INSTITUTO FEDERAL DE SANTA CATARINA
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

KAEL RICARDO KILL

ESTUDO DO LABORATÓRIO REMOTO VISIR PARA A IMPLEMENTAÇÃO DE NOVOS COMPONENTES, INSTRUMENTOS E FUNCIONALIDADES

FLORIANÓPOLIS, JULHO DE 2019

INSTITUTO FEDERAL DE SANTA CATARINA
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

KAEL RICARDO KILL

ESTUDO DO LABORATÓRIO REMOTO VISIR PARA A IMPLEMENTAÇÃO DE NOVOS COMPONENTES, INSTRUMENTOS E FUNCIONALIDADES

Trabalho de conclusão de curso submetido ao Instituto Federal de Santa Catarina como parte dos requisitos para obtenção do título de tecnólogo em eletrônica industrial

Orientador: Prof. Dr. Luis Carlos Martinhago Schlichting
Coorientador: Daniel Dezan De Bona

FLORIANÓPOLIS, JULHO DE 2019

Kill, Kael Ricardo
Estudo do laboratório remoto VISIR para a implementação de novos componentes, instrumentos e funcionalidades [TCC] / Kael Ricardo Kill; orientador, Luis Carlos Martinhago Schlichting; coorientador, Daniel Dezan de Bona – Florianópolis, SC, 2019.
82 p.:il. color.

Trabalho de Conclusão de Curso (Eletrônica Industrial) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Inclui referências.

1. VISIR. 2.Laboratórios remotos 3. Ensino a distância I. Schlichting, Luis Carlos Martinhago. Bona, Daniel Dezan de. II. Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. III. Revisão e adição do elemento diodo e de um novo multímetro ao laboratório remoto VISIR

ESTUDO DO VISIR PARA IMPLEMENTAÇÃO DE NOVOS COMPONENTES, INSTRUMENTOS E FUNCIONALIDADES

KAEL RICARDO KILL

Este trabalho foi julgado adequado para obtenção do Título de Tecnólogo em Eletrônica Industrial e aprovado na sua forma final pela banca examinadora do Curso Superior de Tecnologia em Eletrônica Industrial do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 8 de julho, 2019

Banca examinadora:

Luis Carlos Martinhago Schlichting, Dr. Eng

Daniel Dezan de Bona, M. Eng

Reginaldo Steinbach, M. Eng

Fernando Santana Pacheco, Dr. Eng

Dedico este trabalho a minha
amiga e companheira,
Greyce Estevão,
grande incentivadora e parceira
de todas as horas.

AGRADECIMENTOS

Agradeço a minha família e amigos. Especialmente aos meus pais, Sérgio e Clarinda, minha amiga e companheira, Greyce, e aos pais dela, Amauri e Roseli, por todo o suporte e apoio.

Aos meus orientadores, Schlichting e Dezan, pelo suporte oferecido durante toda minha jornada acadêmica e pela oportunidade e apoio para o desenvolvimento deste trabalho.

Ao IFSC e todo pessoal envolvido, especialmente ao DAELN, por contribuír para a realização deste trabalho.

RESUMO

Este trabalho tem como proposta relatar os conhecimentos obtidos com a revisão do laboratório remoto VISIR, o desenvolvimento de uma nova interface para o multímetro e a inserção do componente diodo. O texto é dividido em três grandes blocos. No primeiro é apresentada uma breve revisão bibliográfica sobre educação a distância, laboratórios remotos e suas tecnologias. Em seguida, é feita uma revisão sobre o laboratório VISIR, descrevendo seu funcionamento e características. Neste bloco é apresentado o hardware do sistema, faz-se a descrição do funcionamento dos diferentes servidores, a relação entre eles e como os dados são transferidos entre cada etapa de uma sessão no laboratório. No terceiro e último bloco são descritos os procedimentos para adicionar o componente diodo e um novo multímetro ao sistema, relatando as alterações necessárias para a inserção de novas funcionalidades às ferramentas disponíveis no sistema. O trabalho atinge os objetivos propostos quando, no final, o diodo e o multímetro desenvolvidos passam a ser parte integrante do laboratório instalado no IFSC e o material gerado serve como referência para o desenvolvimento de novos dispositivos e ferramentas.

Palavras-chave: VISIR. Laboratórios remotos. Educação a distância,

ABSTRACT

This paper intends to report the knowledge obtained by reviewing the VISIR remote lab, the development of a new interface for the multimeter and the insertion of the diode component. The text is divided into three main blocks. The first one presents a brief bibliographic review on distance education, remote laboratories and their technologies. Then, a review about the VISIR lab, describing its operation and characteristics. In this block the system hardware and the description of the operation of the different servers are explained and the relationship between them and how the data is transferred between each stage of a session in the laboratory. The third and final block describes the procedures for adding the diode component and a new multimeter to the system. Reporting the necessary changes for the insertion of new features to the tools available in the system. The presented work achieves the proposed objectives when, in the end, the developed diode and multimeter become an integral part of the IFSC's VISIR and the written material serves as a reference for the development of new devices and tools.

Keywords: VISIR. Remote Labs. Distance Learning

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACK - Acknowledgment

BTH - Blekinge Tekniska Högskola

CSS - Cascading Style Sheets

CI - Circuito Integrado

DAELN - Departamento Acadêmico de Eletrônica

EaD - Ensino a Distância

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

HTTPS - Hyper Text Transfer Protocol Secure

I2C - Inter-Integrated Circuit

IFSC - Instituto Federal de Santa Catarina

IP - Internet Protocol

LabView - Laboratory Virtual Instrument Engineering Workbench

PCI - Peripheral Component Interconnect

PHP - PHP: Hypertext Preprocessor

PXI - PCI eXtensions for Instrumentation

SYN - Synchronisation

TCP - Transmission Control Protocol

USB - Universal Serial Bus

VI - Virtual Instrument

VISA - Virtual Instrument Software Architecture

VISIR - Virtual Instrument Systems in Reality

VISIR + - VISIR Plus Project

W3C - World Wide Web Consortium

XML - eXtensive Markup Language

SUMÁRIO

1 INTRODUÇÃO

No primeiro semestre do ano de 2017, foi implantado no IFSC campus Florianópolis o laboratório remoto *VISIR*. O projeto, uma parceria entre várias instituições de ensino, trazia uma nova ferramenta de ensino para a comunidade acadêmica. Embora o laboratório instalado fosse perfeitamente funcional, o pessoal técnico envolvido no projeto não detinha o conhecimento sobre as tecnologias utilizadas para compor a plataforma. Isso trouxe a oportunidade de iniciar um projeto de estudo e análise do funcionamento do laboratório partindo, praticamente, do zero. Assim surgiu a oportunidade para a realização deste trabalho de conclusão de curso.

1.1 Definição do problema

Com a crescente oferta e demanda de cursos à distância e a possibilidade de reservar até 40% da carga horária de cursos presenciais com Ensino a Distância (EaD) (BRASIL, 2018), é necessário que se desenvolvam tecnologias que garantam a qualidade do ensino e o bom aproveitamento por parte do aluno.

Em cursos de áreas técnicas, como cursos de tecnologia e engenharia, a utilização de laboratórios de ensino é indispensável para a formação do aluno. É no laboratório que o aluno colocará em prática o conhecimento adquirido em sala de aula e terá contato com materiais e instrumentos utilizados na sua área de formação. Essa necessidade de atividades práticas prejudicam a oferta de EaD nestes cursos. Nestes casos, a utilização de laboratórios remotos pode ser uma solução.

Embora seja uma tecnologia conhecida em algumas universidades, o uso de laboratórios remotos como ferramenta pedagógica ainda não é difundido entre professores e alunos das instituições de ensino brasileiras.

A participação do Departamento Acadêmico de Eletrônica (DAELN) e do Instituto Federal de Santa Catarina (IFSC) no projeto *VISIR+* trouxe a oportunidade e necessidade de estudar e conhecer o projeto *Virtual Instrument Systems in Reality* (*VISIR*) e desenvolver *software* e *hardware*, bem como material didático. Assim surgiu a seguinte questão: compreender o sistema tornaria possível o desenvolvimento de novos instrumentos, componentes ou funcionalidades para o laboratório?

1.2 Justificativa

A utilização de laboratórios remotos pode vir a complementar as aulas de EaD, especialmente em cursos tecnológicos, por expor o aluno a um ambiente similar ao que seria encontrado em um laboratório real.

Dominar a tecnologia de laboratórios remotos permitiria aos professores do DAELN otimizar o tempo em sala de aula e programar atividades que possam ser executadas a distância, permitindo que mais alunos tenham acesso aos laboratórios e instrumentos disponíveis.

Os alunos podem executar experimentos e praticar o uso dos instrumentos, de qualquer lugar e horário, de maneira muito semelhante à prática em um laboratório real sem riscos de acidentes ou de danificar equipamentos.

Este trabalho visa estudar e compreender a tecnologia de laboratórios remotos, especificamente o *VISIR*, e propor melhorias e alterações em seu *software* e *hardware* de modo a deixá-lo adequado ao uso no DAELN.

1.3 OBJETIVOS

O trabalho descrito pretende fazer uma revisão bibliográfica sobre laboratórios remotos, especialmente o *VISIR*, descrevendo seu funcionamento num âmbito geral. Se propõe, também, a fazer alterações no sistema, adicionando novas funcionalidades e instrumentos ao laboratório.

1.3.1 Objetivo geral

Estudar e descrever o funcionamento do *hardware* e *software* que compõem o sistema *VISIR* para adicionar novos elementos ao sistema, como componentes e instrumentos.

1.3.2 Objetivos específicos

Para se atingir o objetivo geral, o trabalho foi dividido em algumas etapas:

a) revisar bibliografia sobre laboratórios remotos;
b) estudar e descrever o *hardware* e *software* do *VISIR*;
   - compreender o funcionamento do equipamento instalado;
   - analisar os códigos-fonte dos diversos servidores e entender o funcionamento e a relação entre eles;

- relatar os resultados obtidos a fim de gerar documentação sobre o equipamento.

c) Adicionar o componente diodo;

d) Adicionar o multímetro digital Minipa 2042D.

2 DESENVOLVIMENTO

O desenvolvimento deste trabalho foi dividido em três etapas principais. O primeiro bloco pretende apresentar uma breve revisão sobre ensino à distância e laboratórios remotos e, mais especificamente, o *VISIR*. Em seguida, são apresentadas algumas das tecnologias utilizadas no *VISIR* que formam a base técnica e teórica para o desenvolvimento do trabalho proposto. Por fim, a terceira parte descreve os passos para a inserção de um novo componente e um instrumento no sistema.

2.1 Ensino a Distância

Em dezembro de 1996, o Brasil deu os primeiros passos no ensino a distância por meio da Lei de Diretrizes e Bases da Educação Nacional – Lei nº. 9.394/1996, posteriormente regulamentado pelo Decreto nº 5.622/2005, onde era incentivado o desenvolvimento de programas de ensino a distância em todos os níveis e modalidades de ensino. A Portaria nº 1.428 (BRASIL, 2018), passou a permitir que as instituições de ensino superior incluíssem em seu currículo disciplinas na modalidade semipresencial, desde que a oferta não ultrapassasse 40% da carga horária total do curso.

De acordo com o Censo EAD Brasil (ABED, 2016), a educação a distância mobiliza, atualmente, cerca de 5.048.912 alunos contabilizando cursos totalmente a distância, semipresenciais, corporativos e não corporativos de variados níveis acadêmicos, tipos de curso e áreas de conhecimento. Este número representa um aumento de 30% em relação ao ano 2014.

Alguns dos fatores que contribuíram para a popularização de cursos a distância foram o avanço da internet banda larga e de recursos de informação e comunicação e também um aumento na demanda por ensino superior, além de alguns benefícios oferecidos pela modalidade, como horários flexíveis e possibilidade de estudar em qualquer lugar (UNIVERSIA, 2016).

Alguns dos recursos utilizados para apresentação do conteúdo dos cursos apontados pelo Censo são: textos digitais, áudios, vídeos e tele aulas, jogos eletrônicos e simulações (ABED, 2016).

