reset*, serve para apagar todas as conexões entre linhas e colunas. Quando uma conexão entre linha e coluna é feita, a resistência dessa junção é aproximadamente de 200Ω, valor este adquirido experimentalmente. Quando não há conexão, há um circuito aberto nessa junção, com resistência virtualmente infinita.

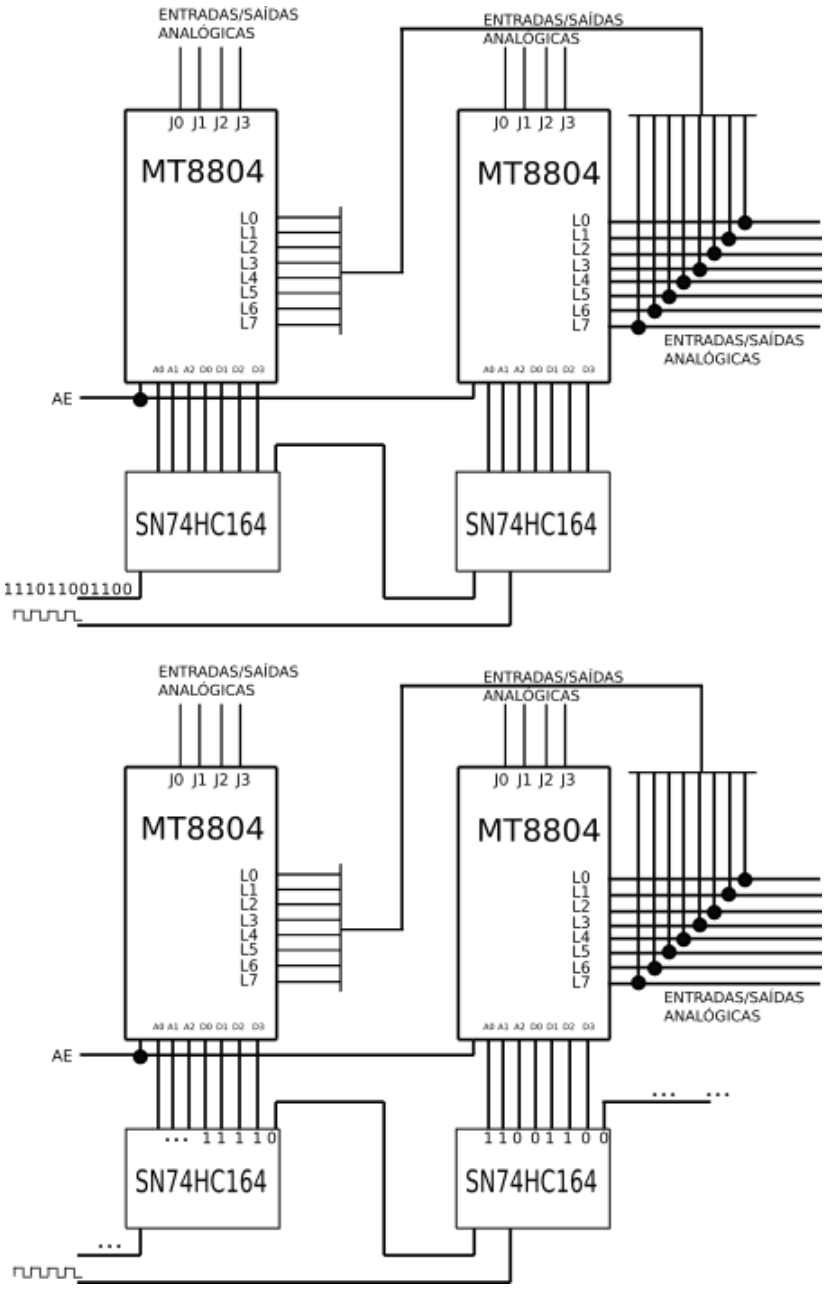
Todas as placas recebem os sinais de configuração de uma placa de *Arduino*. Do *Arduino* chegam sinais de *clock* e dados ao CI registrador de deslocamento. Acontece entre o Arduino e o registrador de deslocamento uma conversão serial-paralelo, onde os 8 bits desse frame de bits são postos nas 8 saídas do CI registrador de deslocamento. Dentre esses 8 bits estão 3 bits de endereço, que mapeiam as 8 linhas possível, 4 bits dados, que acionam as 4 colunas possíveis da matriz do MT8804. Por fim, 1 bit é a saída da placa. Esse bit de saída, por meio do pino de saída, chega à entrada da próxima placa.

5.3 CONFIGURAÇÃO DE VÁRIAS PLACAS

O bit que sai pelo pino de saída da primeira placa do sistema, chega à entrada da segunda placa. Com o sinal de *clock*, este sinal de dados que vem da primeira placa será colocado na saída do CI registrador de deslocamentos desta segunda placa. A segunda placa, por sua vez terá um pino de saída, que será conectado ao pino de entrada da próxima placa, e assim por diante. O sistema foi implementado com 9 placas matriciais, então existem 9 CI registradores de deslocamento. Pode-se então considerar que esses 9 CIs formam um conversor serial-paralelo de 1 entrada para 72 saídas. O primeiro bit, que entrou na primeira placa, ao final do processo, estará no último pino de saída da última placa. A Figura 15 ilustra a ideia.

As saídas do SN74HC164 estão ligadas aos pinos de configuração do MT8804. Cada MT8804 recebe 7 desses bits, que estão organizados para a configuração de um linha da matriz interna do CI. Ao sinal de *enable* (AE), os 4 bits de configuração são gravados no registrador da linha especificada pelos 3 bits de endereço.

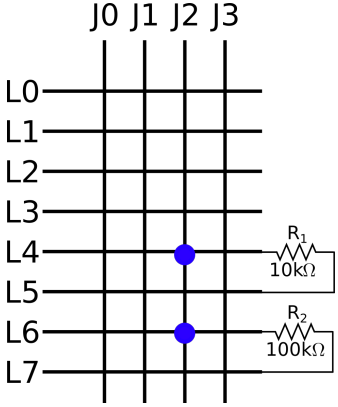
Figura 15: Comunicação entre placas



Fonte: elaborado pelo autor.

5.4 RESISTÊNCIAS CARACTERÍSTICAS

Quando não há conexão entre linha e coluna, há um circuito aberto nessa junção. Quando uma conexão entre linha e coluna é feita, a resistência dessa junção é aproximadamente de 200Ω. A Figura 16 exemplifica. Para fazer a conexão entre dois componentes, faz-se a



(a) Conexão dos resistores na matriz

(b) Resistências características nas conexões

Figura 16: Interligação de componentes em uma matriz
Fonte: elaborado pelo autor.

conexão entre duas linhas, intermediada por uma coluna. Por exemplo, para conectar os resistores de 10kΩ e 100kΩ, as linhas 4 e 6 devem ser conectadas à mesma coluna, por exemplo a coluna 7, e essa coluna exercerá o papel de nó nesse circuito simples, como mostrado na figura 16(a). As resistências características são mostradas na figura 16(b).