Apesar dos esforços e investimentos aplicados, alguns cursos, especialmente da área técnica, que dependem de laboratórios e experimentos

práticos, enfrentam dificuldades técnicas para se adaptarem ao ensino a distância. Estes cursos devem buscar soluções que permitam a formação dos alunos com os mesmos resultados de cursos presenciais. A maior dificuldade é substituir os laboratórios por um modelo que ofereça a mesma qualidade de aprendizagem (SCHLICHTING, 2016). Uma solução para esta dificuldade já existe e vem sendo desenvolvida em alguns países nos últimos anos. Esta possível solução são os laboratórios remotos.

2.2 Laboratórios remotos

Um laboratório remoto é um aparato físico conectado a instrumentos controlados por computador que podem ser acessados remotamente via internet. Os experimentos realizados em um laboratório remoto são uma experiência real, diferentemente de laboratórios virtuais, onde os testes e resultados são obtidos através de um modelo físico-matemático do experimento realizado (ALVES et al., 2016).

Segundo Schlichting (2016), uma característica importante de laboratórios remotos é que os experimentos podem ser executados a qualquer tempo e de qualquer lugar, sem a presença de um professor ou de pessoal técnico. Essa característica possibilita que o tempo em sala de aula seja mais aproveitado para o diálogo e discussões teóricas ao invés de se usar toda uma aula organizando e executando uma atividade prática. Um laboratório remoto permite, ainda, que o aluno tenha acesso aos instrumentos utilizados em um laboratório de ensino fora do período de aula, o que permite que este realize suas atividades no horário que lhe for mais conveniente e otimiza a ocupação de um laboratório real, possibilitando que mais alunos tenham acesso ao ambiente físico.

O uso de laboratórios remotos como ferramenta educacional vem sendo aplicado em diversas áreas do conhecimento como física, biologia, mecânica, elétrica e eletrônica. Alguns exemplos de laboratórios que podem ser citados são: Painel Elétrico CA, Painel Elétrico CC (RELLE, 2019), Microscópio Remoto, *darchmedes-demo* (WEBLAB, 2019) e o *VISIR*.

Os Painéis Elétricos CA e CC são dois laboratórios diferentes que permitem que o usuário estude as associações em série, paralela e mista em rede elétrica CA e CC, respectivamente.

O Microscópio Remoto permite ao usuário analisar a pigmentação presente em diferentes amostras de folhas.

O *darchmedes-demo* permite observar o princípio de Arquimedes usando diferentes corpos.

O *VISIR* permite que o usuário construa circuitos eletrônicos e faça medições usando componentes e instrumentos comuns em bancadas de eletrônica.

Embora cada laboratório apresente um experimento específico e tecnologias de implementação variadas, uma caracteristica em comum entre os laboratórios que estão disponíveis através da internet é o uso de ferramentas básicas para a construção de suas interfaces, como linguagens de programação e marcação. A seguir, será feita uma revisão das linguagens *web* utilizadas neste trabalho: *HTML*, *XML*, *CSS* e *JavaScript*. Serão revisadas também algumas tecnologias usadas especificamente no *VISIR*, como *LabView*, *HTTP* e *TCP/IP*.

2.2.1 HTML

A Linguagem de Marcação de Hipertexto ou *HyperText Markup Language* (*HTML*) é o componente mais básico na construção de páginas *web*. Através dela são definidas a estrutura e o conteúdo das páginas.

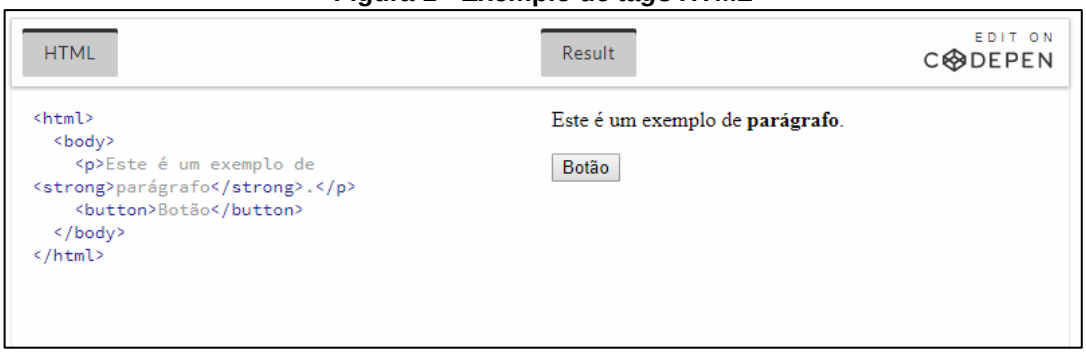
Teve seu desenvolvimento iniciado por Tim Berners-Lee no ano de 1989 e desde então passou por diversas modificações e versões diferentes, sendo mantidas pelo *World Wide Web Consortium* (*W3C*). A versão da linguagem utilizada atualmente é o *HTML5* (*MDN*, 2019a).

O *HTML* é uma linguagem de marcação e é usada para descrever como serão exibidas as páginas *web*. Os elementos básicos para escrever um documento ou arquivo *HTML* são os *elementos* e *textos*.

Um elemento é iniciado por um marcador – ou *tag* – de abertura <*tag*> e um marcador de fechamento <*/tag*>. Para denotar um parágrafo, por exemplo, inicia-se com a *tag* de abertura <*p*> seguida pelo texto do parágrafo e se encerra com <*/p*>. Existem elementos que podem ter um de seus marcadores omitidos e diversos elementos podem ser aninhados, criando assim uma hierarquia ou estrutura para o documento (*W3C*, 2017). A Figura 1 mostra um fragmento de código *HTML* e a interpretação do arquivo por um navegador *web*.

Na Figura 1 é possível observar que nem todo elemento é convertido em um componente visível na página: as *tags* <*html*> e <*body*> servem para estruturar o texto e não tem seu conteúdo exibido diretamente. Todo conteúdo dentro de <*html*> e </*html*> são interpretados como um arquivo *HTML*. Os elementos aninhados em <*body*> e </*body*> são considerados o corpo ou conteúdo da página. A *tag* <*p*> denota o parágrafo e o texto entre as <*strong*> e </*strong*> é exibido em negrito. A *tag* <*button*> cria um botão clicável.

Figura 1 - Exemplo de *tags HTML*



Fonte: Elaboração própria (2019).

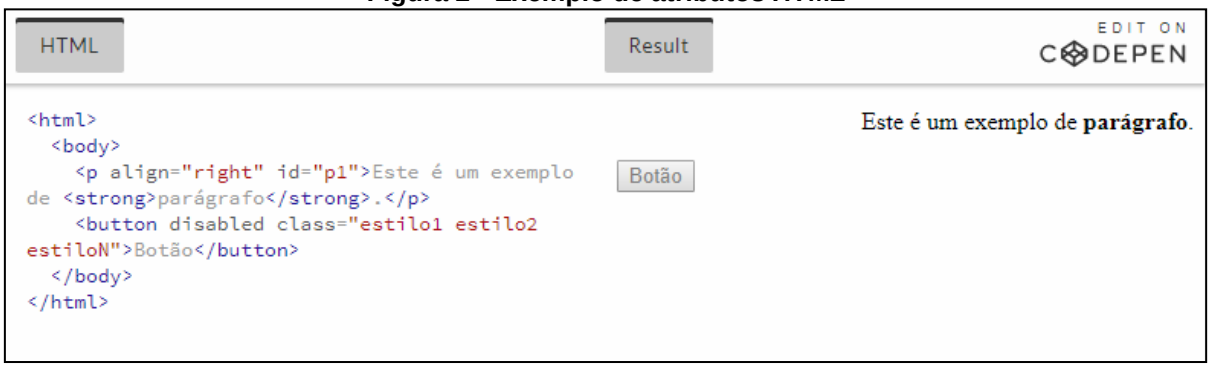
Elementos podem apresentar atributos, que alteram seu comportamento. Os atributos devem ser colocados no marcador de abertura e apresentam um nome e um valor, sendo escritos da seguinte forma:

<tag nome='valor'>[...]</tag>

Alguns atributos podem ter seu valor omitido, em tais casos pode-se usar valor vazio (“”) ou omitir o símbolo de igualdade e valor (=””).

A Figura 2 ilustra o fragmento mostrado na Figura 1 com alguns atributos editados.

Figura 2 - Exemplo de atributos *HTML*



Fonte: Elaboração própria (2019).

No elemento <*p*> foi adicionado o atributo *align=”right”*, que justifica o texto do parágrafo a direita e no elemento <*button*> foi adicionado o atributo *disabled* – com omissão do valor – que desabilitou o botão na página.

Dois atributos que devem ser citados são o *id* e o *class*. O atributo *class* define nomes de classes para os elementos e são comumente usados para definir o estilo dos elementos e o atributo *id* define um identificador único para o elemento. Um mesmo elemento só pode ter um *id* mas pode apresentar quantos *class* forem necessários.

2.2.2 CSS

As Folhas de Estilo em Cascata ou *Cascading Style Sheets* (*CSS*) são usadas para definir o estilo gráfico de páginas *HTML*. Aplicando CSS ao *HTML* pode-se, por exemplo, selecionar as cores e tipos de fonte, definir o espaçamento dos elementos da página, criar animações e efeitos visuais, entre outros (*MDN*, 2019a).

A estrutura básica de um arquivo *CSS* é composta por propriedades e valores, que combinadas formam uma declaração *CSS*. As combinações de declarações *CSS* formam blocos de declarações que, combinados com seletores, formam as regras *CSS* (*MDN*, 2019b).

As propriedades são identificadores que indicam a propriedade que será modificada, como por exemplo, tipo de fonte, cor da fonte, altura, largura e outros. O valor é atribuído à propriedade para definir a modificação no estilo. A combinação de propriedade e valor segue uma sintaxe específica, ilustrada no seguinte exemplo:

```
color: white
```

Neste caso, é atribuído à propriedade cor da fonte a cor branca. O caractere dois-pontos (:) separa as duas entidades.

Os seletores definem os elementos do documento *HTML* onde serão aplicadas as alterações de estilo. Existe uma grande variedade de seletores para marcar os elementos, mas os mais comuns, chamados de seletores simples, são os *id* e *class*. Assim, uma regra *CSS*, segue o padrão do seguinte exemplo:

```
p, #id1, .class1 {
    color: white;
    background-color: black
}
```

Neste bloco são utilizados os seletores *p*, que aplicará o estilo a todos os elementos <*p*> no *HTML*, *#id* que aponta para o elemento *HTML* com atributo *id="id1"* e *.class1* aponta para todos os elementos com atributo *class="class1"*. Os caracteres cerquilha (#) e ponto (.) são especiais na sintaxe, e servem para apontar para os atributos *id* e *class*, respectivamente. Os caracteres "{" e "}" definem um bloco de declarações, que são separadas por ponto-e-vírgula (;).

Existem três maneiras de adicionar as folhas de estilo aos arquivos *HTML*. São elas:

a) *CSS* externo: é escrito um arquivo separado com a extensão *.css*. A ligação entre o *.css* e *.html* se dá pelo marcador <*link*> no cabeçalho - <*head*> - do arquivo *HTML*;

b) *CSS* interno: a regra de estilo é inserida no marcador <*style*> no cabeçalho do arquivo *HTML*;

c) *CSS inline*: a regra *CSS* é adicionada diretamente na *tag* desejada através do atributo *style*.

A Figura 3 ilustra os três tipos de aplicação.

Figura 3 - Adição de *CSS* no *HTML*

```
<head>
  <!-- CSS externo -->
  <link href="nome_do_arquivo.css" rel="stylesheet" type="text/css">

  <!-- CSS interno -->
  <style type="text/css">
       p {
        color: white
       }
  </style>
</head>
<body>
  <!-- CSS inline -->
  <p style="color:red;">Um exemplo de parágrafo.</p>
</body>
```

Fonte: Elaboração própria (2019).

Para ilustrar uma aplicação *CSS* real, o exemplo apresentado na Figura 2 foi estilizado, como mostra a Figura 4.

Figura 4 - Aplicação de *CSS*

```
body {
  background-color: #757575;
}
#p1, .estilo1 {
  text-transform: uppercase;
  font-weight: bold;
  color: white;
  top: 50%;
}
#p1 {
  text-align: center;
}
.estilo1 {
  position: absolute;
  background-color: red;
  border: none;
  border-radius: 30%;
  width: 100px;
  height: 100px;
  left: 40%;
  box-shadow: 0 5px black;
}
```

Fonte: Elaboração própria (2019).

No exemplo da Figura 4, o *HTML* foi omitido e o código apresentado é o *CSS* aplicado ao *HTML*. Usando como seletores os elementos e atributos apresentados na Figura 2, *<body>*, *id="p1"* e *class="estilo1"*, foi alterada a cor de fundo da página, todos os textos foram colocados em caixa-alta, negrito e cor de fonte branca, o alinhamento do parágrafo foi centralizado, o formato, cor de fundo, tamanho e posição do botão também foram modificados.

2.2.3 JavaScript

No contexto *Web*, o *JavaScript* é a principal linguagem de programação utilizada para dar interatividade às páginas. O *JavaScript* é uma linguagem multiparadigma, interpretada, orientada a objetos e multiplataforma (*MDN*, 2019c).

Os elementos básicos para se programar em *JavaScript* são semelhantes a outras linguagens de programação, como *C++* e *Python*.

2.2.3.1 *Variáveis*

O elemento mais básico são as variáveis. Uma variável é declarada pela palavra-chave *var* e serve para alocar um espaço na memória do computador onde serão armazenados valores. A declaração de uma variável segue a seguinte sintaxe:

var nome_da_variavel;

var nome_da_variavel = valor_inicial;

A palavra-chave *var* e o nome da variável são elementos obrigatórios na declaração, o valor pode ser omitido. O ponto-e-vírgula é obrigatório somente para

declarações na mesma linha, mas é considerado uma boa prática adicioná-lo ao fim de qualquer declaração.

As variáveis são sensíveis ao caso, assim, uma variável chamada *minhaVariavel* não será a mesma que *minhavariavel*.

Quando uma variável é declarada não é necessário definir o tipo de valor que será armazenado e diferentes tipos de valor podem ser atribuídos a uma mesma variável, por exemplo:

```
var x;
x = 1;
x = 'texto';
```

Aqui a variável *x* foi declarada sem atribuição de valor, em seguida é armazenado o numérico 1, em seguida é atribuída uma linha de texto (*string*). Os tipos de dados que uma variável em *JavaScript* pode armazenar são: *string*, *number*, *boolean*, *array* e *object*.

2.2.3.2 *Operadores*

Outro elemento importante em *JavaScript* são os operadores. Os operadores servem para relacionar duas ou mais variáveis e produzir um resultado. A Tabela 1 lista os principais operadores da linguagem.

Tabela 1 - Operadores em *JavaScript*

Operador	**Símbolo**	**Função**	**Exemplo**
Adição, concatenação	+	Usado para somar valores numéricos ou concatenar *strings*	2 + 2; "*Hello*" + "*World*";
Subtração, multiplicação divisão	- * /	Operações matemáticas	2-2; 2*3; 6/2;
Atribuição	=	Associa valores a variáveis	x = 2 x = *true*

Operador de comparação	===	Testa se dois valores são iguais e retorna verdadeiro ou falso	2 === 'x' // falso
			3 === 3 // verdadeiro

Negação	!	Retorna o valor oposto ao valor testado	!3 // *false* !*false* // *true*

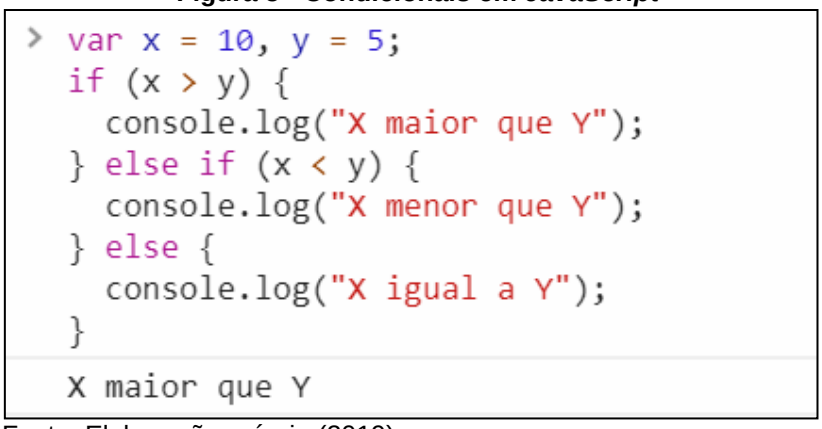
Fonte: Adaptado de *MDN* (2019c).

2.2.3.3 *Condicionais*

Uma outra caraterística comum da linguagem *JavaScript* com outras linguagens são as estruturas condicionais, que permitem tomar decisões de acordo com determinadas condições. As condicionais básicas de *JavaScript* são *if-else*, *while*,

switch-case e *for*. A Figura 5 exibe um trecho de código com uma estrutura condicional *if-else*.

Figura 5 - Condicionais em *JavaScript*



Fonte: Elaboração própria (2019).

2.2.3.4 *Funções*

Funções são blocos construtivos que permitem ao programador criar um conjunto de instruções ou tarefa que podem ser reutilizados sempre que necessário, bastando chamar a função pelo seu nome – e argumentos, se necessário – sem a necessidade de reescrever código.

Para definir ou declarar uma função é necessário usar a palavra-chave *function* seguida do nome da função, lista de argumentos e a declaração da função:

function nomeDaFuncao (arg1, arg2, ..., argN) { //sequência de instruções}

O exemplo da Figura 6 mostra a declaração e o uso de uma função. A função, de nome soma, recebe dois números como argumento e retorna a soma destes dois números. Se a função for atribuída a uma variável, o valor do retorno será armazenado na variável, como também pode ser observado na figura. Além da função soma, a imagem exibe uma função que imprime no console o argumento passado: *console.log()*. Esta função é uma das inúmeras funções disponíveis na biblioteca de *JavaScript* para uso do programador. A *console.log()* imprime no console os parâmetros passados como argumento.

Figura 6 - Exemplo de função em *JavaScript*

```
> // declaração da função
  function soma(numA, numB) {
    return numA + numB;
  }

  // chamada da função
  var x = soma(1, 1)
  var y = soma(10, 5)

  // impressão no console
  console.log("x = " + x)
  console.log("y = " + y)
  x = 2
  y = 15
```

Fonte: Elaboração própria (2019).

2.2.3.5 *Eventos*

Eventos são estruturas de código sensíveis ao que ocorre no navegador do usuário. Quando um evento é disparado, uma instrução é chamada para processar este evento. O exemplo mais básico de evento no navegador é o clique, mostrado na Figura 7. No exemplo da imagem, sempre que o usuário clicar em qualquer lugar da página, será impresso no console a palavra ‘Clique!’ (*MDN*, 2019c).

Figura 7 - Exemplo de evento em *JavaScript*

```
> document.querySelector('html').onclick = function() {
      console.log('Clique!');
  }
< ƒ () {
      console.log('Clique!');
  }
  2 Clique!
```

Fonte: Adaptado de *MDN* (2019c).

2.2.4 PHP

O *PHP* (*PHP: Hypertext Preprocessor*) é uma linguagem de *script* muito utilizada no desenvolvimento *web*. Diferentemente do *JavaScript*, que é executado no lado do cliente, os *scripts PHP* geralmente são executados no servidor e o resultado do *script* é enviado ao navegador. O navegador não tem acesso ao código fonte que

gerou o conteúdo recebido. Isso possibilita a criação de páginas *HTML* dinâmicas, que entreguem conteúdo personalizado para os usuários (*The PHP Group*, 2018).

Um *script PHP* pode ser adicionado diretamente dentro de um arquivo *HTML* através das *tags* *<?php* e *?>*, como no trecho de código abaixo:

```
<?php     echo "Olá, eu sou um script PHP!";    ?>
```

2.2.4.1 *Variáveis*

Assim como em outras linguagens de programação, o *PHP* permite o uso de variáveis que podem armazenar diferentes tipos de dados. Os tipos de dados primitivos em *PHP* são: *boolean*, inteiro, número em ponto flutuante, texto, *array* e objetos.

A declaração de variáveis em *PHP* é feita usando o caractere cifrão ($) seguido por um nome para a variável, assim:

```
<?php
$nome_da_variavel = ‘conteudo_armazenado’;
?>
```

Os nomes das variáveis são sensíveis ao caso e toda declaração de variável deve ser encerrada com ponto-e-vírgula (;).

2.2.4.2 *Operadores*

A função dos operadores em *PHP* é realizar operações entre um ou mais valores e retornar um valor, resultado da operação.

Os operadores mais comuns em *PHP* estão listados na Tabela 2.

Tabela 2 - Operadores em *PHP*

Operadores aritméticos		
Operador	**Função**	**Exemplo**
+	Adição	$a + $b
-	Subtração	$a - $b
*	Multiplicação	$a * $b
/	Divisão	$a / $b
%	Módulo (resto da divisão)	$a % $b
Operadores de atribuição		
Operador	**Função**	**Exemplo**
+=	Atribuição e adição	$a += $b
-=	Atribuição e Subtração	$a -= $b
*=	Atribuição e Multiplicação	$a *= $b
/=	Atribuição e Divisão	$a /= $b
%=	Atribuição e Módulo	$a %= $b
.=	Atribuição e concatenação	$a .= $b

Operadores de incremento e decremento		
Operador	**Nome**	**Função**
++$a	Pré-incremento	Incrementa $a em um, e então retorna $a.
$a++	Pós-incremento	Retorna $a, e então incrementa $a em um.
--$a	Pré-decremento	Decrementa $a em um, e então retorna $a.
$a--	Pós-decremento	Retorna $a, e então decrementa $a em um.
>	Maior que	$a > $b
<=	Menor ou igual	$a <= $b
>=	Maior ou igual	$a >= $b
Operadores lógicos		
Operador	**Nome**	**Exemplo**
AND	E	(10 > 7) AND (9 == 9)
OR	Ou	(10 > 7) OR (9 == 9)
XOR	Ou exclusivo	(10 > 7) XOR (9 == 9)
!	Negação	! (10 > 7)
&&	E	(10 > 7) && (9 == 9)
||	Ou	(10 > 7) || (9 == 9)

Fonte: adaptado de Aprender *PHP* (2010).

2.2.4.3 *Condicionais*

As estruturas condicionais em *PHP*, assim como em outras linguagens, servem para controlar o fluxo do *script*, alterando o que deve ser executado de acordo com os resultados de operações lógicas ou de comparadores. Um exemplo de laço condicional pode ser observado na figura.

Outras estruturas de repetição comuns em *PHP* são: *while*, *do-while*, *for*, *foreach* e *switch*.

Figura 8 - Condicional em *PHP*

```php
<?php
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
?>
```

Fonte: *The PHP Group* (2019).

2.2.4.4 *Funções*

Como em outras linguagens, em *PHP* também podem ser utilizadas as funções para agrupar instruções.

A declaração de uma função em *PHP* é feita através da palavra-chave *function* seguida do nome da função, os argumentos – opcionais – e o corpo da função, delimitado pelos caracteres *chaves* ({ e }). As funções podem ou não retornar algum conteúdo através da instrução *return*.

A Figura 9 ilustra uma função em *PHP*.

Figura 9 - Função em *PHP*

```php
<?php
function foo ($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Exemplo de função.\n";
    return $valor_retornado;
}
?>
```

Fonte: *The PHP Group* (2019).

2.2.5 XML

XML – eXtensive Markup Language – é uma linguagem de marcação simples e flexível, semelhante a *HTML* e tem como propósito principal a transferência de dados através de diferentes sistemas e a separação da apresentação dos dados (*W3C*, 2008).

As *tags* em *XML* não são pré-definidas, como em *HTML*, e podem ser criadas pelo usuário, seguindo uma sintaxe pré-estabelecida.

Um documento *XML* básico apresenta um elemento raiz, que será o elemento 'pai' de todos os demais elementos. O documento apresenta uma hierarquia baseada no aninhamento das *tags*. Todo elemento criado deve, obrigatoriamente, apresentar uma *tag* de abertura (<*tag*>) e uma tag de fechamento (</*tag*>) e o nome das *tags* é sensível ao caso.

Além das *tags* de abertura e fechamento, um elemento *XML* pode apresentar, ainda, texto, atributos e outros elementos.

A Figura 10 ilustra um documento *XML*.