5.5 COMPONENTES ELETRÔNICOS DISPONÍVEIS

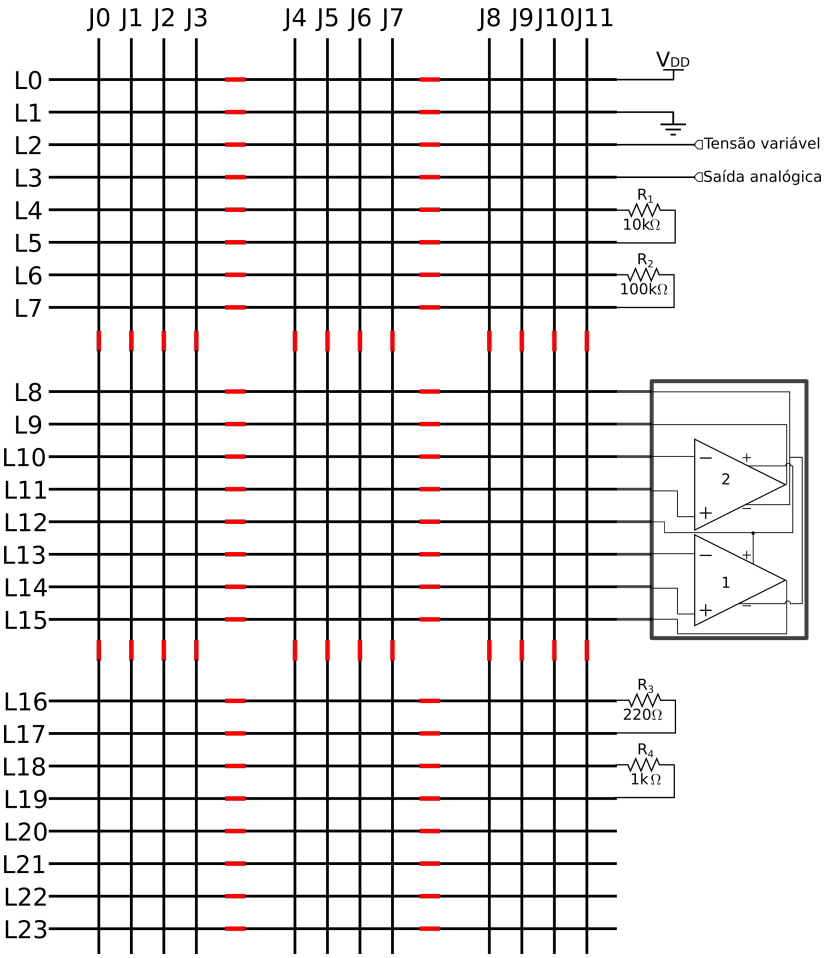
A quantidade de placas influencia no tamanho da matriz de interconexões. Como o sistema tem 9 placas, separadas em 3 colunas de 3

placas cada, a matriz formada possui 24 linha e 12 colunas. Os nós dos circuitos eletrônicos experimentados serão postos nas colunas, enquanto que os terminais dos componentes eletrônicos estarão nas linhas, como já foi dito. A figura 17 e a tabela 1 mostram a matriz completa do sistema, bem como os componentes disponíveis ao usuário. O usuário está restrito a usar os componentes disponíveis no sistema. Pode-se mudar esses componentes dependendo dos circuitos que se queira montar. Já a tabela 2 dá informações sobre diferentes configurações possíveis de escalonamento das placas.

Tabela 1: Componentes disponíveis ao usuário

linha	terminais
L0	GND
L1	VCC
L2	tensão variável
L3	saída analógica
L4	Resistor 10k pino 1
L5	Resistor 10k pino 2
L6	Resistor 100k pino 1
L7	Resistor 100k pino 2
L8	Lm324 pino gnd
L9	Lm324 pino out(2)
L10	Lm324 pino in-(2)
L11	Lm324 pino in+(2)
L12	Lm324 pino vcc
L13	Lm324 pino in-(1)
L14	Lm324 pino in+(1)
L15	Lm324 pino out(1)
L16	Resistor 220 pino 1
L17	Resistor 220 pino 2
L18	Resistor 1k pino 1
L19	Resistor 1k pino 2
L20	Não conectado
L21	Não conectado
L22	Não conectado
L23	Não conectado

Figura 17: Matriz completa com componentes



Fonte: elaborado pelo autor.

5.6 *FIRMWARE*, *SOFTWARE* E CONFIGURAÇÃO

Nesse estágio do trabalho, ainda não há um laboratório remoto, pois o foco do trabalho até aqui foi desenvolver o *hardware* da matriz de interconexões. As placas matriciais foram testadas primeiro individualmente e depois como o sistema completo de 9 placas conectadas,

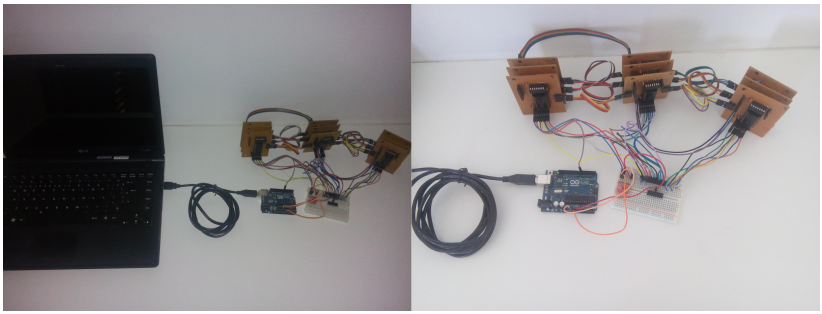
Tabela 2: Exemplos de possíveis configurações de matriz de interconexão

Nº de placas	Disposição das placas	Nós	Terminais
1	1x1	4	8
2	1x2	8	8
4	2x2	8	16
9	3x3	12	24
12	3x4	16	24
16	4x4	24	48

formando a matriz de 24 linhas e 12 colunas.

Foi feito um programa em linguagem de *Python* que serve de interface para o usuário de um computador. O programa em *Python* e o *Arduino* se comunicam por comunicação serial. O *Arduino* é conectado à primeira placa do sistema de 9 placas matriciais. Duas figuras do sistema físico completo estão mostradas na figura 18. À direita está o sistema com mais destaque, e à esquerda está o sistema conectado ao computador.

Figura 18: Sistema Completo



Fonte: elaborado pelo autor.

As placas são conectadas de maneira que os sinais elétricos enviados pelo *Arduino* cheguem de alguma forma a todas as placas. Os sinais de *clock*, *master reset*, *enable*, *clear*, alimentação e *gnd* chegam a todas as placas ao mesmo tempo, enquanto o sinal de dados chega somente à primeira placa. Para uma placa, 8 bits são necessário para configurar uma linha. O sistema possui 9 placas, e cada placa possui 8 linhas. São necessários então 8 bits x 8 linhas x 9 placas, totalizando

576 bits, ou 72 *bytes* de informação para configurar a matriz inteira.

Como explicado anteriormente, é necessário que o *Arduino* envie 8 *frames* de bits, de 72 bits cada. Todos os bits chegam na primeira placa e são deslocados pelo registrador de deslocamento(*shift-register*)para frente quando a placa recebe o sinal de *clock*. Todas as placas tem um pino de saída, que é conectado ao pino de entrada de dados da próxima placa. Assim, pode-se deslocar os bits da primeira até a última placa. Quando todos os 72 bits foram enviados, cada uma das nove placas terá 8 bits na entrada do CI multiplexador matricial. Com um sinal de *enable*, esses bits são gravados no registrador deste CI, efetuando a configuração da linha específica. Por convenção, todas as linhas 0 são configuradas, depois todas as linhas 1 e assim por diante até a linha 7. A organização desses bits na ordem correta é feita pelo programa em *Python*. Quando esse processo acaba, a matriz de interconexões está configurada.