No exemplo da Figura 10, o elemento raiz é a *tag* <*person gender="female*>. Esta *tag* também apresenta um atributo (*gender="female"*) e outros elementos – as *tags firstname* e *lastname* – como conteúdo. As *tags firstname* e *lastname* tem como conteúdo o texto Anna e Smith, respectivamente. Todos os elementos apresentam *tags* de abertura e fechamento.

Figura 10 - Exemplo de *XML*

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

Fonte: *W3Schools* (2019).

Um arquivo *XML* tem como finalidade armazenar dados de forma ordenada. Sozinho ele não produz um conteúdo prático. Um uso comum para o *XML* é armazenar dados que podem ser injetados em um arquivo *HTML* usando linguagens capazes de manipular o *HTML*, como *JavaScript* e *PHP*.

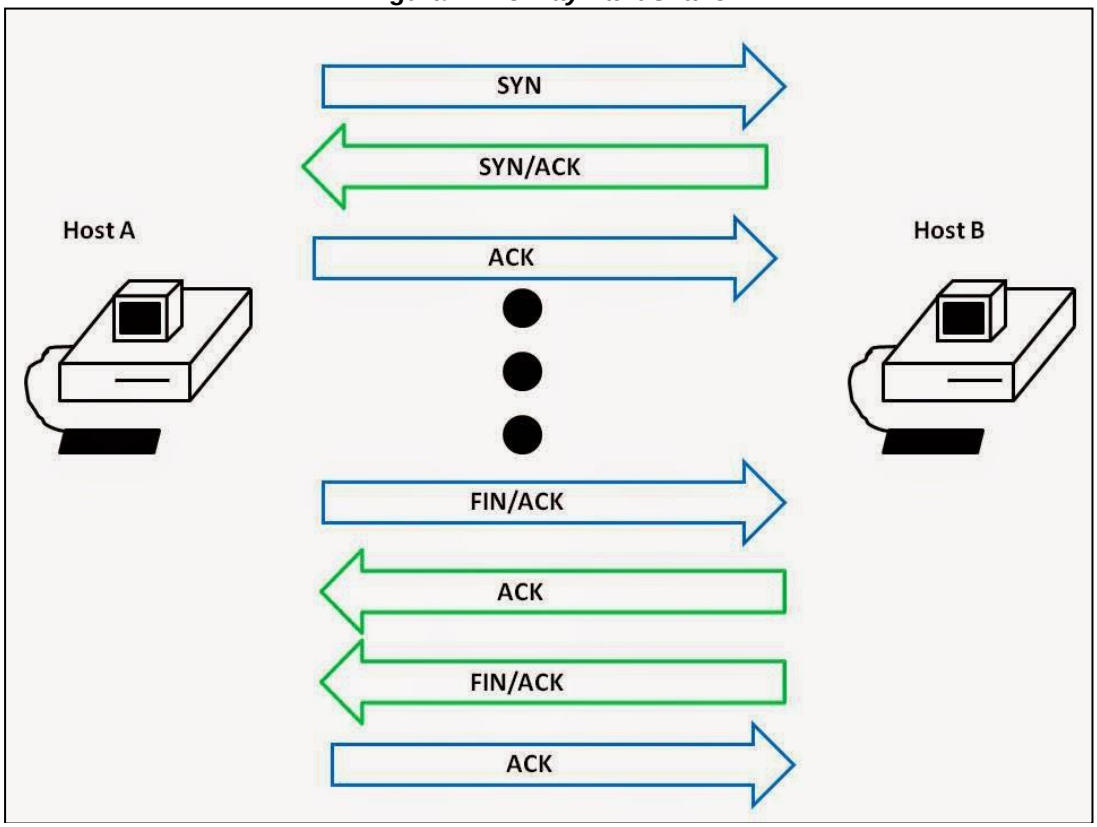
2.2.6 TCP/IP

O *TCP/IP* é a combinação de diversos protocolos que definem seu funcionamento, mas dois deles são consideras de maior importância. O Protocolo de Internet **–** *Internet Protocol* - (*IP*) é a camada de rede principal, responsável pelo endereçamento, roteamento e outras funções relacionadas à internet. O Protocolo de Controle de Transmissão – *Transmission Control Protocol* – (*TCP*) é um protocolo da camada de transporte, responsável pela conexão, gestão e confiabilidade do transporte de dados entre os processos envolvidos (KOZIEROK, 2005).

A comunicação numa sessão *TCP/IP* se dá por um processo conhecido como *3-way handshake*, ilustrado na Figura 11.

Esse processo é iniciado quando um dispositivo **A** envia uma requisição para se comunicar com um dispositivo **B**. Esta requisição contém uma *flag SYN* (*Synchronisation*) e algumas informações relacionadas ao processo de comunicação. O dispositivo **B** responde à requisição enviando um pacote semelhante, com as *flags SYN* e *ACK* (*Acknowledgment*). Por fim, o dispositivo **A** responde com um pacote com a *flag ACK*. Ao finalizar este processo, os dois dispositivos tem as informações necessárias para iniciar a comunicação. Um processo semelhante também é feito para encerrar a comunicação. Além disso, o *TCP* apresenta diversos mecanismos para garantir a integridade dos dados: entrega ordenada dos dados, *checksum*, confirmação de recepção, temporizadores e outros.

Figura 11 - *3-way handshake*



Fonte: DS (2014)

2.2.7 HTTP

O Protocolo de Transferência de Hipertexto – *Hypertext Transfer Protocol* – (*HTTP*) é um protocolo da camada de aplicação enviado sobre *TCP/IP* e usado, principalmente, para a transferência de documentos de hipertexto, como *HTML* e *XML*. É um protocolo cliente-servidor, onde o cliente, geralmente um navegador *web*, inicia a comunicação com o servidor através de requisições (*requests*). As mensagens enviadas do servidor para o cliente são chamadas de respostas (*responses*) (*MDN*, 2019d).

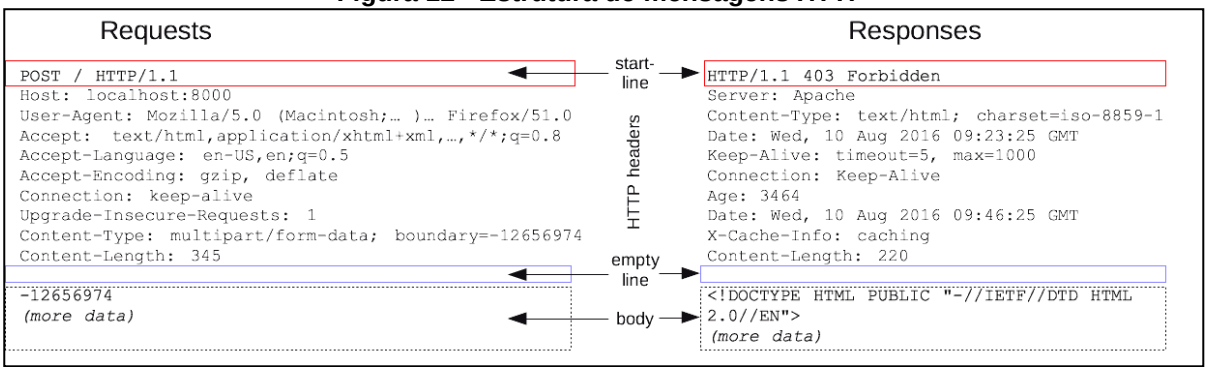
Embora os principais arquivos transferidos via *HTTP* sejam documentos hipertexto, a flexibilidade do protocolo permite que diversos outros tipos de arquivos sejam transferidos, como imagens, vídeos, áudio e outros.

De acordo com Totty et al. (2009), as mensagens de requisições e respostas apresentam estrutura semelhantes (Figura 12) e são compostas por:

a) Uma linha inicial (*start-line*) que descreve a requisição ou *status* de sucesso ou falha;
b) Um cabeçalho *HTTP* (*header*) que especifica a requisição ou descreve o corpo incluso na resposta;

c) Uma linha em branco, indicando que a meta-informação foi enviada;
d) Um corpo (*body*) contendo dados associados à requisição ou o documento associado a resposta.

Figura 12 - Estrutura de mensagens *HTTP*



Fonte: *MDN* (2019d)

As requisições que o cliente envia ao servidor são chamados de métodos *HTTP* e indicam ao servidor o que o cliente está solicitando. A tabela lista os principais métodos de requisição.

Tabela 3 - Principais métodos *HTTP*

Método	**Descrição**
GET	Solicita a representação de um recurso especifico
PUT	substitui as atuais representações do recurso de destino pela carga de dados da requisição
POST	submete uma entidade a um recurso específico
DELETE	remove um recurso específico

Fonte: Adaptado de *MDN* (2019d)

Toda requisição do cliente é respondida com uma mensagem que contém um código de estado (*status code*). A Tabela 4 lista alguns códigos de estado bastante comuns.

Tabela 4 - Principais códigos de estado *HTTP*

Código de estado	**Descrição**
200	Ok. Documento retornado corretamente
302	Redirecionado. O recurso foi movido para outro servidor
404	Não encontrado. Não foi possível encontrar o recurso

Fonte: Adaptado de TOTTY et al. (2009)

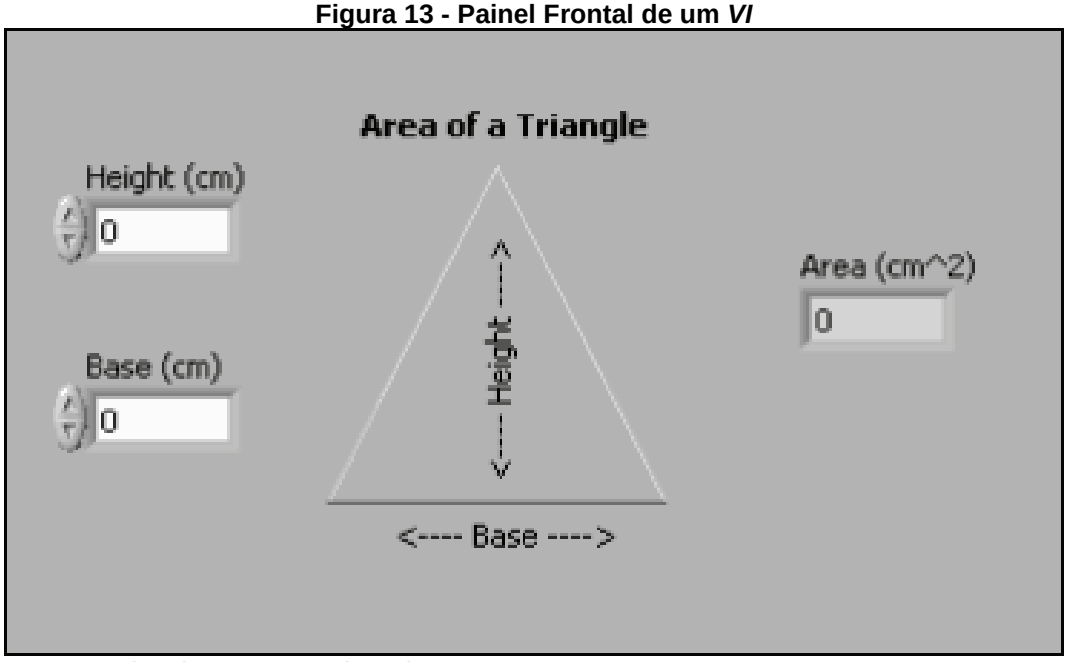
2.2.8 LabView

O *LabView* (*Laboratory Virtual Instrument Engineering Workbench*) é uma linguagem de programação gráfica desenvolvida pela *National Instruments* para facilitar a aquisição de dados e o controle de instrumentos. Com suporte multiplataforma, é também um poderoso *software* de instrumentação e análise de dados (LARSEN, 2011).

Um programa em *LabView*, chamado de instrumento virtual – *Virtual Instruments* (VI) – é construído usando uma interface de desenvolvimento gráfica, que fornece as ferramentas necessárias para a aquisição, processamento e apresentação de dados. Um *VI* pode ser dividido em dois blocos principais: o painel frontal e o diagrama de blocos.

O Painel Frontal apresenta os controles e indicadores do sistema, como *displays*, botões, seletores e outros. A Figura 13 mostra o Painel Frontal de um *VI*.

Figura 13 - Painel Frontal de um *VI*



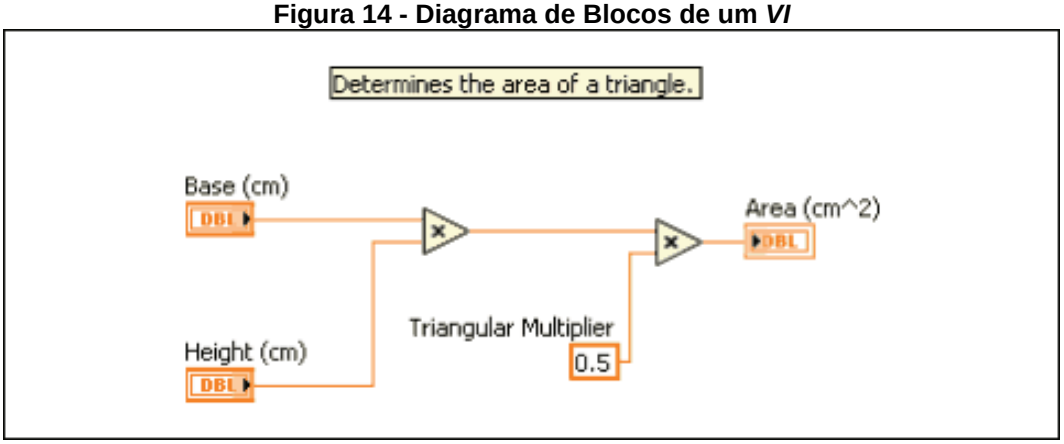
Fonte: *National Instruments* (2019)

No Diagrama de Blocos (Figura 14) é construído o código funcional do *VI*. A programação dos blocos é muito semelhante a um fluxograma, onde o fluxo de dados parte de um elemento, ou nó, para outro.

Um *VI*, com algumas modificações, pode ser transformado em um *subVI*. Para isso, o Painel Frontal é substituído por um painel de conectores, que define as

entradas e saídas de dados da função. Quando inserido dentro de outro *VI* este painel servirá para a conexão dos conectores, ou fios, de entrada e saída.

Figura 14 - Diagrama de Blocos de um *VI*

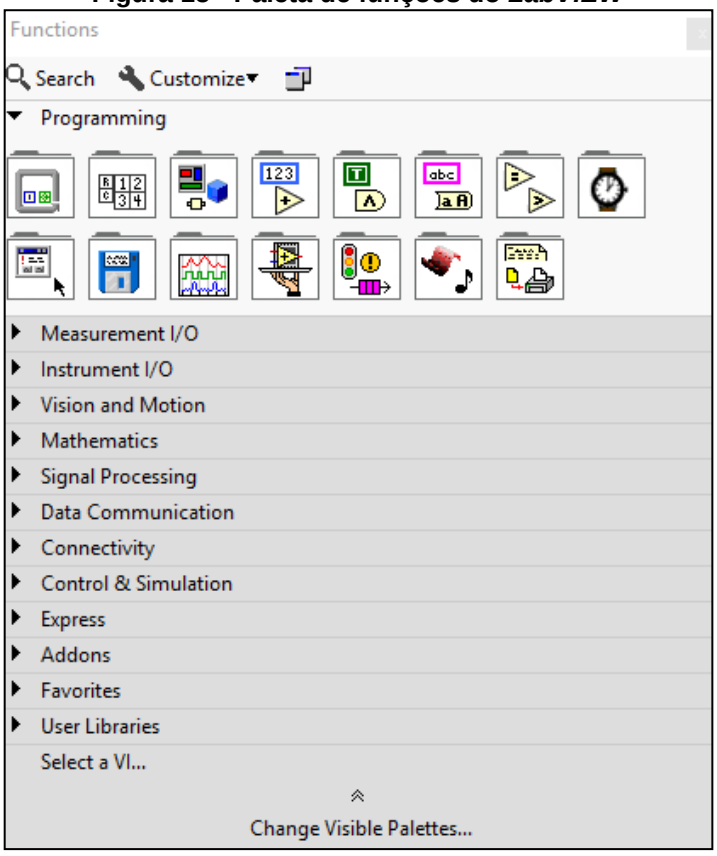


Fonte: *National Instruments* (2019)

Assim como em linguagens convencionais, os diagramas de blocos apresentam variáveis, constantes e estruturas de repetição, decisão e funções.

As funções são elementos fundamentais no *LabVIEW* e o programa apresenta uma paleta bastante extensa delas (Figura 15).

Figura 15 - Paleta de funções do *LabVIEW*



Fonte: Elaboração própria (2019)

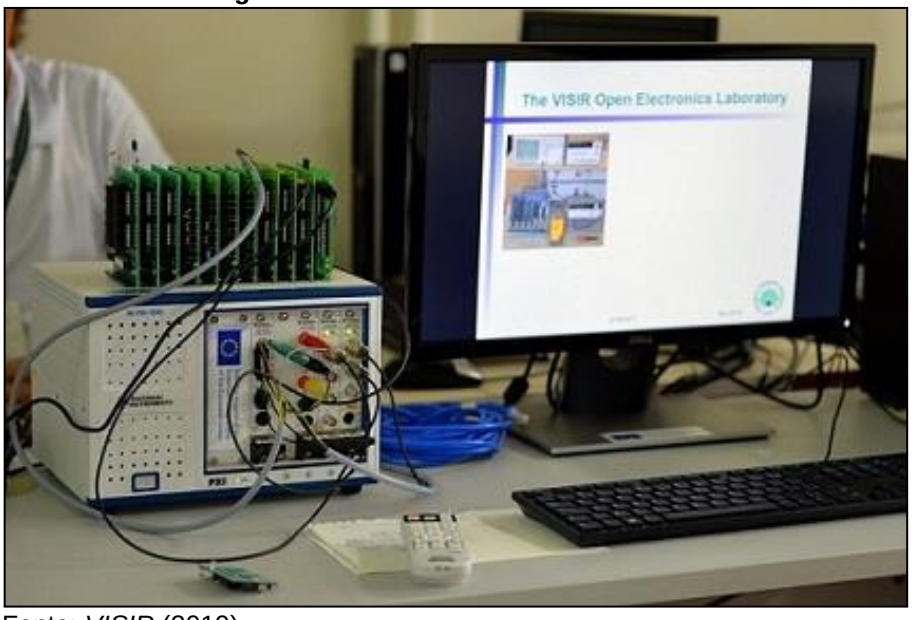
Algumas das funções nativas do *LabVIEW* permitem que o computador se comunique e controle instrumentos reais usando o protocolo *VISA* (*Virtual Instrument Software Architecture*) e funções que permitem a comunicação através de protocolos como, por exemplo, o *TCP/IP*.

2.3 VISIR

O *Virtual Instrument System in Reality* (*VISIR*) é um laboratório remoto que permite o desenho, conexões e medições de circuitos eletrônicos. Seu desenvolvimento foi iniciado em 2006 no departamento de processamento de sinais do Instituto de Tecnologia de Blekinge (BTH). A proposta do laboratório é apresentar ao usuário um ambiente idêntico ao encontrado em um laboratório real. Através de um computador ou outro dispositivo que permita acessar o laboratório o usuário pode interagir com instrumentos e equipamentos que simulam modelos reais, podendo montar circuitos e configurar os instrumentos. O sistema constrói o experimento do usuário usando componentes reais e retorna para a tela as medições desejadas (TAWFIK et al, 2012).

O *hardware* que compõem o *VISIR* é constituído de um computador conectado à internet, instrumentos *PCI eXtensions for Instrumentation* (*PXI*) e a matriz de comutação de relés. O equipamento completo ocupa um espaço bastante reduzido, podendo ser instalado em uma mesa de trabalho comum, como pode ser observado na Figura 16.

Figura 16 - Bancada de trabalho online



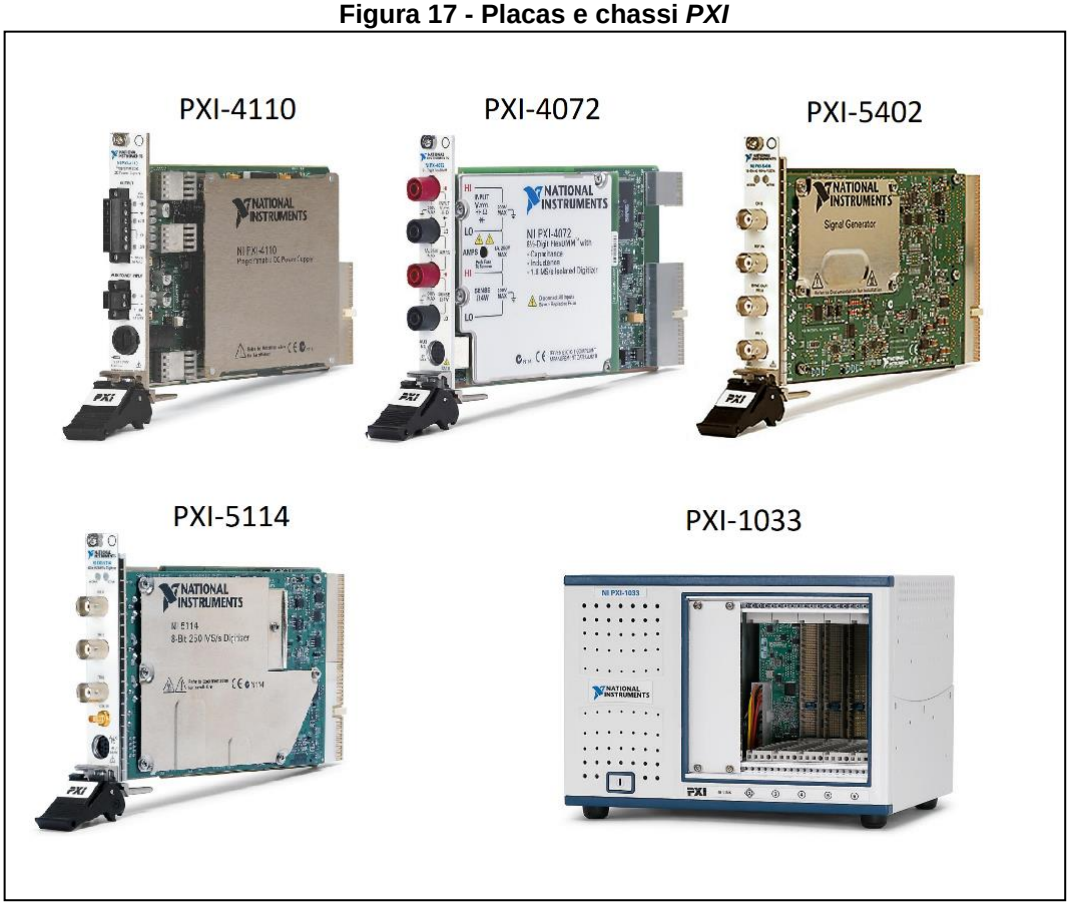
Fonte: *VISIR* (2019)

2.3.1 Plataforma PXI

Para emular uma bancada comum e permitir a realização de experimentos o *VISIR* conta com instrumentos tradicionais de uma bancada de eletrônica: fonte de tensão, gerador de funções, osciloscópio e multímetro digital. Diferentemente dos instrumentos encontrados em um laboratório real, onde cada instrumento tem seu próprio gabinete com botões, seletores e *displays*, no *VISIR* os instrumentos são placas modulares instaladas em um chassi. As placas têm as mesmas funções de instrumentos convencionais e o chassi é responsável por energizar as placas e pela comunicação da plataforma com o computador através do protocolo *Peripheral Component Interconnect* (*PCI*) (TAWFIK, 2011).

No IFSC-Florianópolis os instrumentos instalados são: fonte de tensão ajustável PXI-4110, multímetro PXI-4072, gerador de funções PXI-5402 e osciloscópio PXI-5114. O chassi é um PXI-1033. Todos são fabricados pela *National Instruments* e podem ser observados na Figura 17.

Figura 17 - Placas e chassi *PXI*



Fonte: elaboração própria (2018)

2.3.2 Matriz de comutação de relés

A matriz de comutação de relés é um conjunto de placas padronizadas e empilháveis, desenvolvidas para operar com circuitos eletrônicos analógicos de baixa frequência. Existem dois tipos de placas: placas de instrumentos, onde são conectados os módulos *PXI,* e placas de componentes, onde são instalados os componentes eletrônicos – resistores, capacitores, circuitos integrados e outros – disponibilizados para uso.