Essa interface também possibilita o controle do MPC4725, um conversor digital analógico. O MPC4725 tem uma resolução de 12 bits. Funciona com tensões de 3V a 5V. Possui saída *rail-to-rail*, que foi importante devido às limitações de tensão do *Arduino*. Dessa forma, usando a alimentação do próprio *Arduino*, é possível gerar tensões de 0 a 5V com o DAC. Existem bibliotecas prontas para o *Arduino*, feitas pelo fabricante do MPC4725, que possibilitam a criação de sinais oscilatórios, como uma senoide. O usuário pode configurar a tensão de entrada adequada para o seu circuito. O pino de saída do DAC está conectado à uma das linhas da matriz, então o usuário configura os pontos do circuito experimental onde estará a tensão de entrada.

Também é possível controlar uma das entradas analógica do *Arduino* e ler o valor na *interface*. A entrada analógica também está conectada à uma das linhas da matriz, assim o usuário escolhe o ponto onde fará a leitura de tensão. Com um sinal do usuário pela interface, o *Arduino* lê a tensão no ponto específico e envia a informação para a interface. Assim é feita a interação com o seu circuito experimental.

6 TESTES E RESULTADOS

Três testes foram realizados para validação do sistema matricial. Os dois primeiros são circuitos resistivos divisores de tensão, com tensão de entrada formada por uma senoidal. A captação dos valores da onda de saída foi feita com entrada analógica do próprio *Arduino*. Para gerar a forma de onda na interface, foi usada a biblioteca gráfica *matplotlib*(HUNTER, 2007) do *Python*. No último experimento foi utilizado um amplificador operacional.

6.1 PRIMEIRO EXPERIMENTO

No primeiro circuito resistivo foram usados os resistores de 1kΩ e 220Ω, com 5% de variação. Abaixo temos as fórmulas do divisor de tensão.

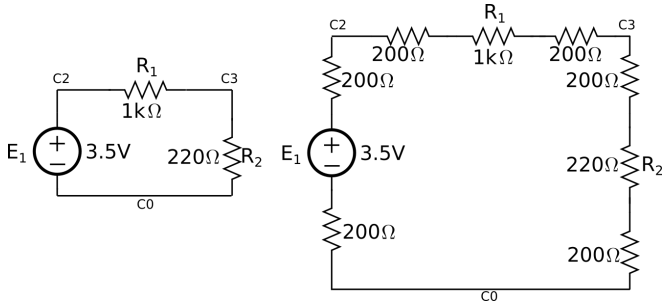
$$V_{saída\ ideal} = \frac{R2}{R1 + R2} \times V_{entrada} = \frac{220\Omega}{1k\Omega + 220\Omega} \times 3.5v = 0,631v$$

$$V_{saída\ real} = \frac{R2}{R1 + R2} \times V_{entrada} = \frac{820\Omega}{1600\Omega + 820\Omega} \times 3.5v = 1,186v$$

$$V_{entrada\ real} = \frac{2220\Omega}{200\Omega + 2220\Omega} \times 3.5v = 3,21v$$

Na Figura 19 temos à esquerda o circuito projetado, enquanto que à direita temos o circuito já considerando as resistências características.

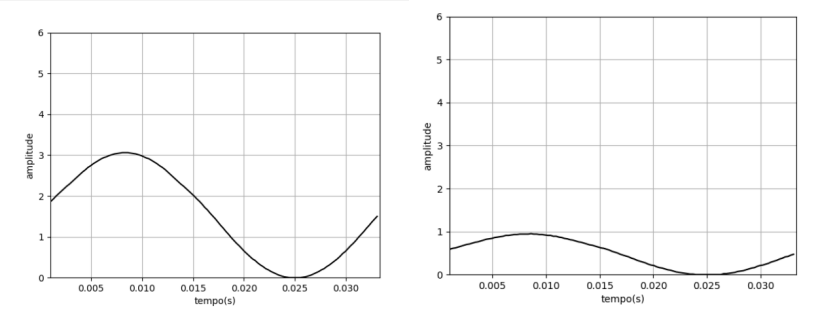
Figura 19: Circuito 1



Fonte: elaborado pelo autor.

Na Figura 20, à esquerda está o sinal de entrada e à direita o de saída. Considerando o divisor de tensão real que ocorre devido aos resistores do experimento e às resistências características, vê-se que o circuito apresentou ou valores corretos. Porém o valor de saída real teve uma diferença muito grande do valor ideal, com erro relativo de 46,79%. A entrada também é afetada, pois neste circuito, a entrada está sendo medida no ponto C2. O erro relativo da entrada é cerca de 9,01%.

Figura 20: Resultado do teste resistivo 1



Fonte: elaborado pelo autor.

6.2 SEGUNDO EXPERIMENTO

O segundo circuito testado está apresentado na Figura 21. Ele consiste em um divisor de tensão com dois resistores de 10kΩ e 5% de variação no valor nominal. Abaixo estão as fórmulas do divisor.

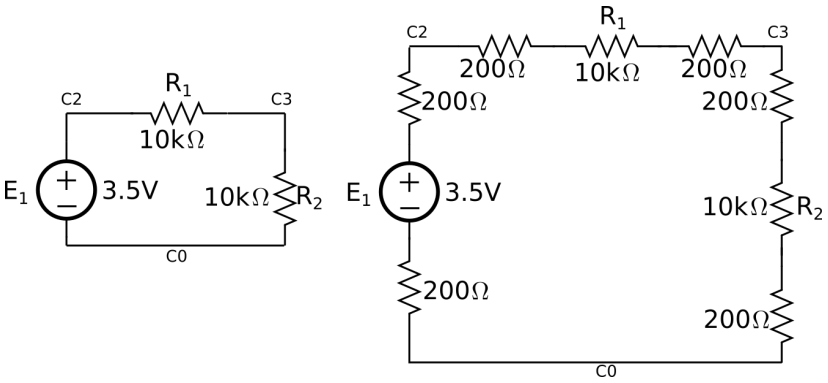
$$V_{saída\ ideal} = \frac{R2}{R1 + R2} \times V_{entrada} = \frac{10k\Omega}{10k\Omega + 10k\Omega} \times 3.5v = 1,75v$$

$$V_{saída\ real} = \frac{R2}{R1 + R2} \times V_{entrada} = \frac{10,6k\Omega}{10,6k\Omega + 10,6k\Omega} \times 3.5v = 1,75v$$

$$V_{entrada\ real} = \frac{21k\Omega}{200\Omega + 21k\Omega} \times 3.5v = 3,46v$$

Por usar resistências grandes e iguais, o resultado apresentado na figura 22 está dentro esperado. Nesse caso, a influência das resistências características foi muito menor do que no primeiro circuito, pois 200Ω é um valor pequeno em comparação com 10kΩ, então cerca de 2% de

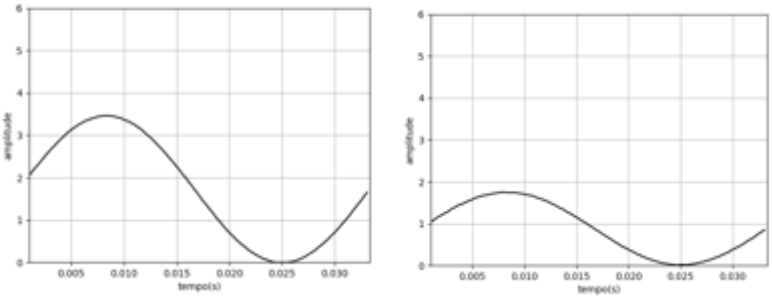
Figura 21: Circuito 2



Fonte: elaborado pelo autor.

variação já é esperada. O valor real da entrada do circuito 2 teve uma variação baixa, com erro relativo de 1.15%.