Cada placa conta com um microcontrolador PIC16F767, chamado de *board controller* e dois conectores, um com 28 pinos e outro com 17 pinos, que se interconectam quando as placas são empilhadas (TAWFIK, 2011).

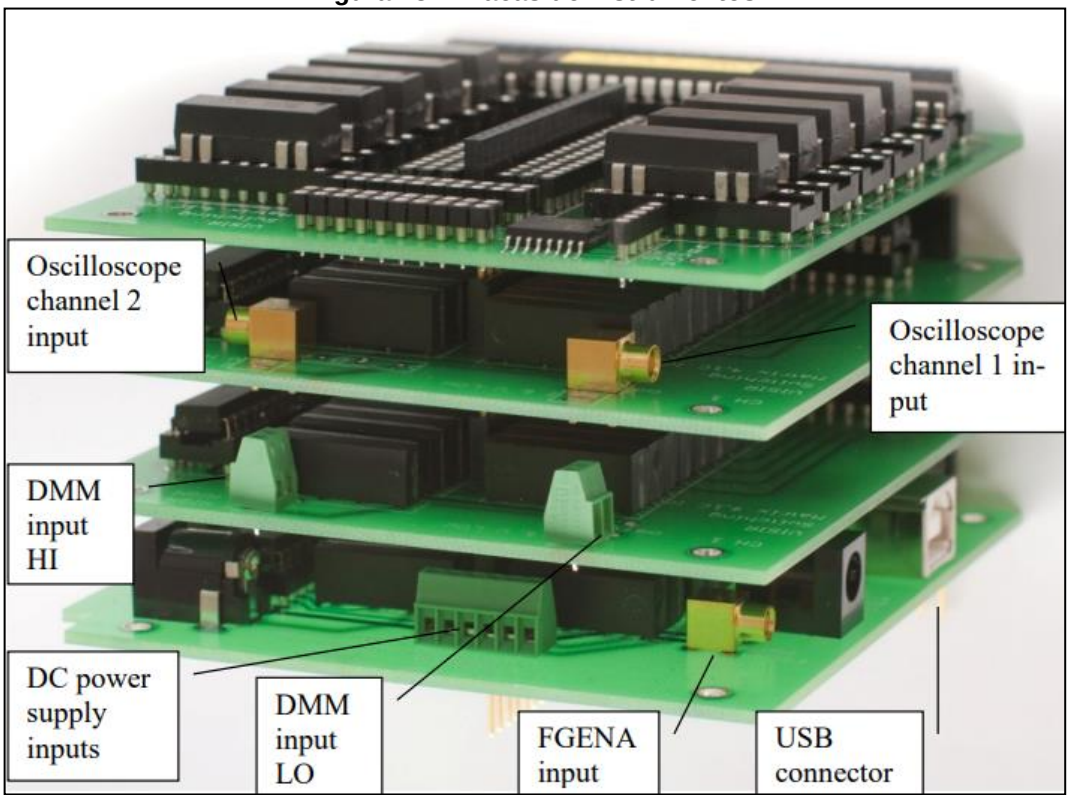
O conector de 28 pinos é o barramento digital e é usado para a comunicação entre as placas. O conector de 17 pinos é chamado de conector de nós. É através deste conector que os componentes e instrumentos são conectados para montar os experimentos. Os nós são divididos em dois grupos:

a) A à I e 0 (GND): É neste grupo que os componentes podem ser conectados;
b) X1 à X6 e COM: Apenas a placa *source* pode ser conectada a estes nós.

As placas de instrumentos (Figura 18) fazem a conexão da matriz de comutação com os instrumentos *PXI*. Cada módulo *PXI* tem uma placa correspondente, onde serão conectados os cabos e ponteiras de cada instrumento. O diagrama da Figura 19 ilustra estas conexões. Para simplificar o diagrama, os relés foram representados como chaves simples.

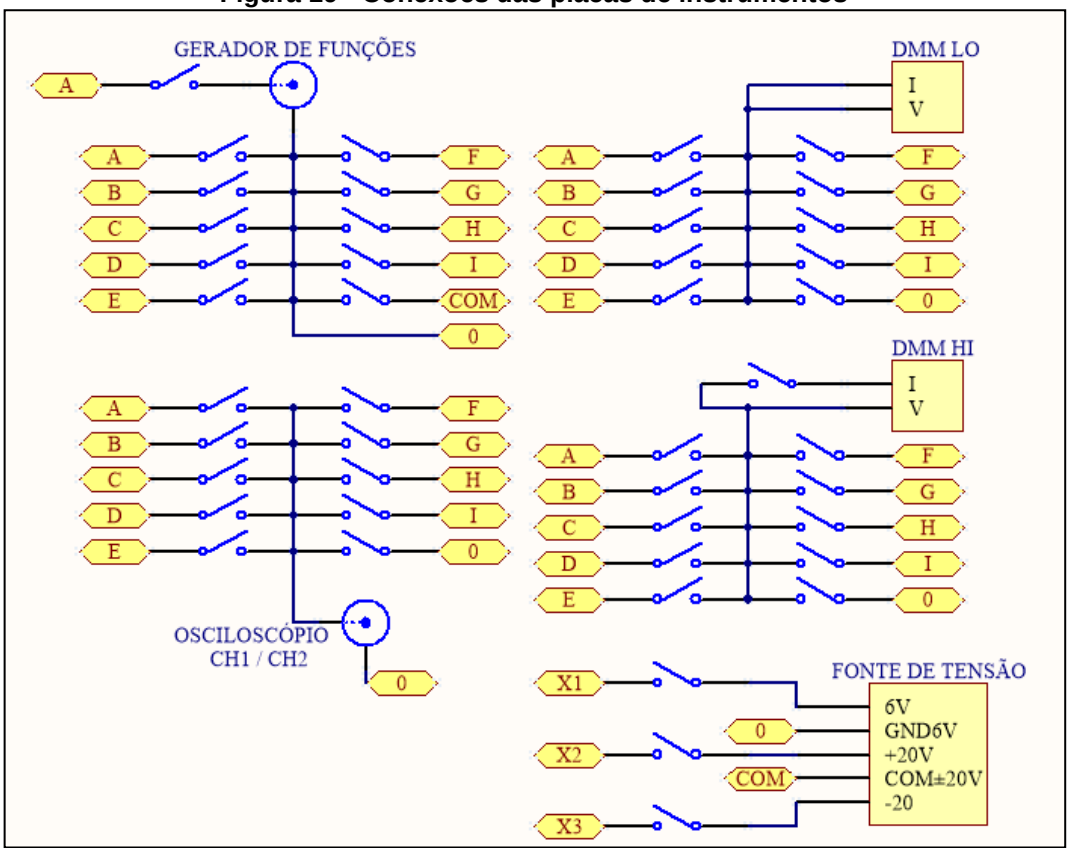
A placa *source*, onde são conectadas as fontes de tensão e gerador de funções conta, ainda, com um conector de 2,1 mm para uma fonte de alimentação externa que fornece energia para as bobinas dos relés e um microcontrolador PIC18F4550, que é o controlador da matriz de comutação de relés. Esse microcontrolador se conecta ao computador via *Universal Serial Bus* (*USB*) e comunica com os PIC16F767 de cada placa via *Inter-Integrated Circuit* (I^2C). Cada *board controller* tem um endereço I^2C próprio gravado no *firmware* (GUSTAVSSON, 2016).

Figura 18 – Placas de instrumentos



Fonte: GUSTAVSSON (2016)

Figura 19 - Conexões das placas de instrumentos

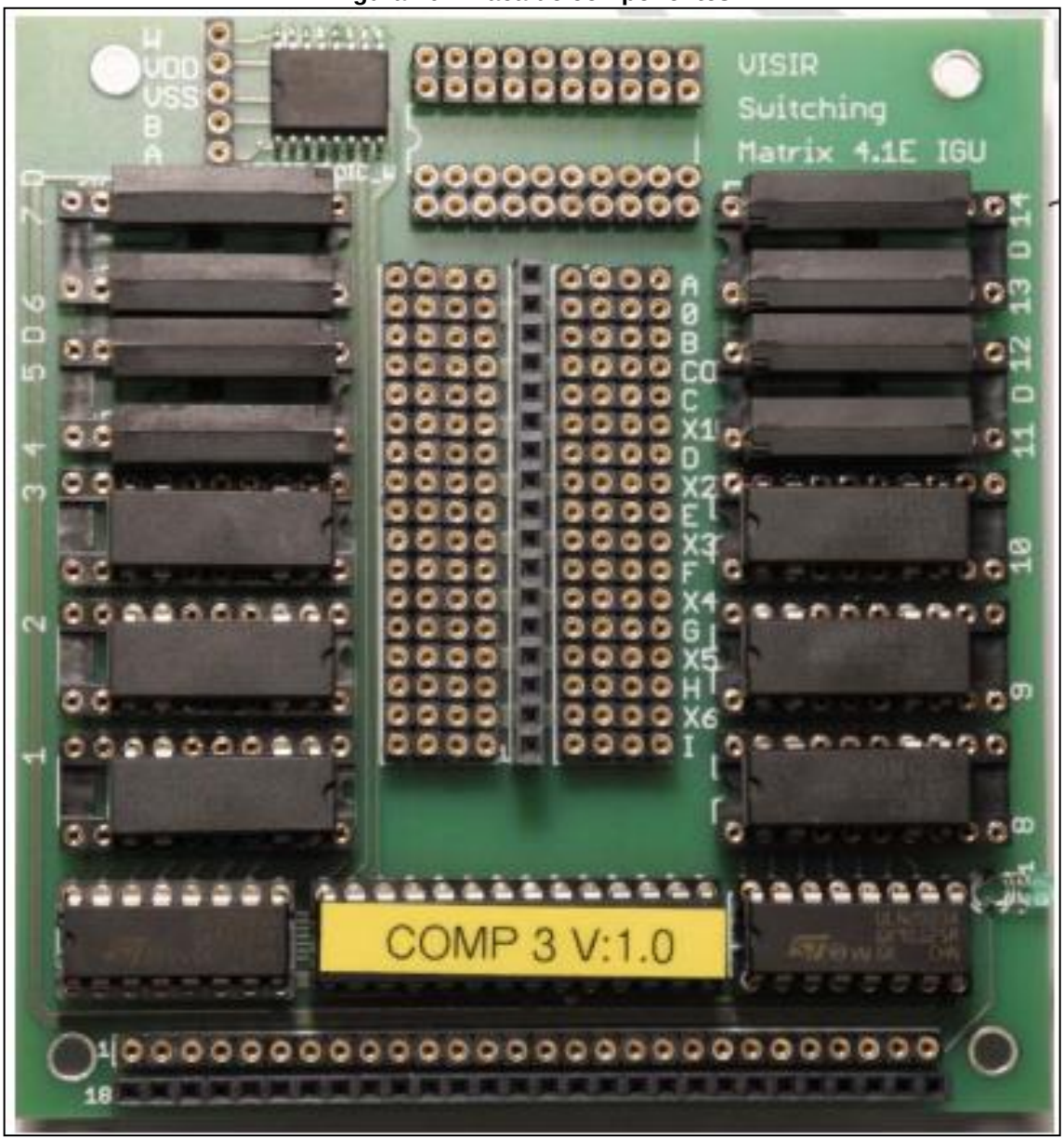


Fonte: Adaptado de GUSTAVSSON (2016)

As placas de componentes (Figura 20) possuem 10 soquetes para componentes de dois terminais conectados diretamente aos relés e um soquete de 20 pinos, que pode ser utilizado para componentes mais complexos. As placas mais recentes, na versão 4.1E, suportam 6 relés duplos – relés 1, 2, 3, 8, 9 e 10 – e 8 relés simples – relés 4, 5, 6, 7, 11, 12, 13 e 14. Os relés simples podem ser substituídos, se necessário, por relés duplos, que serão denominados 5, 7, 12 e 14 (GUSTAVSSON, 2016).

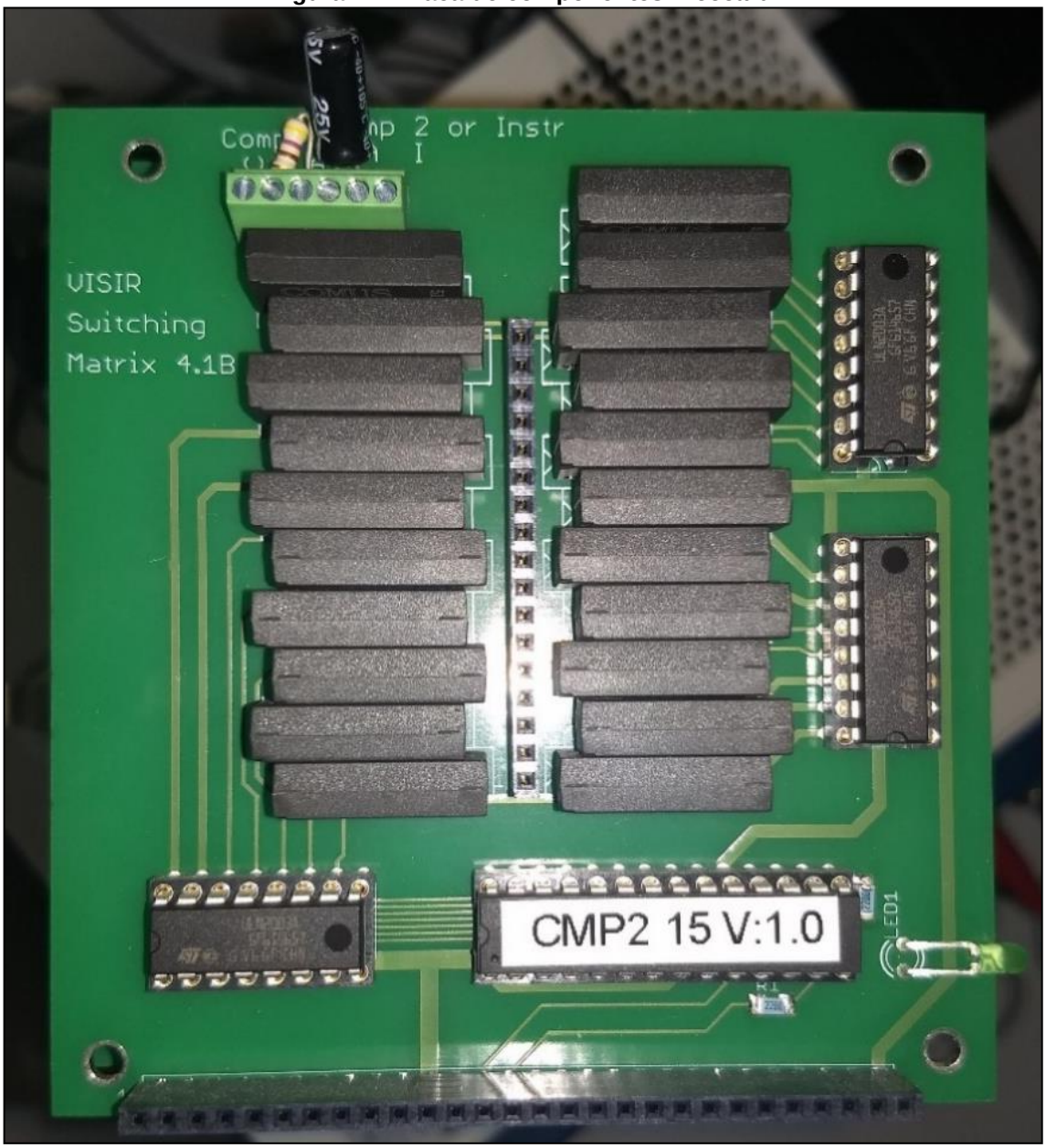
Um segundo tipo de placa de componentes são as chamadas *freecard* (Figura 21). Nestas placas podem ser instalados até dois componentes com dois terminais, que podem ser conectados livremente entre os nós A, D, F, H e 0.

Figura 20 - Placa de componentes



Fonte: GUSTAVSSON (2016)

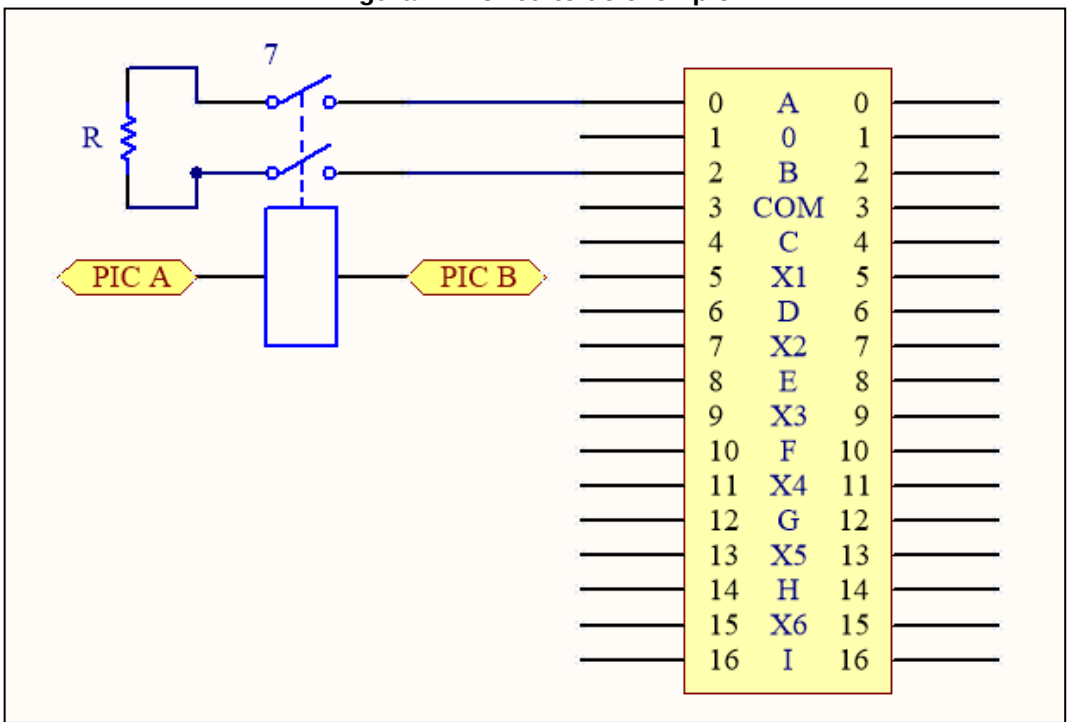
Figura 21 - Placa de componentes *Freecard*



Fonte: elaboração própria (2018)

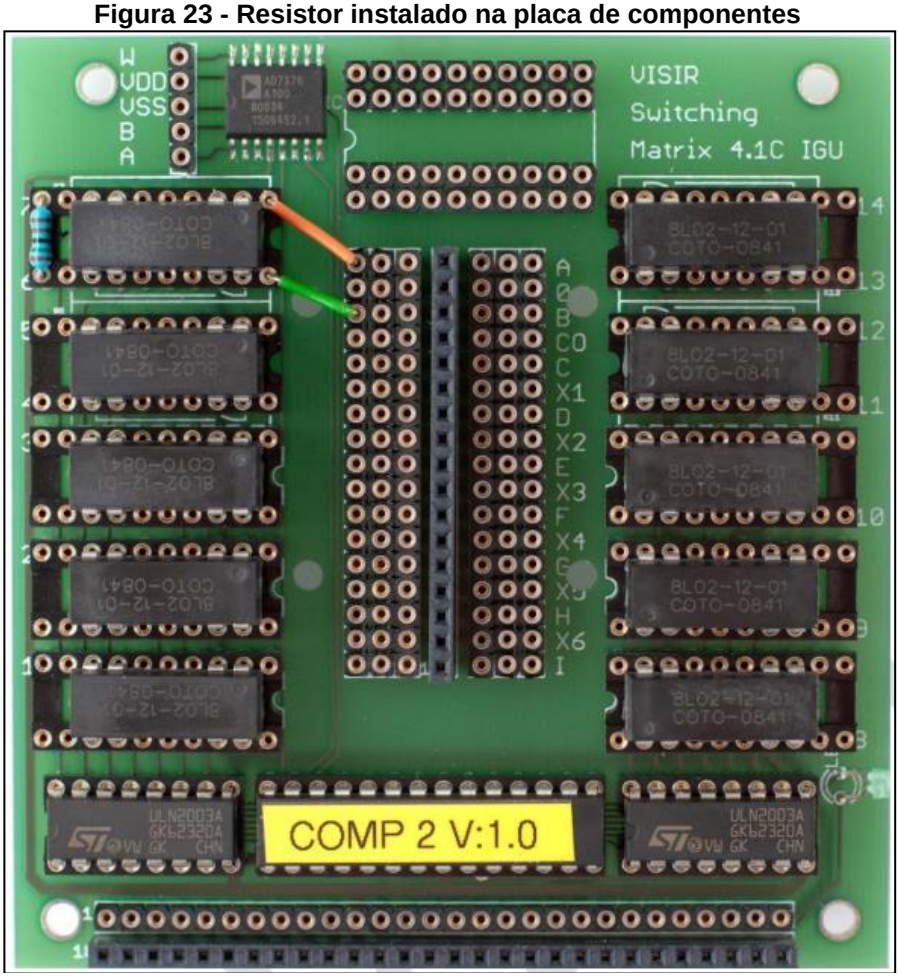
Para conectar um componente na matriz, deve-se instalar o componente usando os soquetes de relés ou o soquete de circuito integrado (CI) de 20 pinos, então, usando fios, o relé será conectado a um determinado nó no conector de nós. A Figura 22 mostra o esquemático um resistor que, através do relé 7, pode ser conectado aos nós A e B. A Figura 23 ilustra esse circuito montado numa placa de componentes.

Figura 22 - Circuito de exemplo



Fonte: elaboração própria (2018)

Figura 23 - Resistor instalado na placa de componentes

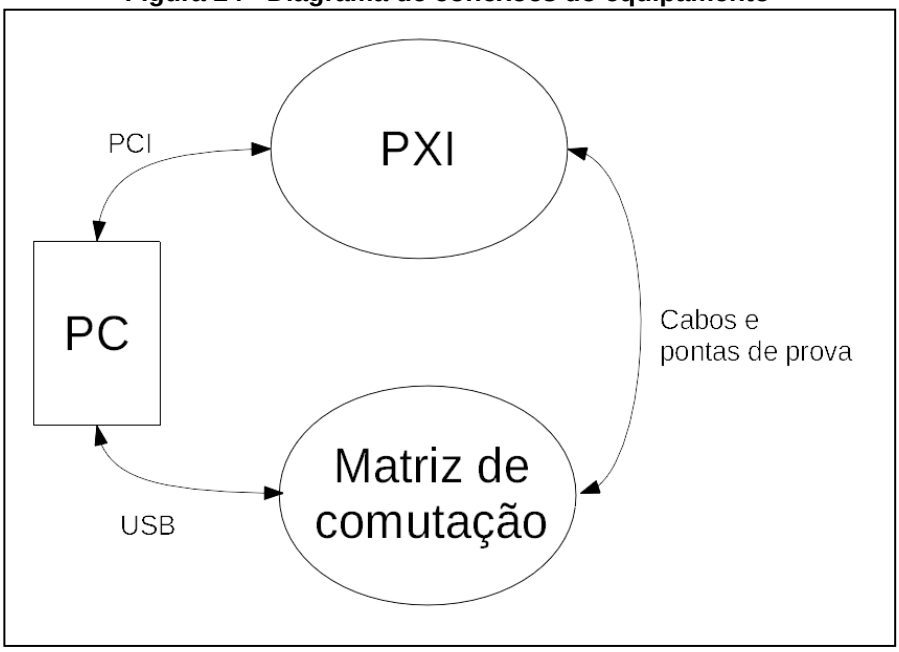


Fonte: GUSTAVSSON (2016)

O controle e interação com a matriz de comutação de relés é feita através de um computador. O diagrama da **Erro! Autoreferência de indicador não válida.** resume as conexões do equipamento.

Além do programa para controlar o *hardware*, este computador também hospeda outros serviços fundamentais para o funcionamento do laboratório remoto.