Figura 22: Teste resistivo 2



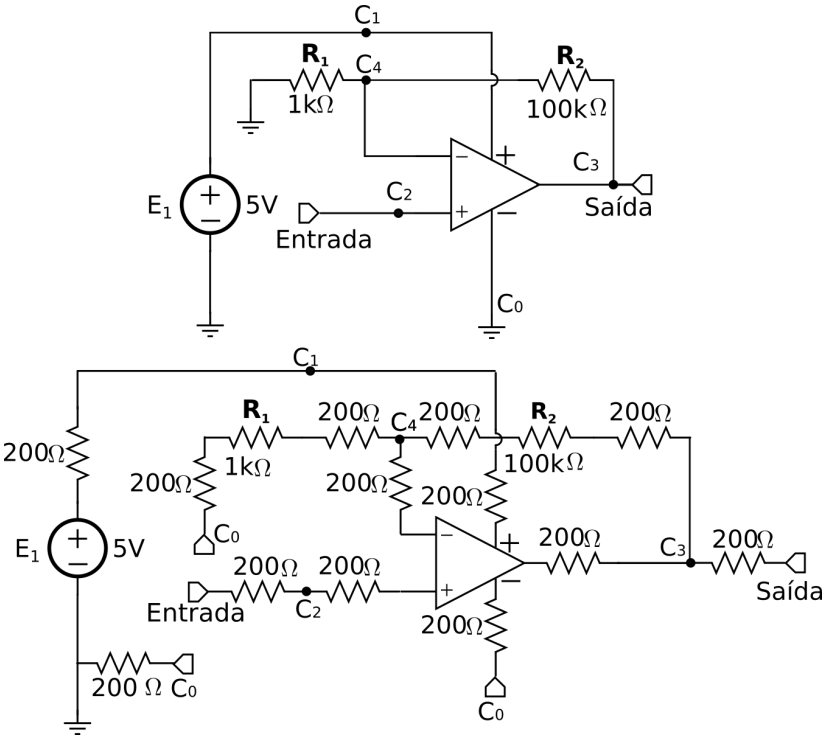
Fonte: elaborado pelo autor.

Dessa forma, quanto maior os valores dos resistores nos experimentos, ao menos para divisores resistivos simples, menor será a influência negativa das resistências características.

6.3 TERCEIRO EXPERIMENTO

Para o terceiro experimento foi montado o circuito de um amplificador na configuração não inversora, como mostra a figura 31. Os circuitos que utilizam amplificadores operacionais são muito utilizados nas disciplinas de circuitos elétricos e aquisição de sinais, disciplinas comuns nos curso tecnológicos e de engenharia.

Figura 23: Circuito 3



Fonte: elaborado pelo autor.

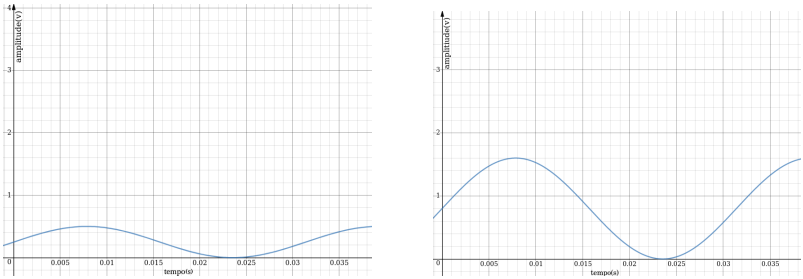
A equação abaixo mostra o valor de ganho esperado para amplificadores não inversores, usando R1 = 220Ω e R2 = 1kΩ e variação de 5% do valor nominal, considerando as resistências características de aproximadamente 200Ω. A tabela 3 mostra os resultados para essa configuração, além do ganho de um experimento real e o erro comparando o resultado real com o ideal. As Figuras de 24 até 29 mostram

os valores de entrada e saída do teste.

$$Ganho = \frac{R2}{R1} + 1 = \frac{1000 + 200 + 200}{220 + 200 + 200} + 1 = 3,2581V$$

Tabela 3: Valores de teste para configuração não inversora

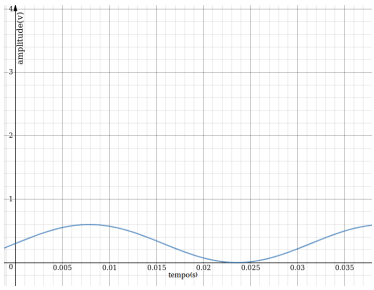
Vin(V)	Vout(V)	Ganho(V/V)	Erro
0,50	1,60	3,20	-1,78%
0,60	1,94	3,23	-0,86%
0,70	2,30	3,25	-0,25%
0,80	2,65	3,31	1,59%
0,90	3,00	3,33	2,21%
1,00	3,30	3,30	1,28%



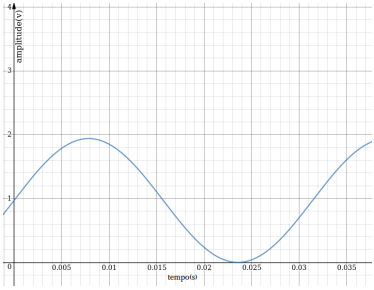
(a) Entrada 0.5V (b) Saída 1,6V

Figura 24: Teste 3 medição 1
Fonte: elaborado pelo autor.

As taxas de erros entre o valor de ganho esperado (3,2581 V/V) e os valores obtidos no experimento real foram todas inferiores a 3%. Essa taxa é considerada baixa, especialmente pois há mais fatores além da matriz de interconexão que causam desvios entre o valor de ganho calculado e o valor de ganho obtido no experimento real (limitações do amplificador operacional, tolerância dos resistores, variações na alimentação do circuito, entre outros). É importante observar que as resistências características influenciam consideravelmente os resultados, de tal forma que é importante considerá-las para a análise dos resultados esperados. Diminuir a influência das resistências características é

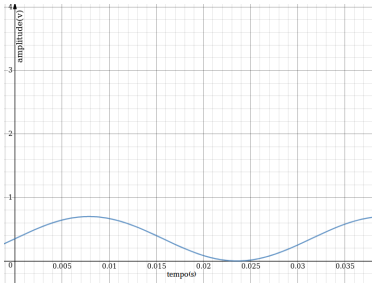


(a) Entrada 0,6V

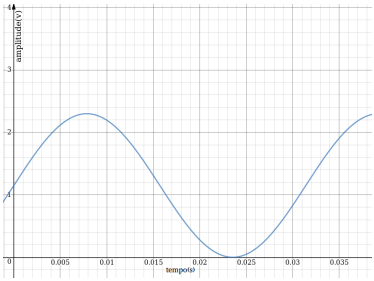


(b) Saída 1,94V

Figura 25: Teste 3 medição 2
Fonte: elaborado pelo autor.



(a) Entrada 0,7V

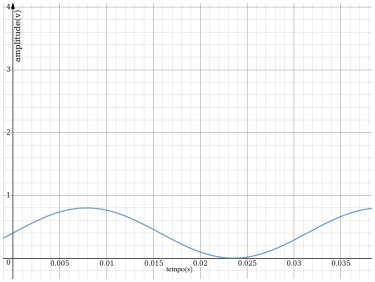


(b) Saída 2,3V

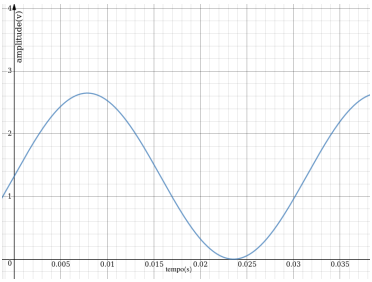
Figura 26: Teste 3 medição 3
Fonte: elaborado pelo autor.

uma característica desejável. Por outro lado, se o usuário estiver ciente destas resistências características, ele poderá fazer seus experimentos levando em conta as mesmas, como foi feito neste trabalho.

Minimizar o problema das resistências características é muito importante para que essa matriz de interconexões seja usada em laboratório remotos. No CI usado nesta implementação, MT8804, as resistências internas das conexões podem ser modificadas dependendo da tensão de alimentação do CI. Segundo as especificações do *datasheet*, a uma temperatura ambiente de 25 graus, para uma tensão de entrada de 5V, é esperada uma resistência entre as conexões de 290Ω. Para 10V, espera-se 105Ω. Já para 13V de entrada, a resistência característica entre a conexão coluna linha seria de 60Ω. Então uma estratégia

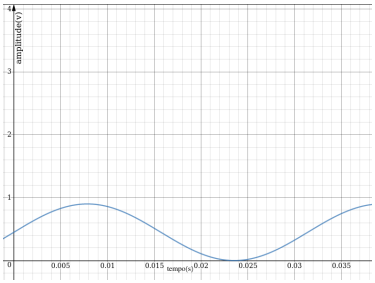


(a) Entrada 0,8V

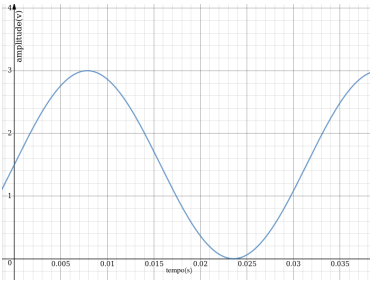


(b) Saída 2,65V

Figura 27: Teste 3 medição 4
Fonte: elaborado pelo autor.



(a) Entrada 0,9V



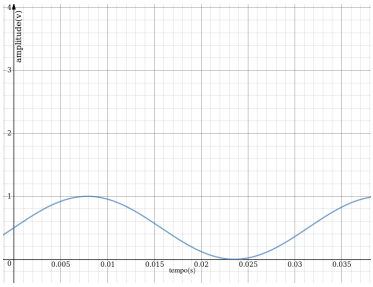
(b) Saída 3.0V

Figura 28: Teste 3 medição 5
Fonte: elaborado pelo autor.

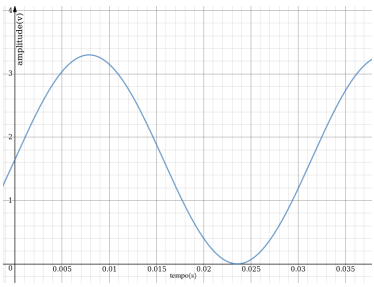
para diminuir essas resistências entre conexões é aumentar a tensão de entrada para 13V.

Outra limitação da matriz é a corrente máxima permitida em cada interconexão é de 8mA para uma tensão de entrada de 13V. Esse é uma valor baixo, e que deve ser levado em conta por quem colocar os componentes a disposição do usuário, bem como as regras que assegurem a segurança do sistema, como o impedimento de que sejam fechados curtos-circuitos pelo usuário, ou configurações que extrapolem essa limitação.

Felizmente, com a matriz de interconexões eletrônica foi possível interconectar todos os componentes usados no circuito. Também foi possível obter valores de nós específicos do circuito, por meio da



(a) Entrada 1,0V



(b) Saída 3,30V

Figura 29: Teste 3 medição 6
Fonte: elaborado pelo autor.

manipulação das conexões internas da matriz. Dessa forma, dadas as devidas precauções e considerando as limitações que existem na tecnologia utilizada nessa implementação, pode-se dizer que essa matriz de interconexões feita de multiplexadores matriciais com tecnologia MOS pode ser inserida em laboratórios remotos já existentes como o VISIR, ou em novos projetos de laboratórios remotos de eletrônica analógica.

7 CONCLUSÃO

O objetivo deste trabalho de conclusão de curso foi implementar a matriz de interconexões eletrônica com tecnologia MOS. No decorrer do processo de implementação, vários testes foram feitos, alguns protótipos foram implementados. No fim do processo, chegou-se ao sistema apresentado aqui anteriormente.

A matriz se comportou de maneira satisfatória com relação à interconexão dos componentes. Nos três experimentos que serviram de validação para o sistema, este se comportou como esperado.

Outro ponto de reflexão foi a averiguar influência negativa das impedâncias características das junções do CI matricial. Conclui-se que, para resistores de valores grandes, essas impedâncias características afetaram pouco o resultado final. Já no teste com pequenas resistências, a influência foi grande, chegando a mais de 40% de erro relativo.

Com essas considerações, é necessário que sejam feitos mais estudos a fim de diminuir a influência das impedâncias internas das junções. Quando essas influência estiver em um nível aceitável, de acordo com a metodologia escolhida, o sistema poderá ser incluído em algum laboratório remoto existente, como o VISIR. Outra alternativa é cadastrar a matriz de interconexões na plataforma RELLE (SIMãO et al., 2016), desenvolvida pelo grupo de estudos do REXLAB/UFSC, que disponibiliza um ambiente que permite a manipulação e o gerenciamento de experimentos.

REFERÊNCIAS

(ABED), A. B. de Educação a D. **Censo EAD.BR 2016**. [S.l.]: Editora do Grupo Uninter, 2017.

CHIRICO, M.; SCAPOLLA, A. M.; BAGNASCO, A. A new and open model to share laboratories on the internet. **IEEE Transactions on Instrumentation and Measurement**, v. 54, n. 3, p. 1111–1117, June 2005. ISSN 0018-9456.

DíAZ, G. et al. Remote electronics lab within a mooc: Design and preliminary results. In: **2013 2nd Experiment@ International Conference (exp.at'13)**. [S.l.: s.n.], 2013. p. 89–93.

EMEC. 2016. Last accessed 02 december 2017. Disponível em: <http://download.inep.gov.br/educacao_superior/censo_superior-/documentos/2016/notas_sobre_o_censo_da_educacao_superior_2016-.pdf>.

GARCIA-ZUBIA, J. et al. Empirical analysis of the use of the visir remote lab in teaching analog electronics. **IEEE Transactions on Education**, v. 60, n. 2, p. 149–156, May 2017. ISSN 0018-9359.

GARCIA-ZUBIA, J. et al. Empirical analysis of the use of the visir remote lab in teaching analog electronics. **IEEE Transactions on Education**, v. 60, n. 2, p. 149–156, May 2017. ISSN 0018-9359.

GARCIA-ZUBIA, J.; DIEGO, L.-d.-I.; PABLO, O. Evolving towards better architectures for remote laboratories: a practical case. v. 1, 11 2005.

GARCIA-ZUBIA, J. et al. Academic effectiveness of visir remote lab in analog electronics. 11 2011.

GUSTAVSSON, I. et al. A remote electronics laboratory for physical experiments using virtual breadboards. 01 2005.

GUSTAVSSON, I.; ZACKRISSON, J.; OLSSON, T. Traditional lab sessions in a remote laboratory for circuit analysis. 06 2018.

HERNANDEZ, U. et al. Lxi technologies for remote labs: An extension of the visir project. v. 6, 09 2010.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing In Science & Engineering**, IEEE COMPUTER SOC, 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1314 USA, v. 9, n. 3, p. 90–95, May-Jun 2007.

INEP. 2017. Last accessed 02 december 2017. Disponível em: <http://portal.inep.gov.br/sinopses-estatisticas-da-educacao-basica>.

KAZMIERKOWSKI, M. P.; LISERRE, M. Advances on remote laboratories and e-learning experiences (gomes, l. and garcia-zubia, j., eds.) [book news]. **IEEE Industrial Electronics Magazine**, v. 2, n. 2, p. 45–46, June 2008. ISSN 1932-4529.

MATTHEWS, D. The origins of distance education and its use in the united states. **THE Journal - Transforming Education Through Technology**, 1999.

NEDIC, Z.; MACHOTKA, J. Remote laboratory netlab for effective teaching of 1st year engineering students. v. 3, 08 2007.

NEDIC, Z.; MACHOTKA, J.; NAFALSKI, A. Remote laboratories versus virtual and real laboratories. In: **33rd Annual Frontiers in Education, 2003. FIE 2003.** [S.l.: s.n.], 2003. v. 1, p. T3E–1–T3E–6 Vol.1. ISSN 0190-5848.

NETLAB. 2018. Acessado em 26 de ago. de 2018. Disponível em: <http://netlab.unisa.edu.au/screenshots.xhtml>.

SEDRA, A. S.; SMITH, K. C. **Microelectronic Circuits**. fifth. [S.l.]: Oxford University Press, 2004.

SIMãO, J. S. et al. Blackout: um game para ensino de associação de resistores através da experimentação remota. In: . [S.l.: s.n.], 2016.

SOUSA, N.; ALVES, G. R.; GERICOTA, M. G. An integrated reusable remote laboratory to complement electronics teaching. **IEEE Transactions on Learning Technologies**, v. 3, n. 3, p. 265–271, July 2010. ISSN 1939-1382.

ZUTIN, D. G. et al. Lab2go — a repository to locate educational online laboratories. In: **IEEE EDUCON 2010 Conference**. [S.l.: s.n.], 2010. p. 1741–1746. ISSN 2165-9559.

ANEXO A – Códigos

Code Listing A.1: Código de validação das placas

```
#define linhas 8
#define tempo 100

const int MR = 11;
const int AE = 10;
const int DATA = 9;
const int CLK = 8;
const int CLR = 7;

const int CLR_X = 6;
const int CLK_X = 5;
const int DATA_X = 4;

const int J0 = A5;
const int J1 = A4;
const int J2 = A3;
const int J3 = A2;

void setup() {
  Serial.begin(9600);

  //pinos para configurar a placa
  pinMode(MR, OUTPUT); //MR
  pinMode(CLR, OUTPUT); //CLR
  pinMode(CLK, OUTPUT); //CLK
  pinMode(DATA, OUTPUT); //DATA

  //pinos para o shift-register externo, usado para
      colocar tens es nas 8 linhas
  pinMode(CLR_X, OUTPUT);
  pinMode(CLK_X, OUTPUT);
  pinMode(DATA_X, OUTPUT);

  //pinos para leitura das linhas da placa
  pinMode(J0,  INPUT);
  pinMode(J1,  INPUT);
  pinMode(J2, INPUT);
  pinMode(J3, INPUT);

  digitalWrite(CLR_X, LOW);
```

```
40   digitalWrite(CLR, LOW);
41   digitalWrite(MR, HIGH);
42   delay(10);
43   digitalWrite(CLR_X, HIGH);
44   digitalWrite(CLR, HIGH);
45   digitalWrite(MR, LOW);
46 }
47
48 float t;
49
50 bool problemas = false;
51 int coluna0, coluna1, coluna2, coluna3;
52
53 void loop() {
54   //coloca bit 1 na linha 0, liga a linha L0
        coluna J0 e L  a coluna J0, deve aparecer
        bit 1. Depois liga a linha L0 0   coluna J1
        e L  bit 1, depois J2 e depois J3. Repetir
55   //o processo para pr xima linha at  L7
56
57   for (int i = 0; i < linhas; i++ )
58   {
59     //limpa shift-register auxiliar, limpa shift-
          register da placa, limpa matriz
60     digitalWrite(CLR_X, LOW);
61     digitalWrite(CLR, LOW);
62     digitalWrite(MR, HIGH);
63     delay(tempo);
64     digitalWrite(CLR_X, HIGH);
65     digitalWrite(CLR, HIGH);
66     digitalWrite(MR, LOW);
67     delay(tempo);
68     coluna0 = 0;
69     coluna1 = 0;
70     coluna2 = 0;
71     coluna3 = 0;
72
73     //coloca tens o na linha i
74     //shiftOut(DATA_X, CLK_X, MSBFIRST, pow(2,i));
75     shiftOut(DATA_X, CLK_X, MSBFIRST, 255);
76
```

```
77      delay(tempo);
78
79      //d0 d1 d2 d3 a2 a1 a0 output
80      //   necessrio ligar a linha i  s  4 colunas
            de 0 a 3
81      int x = 0b11110000 | (i << 1);
82      Serial.println(x,BIN);
83      delay(tempo);
84      shiftOut(DATA, CLK, LSBFIRST, x );
85      delay(tempo);
86
87      digitalWrite(AE, HIGH);
88      delay(tempo);
89      digitalWrite(AE, LOW);
90      delay(tempo);
91
92      //fazendo a leitura da tens o em cada coluna
93      coluna0 = analogRead(J0);
94      coluna1 = analogRead(J1);
95      coluna2 = analogRead(J2);
96      coluna3 = analogRead(J3);
97      delay(tempo);
98
99      //guardar os valores das conexoes em uma matriz
            para visualiza o
100
101     Serial.print("␣valores␣linha␣");
102     Serial.print(i);
103     Serial.print("---");
104     Serial.print("␣␣coluna␣0:");
105     t = (coluna0 * 5.0) / 1024;
106     Serial.print(t, 2);
107     Serial.print("␣␣coluna␣1:");
108     t = (coluna1 * 5.0) / 1024;
109     Serial.print(t, 2);
110     Serial.print("␣␣coluna␣2:");
111     t = (coluna2 * 5.0) / 1024;
112     Serial.print(t, 2);
113     Serial.print("␣␣coluna␣3:");
114     t = (coluna3 * 5.0) / 1024;
115     Serial.print(t, 2);
```

```
116
117     Serial.println("");
118   }
119   Serial.flush();
120
121   while (true) {}
122 }
```

Code Listing A.2: Código da parte do Arduino que controla a matriz

```
1  #define placas 9
2  #define AMOSTRAS 128
3  #include <stdarg.h>
4  #include <Wire.h>
5  #include <Adafruit_MCP4725.h>
6  Adafruit_MCP4725 dac; // constructor
7
8  const PROGMEM uint16_t DACLookup_FullSine_7Bit[128]
       =
9  {
10   2048, 2148, 2248, 2348, 2447, 2545, 2642, 2737,
11   2831, 2923, 3013, 3100, 3185, 3267, 3346, 3423,
12   3495, 3565, 3630, 3692, 3750, 3804, 3853, 3898,
13   3939, 3975, 4007, 4034, 4056, 4073, 4085, 4093,
14   4095, 4093, 4085, 4073, 4056, 4034, 4007, 3975,
15   3939, 3898, 3853, 3804, 3750, 3692, 3630, 3565,
16   3495, 3423, 3346, 3267, 3185, 3100, 3013, 2923,
17   2831, 2737, 2642, 2545, 2447, 2348, 2248, 2148,
18   2048, 1947, 1847, 1747, 1648, 1550, 1453, 1358,
19   1264, 1172, 1082,  995,  910,  828,  749,  672,
20    600,  530,  465,  403,  345,  291,  242,  197,
21    156,  120,   88,   61,   39,   22,   10,    2,
22      0,    2,   10,   22,   39,   61,   88,  120,
23    156,  197,  242,  291,  345,  403,  465,  530,
24    600,  672,  749,  828,  910,  995, 1082, 1172,
25   1264, 1358, 1453, 1550, 1648, 1747, 1847, 1947
26 };
27
28 //VARI VEIS DA PLACA
29 const int MR = 12;
30 const int AE = 11;
31 const int datapin = 10;//pin1 para placa 1
```

```
32 const int clockpin = 9;
33 const int CLR = 8;
34
35 void setup() {
36   Serial.begin(9600);
37   pinMode(clockpin, OUTPUT);
38   pinMode(datapin, OUTPUT);
39   pinMode(MR, OUTPUT);
40   pinMode(CLR, OUTPUT);
41   pinMode(AE, OUTPUT);
42   pinMode(A1, INPUT);
43
44   //comandos de inicializa o do sistema
45   digitalWrite(AE, LOW);
46   digitalWrite(CLR, LOW);
47   digitalWrite(MR, HIGH);
48   delay(100);
49   digitalWrite(CLR, HIGH);
50   digitalWrite(MR, LOW);
51
52   dac.begin(0x62);
53 }
54
55 int DADO[AMOSTRAS];
56
57 boolean osciloscopio = false;
58 boolean serial_received = false;
59 boolean dac_received = false;
60 boolean adc_send = false;
61 boolean enable = false;
62 boolean seno_received = false;
63
64 byte buffer_dac_in[4];
65 byte inByte = 0;
66
67 union u_tag {
68   byte b[4];
69   float fval;
70 } u;
71
72 float dac_in;
```

```
73 uint16_t dac_value;
74 byte row_value = 0;
75 int adc_value=0;
76
77 int i;
78
79 void loop() {
80   Serial.flush();
81   if (Serial.available() > 0)
82   {
83     while (Serial.available() > 0)
84     {
85       inByte = Serial.read();
86       if (inByte == 255)
87       {
88         row_value = Serial.read();
89         //Serial.println(row_value, DEC);
90         serial_received = true;
91       }
92       if (inByte == 65) //0x41 enable
93       {
94         enable = true;
95       }
96       if (inByte == 66) //0x42 master reset
97       {
98         digitalWrite(MR, HIGH);
99         delay(100);
100        digitalWrite(MR, LOW);
101      }
102      if (inByte == 67) //0x43 acionamento dac
103      {
104        u.b[0] = Serial.read();
105        u.b[1] = Serial.read();
106        u.b[2] = Serial.read();
107        u.b[3] = Serial.read();
108
109        dac_received = true;
110      }
111      if (inByte == 68)//0x44 acionamento onda
             senoidal
112      {
```

```
113         u.b[0] = Serial.read();
114         u.b[1] = Serial.read();
115         u.b[2] = Serial.read();
116         u.b[3] = Serial.read();
117         seno_received = true;
118       }
119       if (inByte == 69) //0x45 leitura adc
120       {
121         adc_send = true;
122       }
123       if (inByte == 70) //0x46  osciloscopio
124       {
125         osciloscopio = true;
126       }
127     }
128   }
129
130   if (serial_received)
131   {
132     shiftOut(datapin, clockpin, LSBFIRST, row_value
          );
133     serial_received = false;
134   }
135
136   if (enable)
137   {
138     digitalWrite(AE, HIGH);
139     delay(5);
140     digitalWrite(AE, LOW);
141     delay(5);
142     enable = false;
143   }
144
145   if (dac_received)
146   {
147     dac_value = u.fval * 4095 / 5.0;
148     dac.setVoltage(dac_value, false);
149     dac_received = false;
150   }
151   if(adc_send)
152   {
```

```
153     adc_value = analogRead(A1);
154     Serial.println(adc_value);
155     adc_send = false;
156   }
157
158   uint16_t var_seno;
159   if (seno_received)
160   {
161     for (i = 0; i < AMOSTRAS; i++)
162     {
163       dac.setVoltage(u.fval * pgm_read_word(&(
              DACLookup_FullSine_7Bit[i])) / 5, false);
164       adc_value = analogRead(A1);
165       DADO[i] = adc_value;
166       delayMicroseconds(148);
167     }
168     for (i = 0; i < AMOSTRAS; i++)
169     {
170       Serial.println(DADO[i]);
171     }
172     seno_received = false;
173   }
174
175   if (osciloscopio)
176   {
177     for (i = 0; i < AMOSTRAS; i++)
178     {
179       adc_value = analogRead(A1);
180       DADO[i] = adc_value;
181       delayMicroseconds(148);
182     }
183     for (i = 0; i < AMOSTRAS; i++)
184     {
185       Serial.println(DADO[i]);
186     }
187     osciloscopio = false;
188   }
189
190   Serial.flush();
191   delay(20);
192 }
```

Código em Python adaptado. Foi usado no computador local.

```
import serial
import time
import struct
import csv
import matplotlib.pyplot as plt
from numpy import arange

ser = serial.Serial(port='/dev/ttyACM0',baudrate=9600,bytesize=
                                serial.EIGHTBITS,parity=serial.
                                PARITY_NONE,timeout=2)
try:
    ser.isOpen()
    time.sleep(2)
    print("Conex o serial aberta")
except:
    print("Erro")
    exit()

LINHAS = 8
COLUNAS = 4
LINHAS_MATRIZ = 24
COLUNAS_MATRIZ = 12
PLACAS = 9
ORDEM = 3

# cria array de dados de configuracao da matriz
def LimparMatriz():
    mensagem = [[0 for i in range(PLACAS)] for j in
                range(LINHAS)]
    mensagem = [[b'\x00',b'\x00',b'\x00',b'\x00',b'\x00',b'\x00'
                                    ,b'\x00',b'\x00',b'\x00'],
                [b'\x02',b'\x02',b'\x02',b'\x02',b'\x02',b'\x02'
                                                ,b'\x02',b'\x02'
                                                ,b'\x02'],
                [b'\x04',b'\x04',b'\x04',b'\x04',b'\x04',b'\x04'
                                                ,b'\x04',b'\x04'
                                                , b'\x04'],
                [b'\x06',b'\x06',b'\x06',b'\x06',b'\x06',b'\x06'
                                                ,b'\x06',b'\x06'
                                                ,b'\x06'],
                [b'\x08',b'\x08',b'\x08',b'\x08',b'\x08',b'\x08'
                                                ,b'\x08',b'\x08'
                                                ,b'\x08'],
                [b'\x0a',b'\x0a',b'\x0a',b'\x0a',b'\x0a',b'\x0a'
                                                ,b'\x0a',b'\x0a'
                                                ,b'\x0a'],
                [b'\x0c',b'\x0c',b'\x0c',b'\x0c',b'\x0c',b'\x0c'
                                                ,b'\x0c',b'\x0c'
                                                ,b'\x0c'],
                [b'\x0e',b'\x0e',b'\x0e',b'\x0e',b'\x0e',b'\x0e'
```

```
                                                    ,b'\x0e',b'\x0e'
                                                    ,b'\x0e'] ]
    return mensagem

def LimparMatrizNaPlaca():
    ser.write(b'\x42')
    return

def ConfigurarMatriz(mensagem):
    componentes = MostrarComponentes()
    # o vetor coluna[] recebe 24 valores para as 24 linhas da
                                    matriz desse sistema
    print('DIGITE A CONFIGURA  O DA MATRIZ (24 linhas por 12
                                    colunas)')
    for i in range (LINHAS_MATRIZ):
        # de 0 a 11, tem 12 colunas
        coluna_aux=input(componentes[i]+"COLUNA:")
        if(coluna_aux==''):
            coluna = 99
        else:
            coluna = int(coluna_aux)

        if((coluna>-1) and (coluna<12)):
            pilha = int(i/LINHAS)
            print('pilha: ' + repr(pilha))

            # descobrir a linha da placa
            linha_da_placa = i % LINHAS
            print("linha da placa: " + repr(linha_da_placa))

            # descobrir a coluna da placa
            coluna_da_placa = (int(coluna)%COLUNAS)
            print("coluna da placa: ", coluna_da_placa)

            coluna_de_placas = coluna/4
            # descobrir o n   da placa(de 0 a 8)
            n_placa = int(pilha * ORDEM + coluna_de_placas)
            if(pilha==1):
                n_placa = (n_placa-8)*-1
            print("n mero placa :", n_placa)

            mb = int.from_bytes(mensagem[linha_da_placa][n_placa
                                    ],byteorder='big',
                                    signed=False)
            mb |= (1 << (7-coluna_da_placa))
            mensagem[linha_da_placa][n_placa]=mb.to_bytes(1,
                                    byteorder='big',
                                    signed=False)
            print(">> " + "{0:b}".format(mb))
            print(">> ""{0:x}".format(mb))
    print(mensagem)
    return mensagem
```

```
def MandarDados(mensagem):
    tempo = 0.01
    for i in range (LINHAS):
        for j in range (PLACAS-1,-1,-1):
            print("linha ",i, "placa ", j)
            time.sleep(tempo)
            ser.write(b'\xff')
            ser.write(mensagem[i][j])
            print(mensagem[i][j])
            time.sleep(tempo)
        ser.write(b'\x41')
        time.sleep(tempo)
    return mensagem

def MudarTensao():
    dac_value = float(input("Tensao(0-5V): "))
    dac_value_to_send = bytearray(struct.pack("f",dac_value))
    dac_value_to_send.insert(0,67)
    ser.write(dac_value_to_send)
    return

def seno():
    dac_value =
    float(input("Valor da seno(0-5V):"))
    dac_value_to_send = bytearray(struct.pack("f", dac_value))
    dac_value_to_send.insert(0,68)
    ser.write(dac_value_to_send)

    # parte do osciloscopio
    dado = []
    for i in range (0, AMOSTRAS):
        VALOR_SERIAL = int(ser.readline())
        aux_val = round((VALOR_SERIAL*5/1024),2)
        dado.append(aux_val)

    # defini  o de parametros
    n_ondas = 2 #numero medicoes capturadas
    n = n_ondas*64 # 64 dados para cada onda
    T = n_ondas*1.0/60 # periodo
    dt = T/n # intervalo entre cada medida
    t = []
    t = dt*arange(0, n) # gera vetor com instantes de tempo

    ARQUIVO = open('dado.csv', "w")
    writer = csv.writer(ARQUIVO, delimiter='\t')

    for i in range(0, AMOSTRAS):
        writer.writerow([dado[i]])
        print(dado[i])
    ARQUIVO.close()
```

```
    # Tra ado de graficos
    # Forma de onda
    plt.subplot(1, 1, 1)
    plt.xlim(0.001, T) # define limites do eixo x
    plt.ylim(0, 6) # define limites do eixo y
    plt.plot(t, dado,'k-')
    plt.xlabel('tempo(s)')
    plt.ylabel('amplitude')
    plt.grid()
    plt.show()
    return

def LerEntradaAnalogica():
    ser.write(b'\x45')
    # time.sleep(0.01)
    value = ser.readline()
    print(int(value)*5/1023)
    return

AMOSTRAS = 128
def osciloscopio():
    ser.write(b'\x46')
    dado = []
    for i in range (0, AMOSTRAS):
        VALOR_SERIAL = int(ser.readline())
        aux_val = round((VALOR_SERIAL*5/1024),2)
        dado.append(aux_val)

    # definicao de parametros
    n_ondas = 2 #medidas capturadas
    n = n_ondas*64 # 64 dados para cada onda
    T = n_ondas*1.0/60 # periodo
    dt = T/n # i
    t = []
    t = dt*arange(0, n)


    ARQUIVO = open('dado.csv', "w")
    writer = csv.writer(ARQUIVO, delimiter='\t')

    for i in range(0, AMOSTRAS):
        writer.writerow([dado[i]])
        print(dado[i])
    ARQUIVO.close()

    # Tra ado de graficos
    plt.subplot(1, 1, 1)
    plt.xlim(0.001, T) # define limites do eixo x
    plt.ylim(0, 6) # define limites do eixo y
    plt.plot(t, dado,'k-')
    plt.xlabel('tempo(s)')
    plt.ylabel('amplitude')
```

```
    plt.show()

    return

def MostrarComponentes():
    componentes = [
    "linha 0 - gnd",
    "linha 1 - vcc",
    "linha 2 - tens o vari vel 0-5V",
    "linha 3 - entrada anal gica",
    "linha 4 - 1k(pin1)",
    "linha 5 - 1k(pin2)",
    "linha 6 - 220(pin1)",
    "linha 7 - 220(pin2)",
    "linha 8 - lm324-ampop(gnd)",
    "linha 9 - lm324-ampop(2out)",
    "linha 10 - lm324-ampop(2in-)",
    "linha 11 - lm324-ampop(2in+)",
    "linha 12 - lm324-ampop(Vcc)",
    "linha 13 - lm324-ampop(1in+)",
    "linha 14 - lm324-ampop(1in-)",
    "linha 15 - lm324-ampop(1out)",
    "linha 16 - 220(pin1)",
    "linha 17 - 220(pin2)",
    "linha 18 - 1k(pin1)",
    "linha 19 - 1k(pin2)",
    "linha 20 - BC548 - coletor",
    "linha 21 - BC548 - base",
    "linha 22 - BC548 - emissor",
    "linha 23 -n o conectada"
    ]
    return componentes

def SAIR():
    ser.close()
    exit()
    return

while 1:
    val = input("0 - Limpa Matriz F sica\n
    1 - Limpar Vari vel da Matriz\n
    2 - Configurar Matriz\n
    3 - Mandar Dados\n
    4 - Mudar Tens o\n
    5 - seno\n     6 - Mostrar Componentes\n
    7 - Ler Entrada Anal gica \n
    8 - osciloscpio\n9 - SAIR\nEscolha: ")
    if ( val == '0' ): LimparMatrizNaPlaca()
    if ( val == '1' ): m = LimparMatriz()
    if ( val == '2' ): m = ConfigurarMatriz(m)
    if ( val == '3' ): m = MandarDados(m)
    if ( val == '4' ): MudarTensao()
```

```
    if ( val == '5' ): seno()
    if ( val == '6' ):
        componentes = MostrarComponentes()
        for i in range (LINHAS_MATRIZ):
            print(componentes[i])
    if ( val == '7' ): LerEntradaAnalogica()
    if ( val == '8' ): osciloscopio()
    if ( val == '9' ): SAIR()
```

ANEXO B – PCB

Figura 30: PBC trilhas

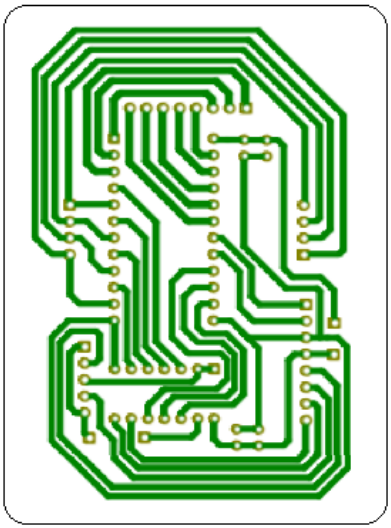


Figura 31: PBC com componentes