Figura 24 - Diagrama de conexões do equipamento

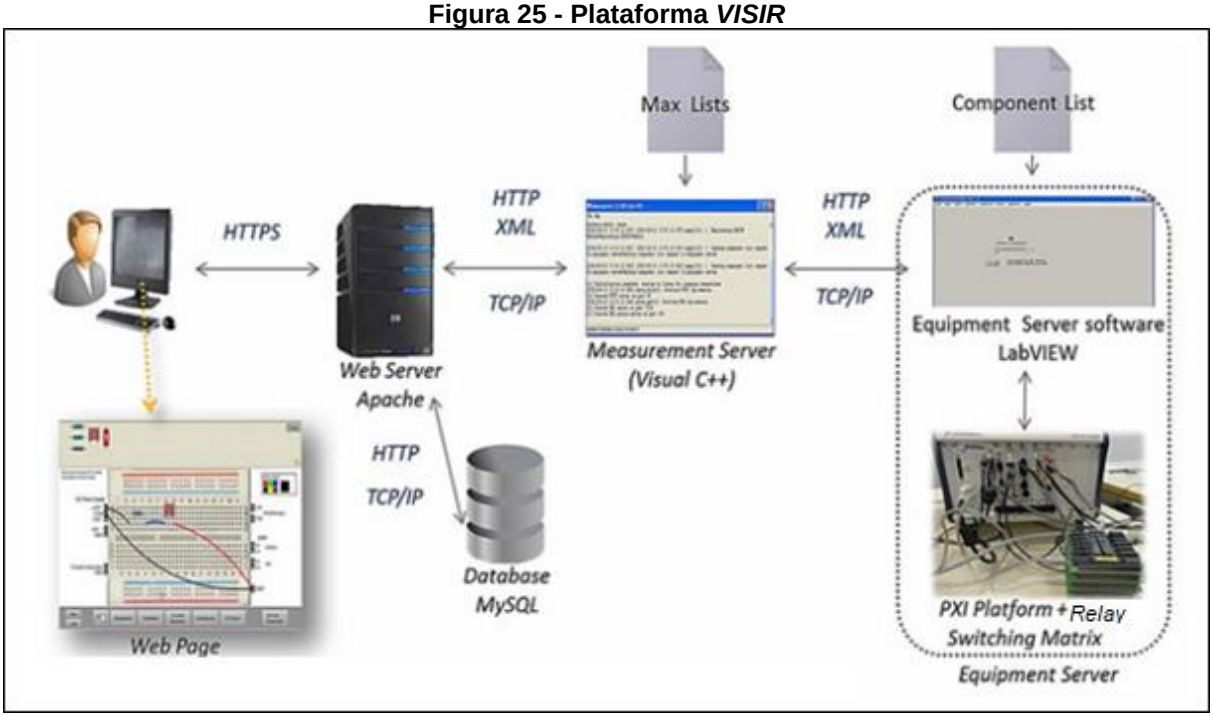


Fonte: elaboração própria (2018)

Segundo Tawfik et al. (2012), a plataforma *VISIR* consiste de quatro blocos: *Equipment Server***,** *Measurement Server*, *Web Server* e *Web Interface* (Figura 25).

Cada bloco representa um servidor que hospeda serviços diferentes, e, embora possam ser hospedados todos em um único computador, é conveniente analisar cada um separadamente buscando entender a função de cada servidor no laboratório remoto, entender como é feita a comunicação entre os servidores e o conteúdo das informações trocadas. Compreender o funcionamento de cada servidor permitirá que sejam feitas alterações no laboratório.

Figura 25 - Plataforma *VISIR*



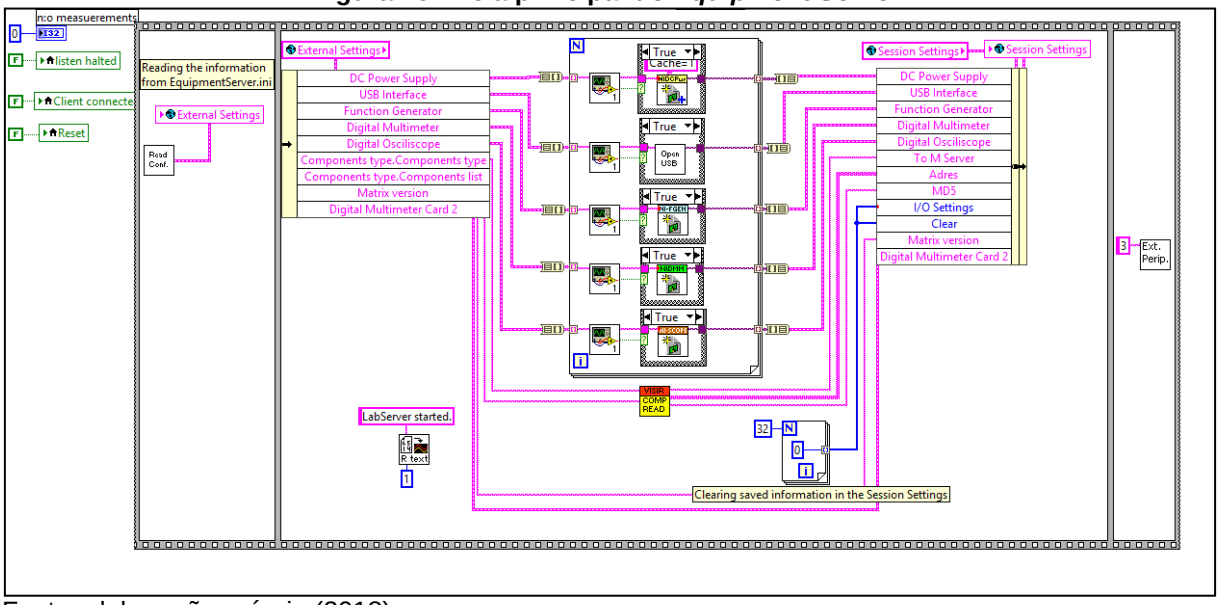
Fonte: modificado de TAWFIK et al. (2012)

2.3.3 Equipment Server

O *Equipment Server* controla os instrumentos *PXI* e a matriz de comutação através de um servidor escrito em *LabView* (Figura 26). Este servidor recebe instruções do *Measurement Server* em formato *XML* em uma sessão *HTTP* encapsulado em *TCP/IP*.

Cada placa que compõe o *hardware* do *VISIR* possui um código-fonte próprio, com funcionalidades específicas. A configuração dos instrumentos é definida pelas requisições do usuário. Cada instrumento apresenta uma série de variáveis que definem como o instrumento irá operar, como resolução, faixa de medição, modo e outros. Os valores dessas variáveis são definidos no servidor *web* e repassados aos demais servidores através de pacotes de dados trocados via arquivos *XML* em requisições feitas pelo navegador.

Figura 26 - Tela principal do *Equipment Server*



Fonte: elaboração própria (2018)

Tabela 5 lista os elementos que podem ser configurados em cada instrumento através de dados enviados nos pacotes de requisições do usuário.

As respostas das requisições retornam, também, em um arquivo *XML*. Mesmo instrumentos que não são de medição, como fonte de tensão ou gerador de funções, retornam algum parâmetro para o usuário. Cada instrumento gera uma *string* com os parâmetros que são anexados ao *XML* que retorna como resposta ao usuário. A Tabela 6 lista estes parâmetros, que foram obtidos a partir da análise das respostas do servidor para diferentes experimentos.

Através desses parâmetros é possível saber o tipo de leitura e medições que podem ser feitas usando o *hardware* do *VISIR* e quais as informações o servidor pode fornecer.

Tabela 5 - Parâmetros de configuração dos instrumentos

GERADOR DE FUNÇÕES	
Waveform	*Sine, Square, Triangle, Ramp Up, Ramp Down, Dc, Noise, User Defined*
Amplitude	*0 – 10.0 V_{pp}*
Frequency	*Sine, square = 20MHz, others = 1MHz*
Dc Offset	*-5 V to +5 V, |(DC Offset + amplitude)| < 10V*
Start Phase	*-180.0 to +180.0 degrees*
Trigger mode	*Single, continuous, stepped, burst*
Burst count	*Immediate, external*
Duty cycle high	*Default = 50%*
User defined waveform	

MULTÍMETRO DIGITAL		
Function		*DC Volts, AC Volts, DC Current, AC Current, 2-wire Resistance, 4-wire Resistance, Frequency, Period, Diode*
Resolution		*3.5 Digit Precision, 4.5 Digit Precision, 5.5 Digit Precision, 6.5 Digit Precision,*
Range		*Autorange = -1.0*
Autozero		*Auto, Off, On, Once*
OSCILOSCÓPIO		
Autoscale		*Enable / Disable*
Horizontal Conf		
	Min. Sample Rate	*Samples per second*
	Reference Position	*Default = 50%*
	Record Length	*Default = 1000 points*
*Channel *2*		
	Vertical Coupling	*AC, DC, GND*
	Vertical Range	*Absolute vertical range*
	Vertical Offset	*Location of the center of vertical range*
	Probe Attenuation	*Any positive real number*
Trigger		
	Source	*Channel 1, Channel 2, Immediate, External trigger*
	Slope	*Positive, negative*
	Coupling	*AC, DC*
	Level	*Voltage threshold for the trigger*
	Holdoff	*Length of time digitizer waits after detecting trigger, in seconds*
	Delay	*Trigger delay time, in seconds*
	Trigger mode	*Normal, Auto, Auto level*
	Timeout	
*Measurement *3*		
	Channel	*Channel 1, Channel 2, Channel 3, Channel 4*
	Selection	*AC Estimate, Area, Average frequency, Average period, Cycle area, DC estimate, Fall time, Falling slew rate, FFT amplitude, FFT frequency, Frequency, Integral, Negative duty cycle, Negative width, None, Overshoot, Period, Phase Delay, Positive duty cycle, Positive width, Preshoot, Rise Time, Rising Slew Rate, Time Delay, Voltage Amplitude, Voltage Average, Voltage Base, Voltage Base To Top, Voltage Cycle Average, Voltage Cycle RMS, Voltage High, Voltage Low, Voltage Max, Voltage Min, Voltage Peak To Peak, Voltage RMS, Voltage Top*

FONTE DE TENSÃO DC	
Parâmetro	Opções / definição
Voltage +6	*0.0 V up to 6.0 V*
Current Limit +6	*0.0 A up to 1.0 A*
Voltage +20	*0.0 V up to 20.0 V*
Current Limit +20	*0.0 A up to 100 mA, 0.0 A up to 1.0 A with external power supply*
Voltage -20	*-20.0 V up to 0.0 V*
Current Limit -20	*0.0 A up to 100 mA, 0.0 A up to 1.0 A with external power supply*

Fonte: adaptado de PROTOCOL (s.d.)

A partir disso, é possível a avaliar a viabilidade e dificuldade de implementar novas funcionalidades ou ferramentas que dependam do equipamento instalado.

Tabela 6 - Parâmetros de resposta do *Equipment Server*

Gerador de funções
Waveform
Amplitude
Frequency
DC offset
Start phase
Trigger mode
Trigger source
Burst count
Duty cycle
Fonte de Tensão DC
DC outputs
*Dc output * 3 (+6V, +20V, -20V)*
DC voltage
DC current
DC voltage actual
DC current actual
DC output enabled
DC output limited
Multímetro digital
id
Function
Resolution
Range
Result

Osciloscópio
Horizontal
Actual Sample Rate
Actual Record Length
Actual Reference Position
*Channel * 2*
Coupling
Vertical range
Vertical offset
Probe attenuation
Gain
Samples
Trigger
Source
Slope
Coupling
Level
Mode
Delay
Received
Measurement 3*
Channel
Selection
Result

Fonte: elaboração própria (2018)

Ainda no *Equipment Server* há um arquivo chamado ‘*component.list’* que define os componentes e instrumentos instalados na matriz. Qualquer alteração no *hardware* deve ser descrita neste arquivo, pois é através dele que o sistema identifica os componentes instalados. Os elementos previstos no sistema são: resistor (R), capacitor (C), curto (*SHORTCUT*), indutor (L), transistores (Q), amplificadores operacionais (OP), potenciômetros (POT), as fontes de tensão (VDCCOM, VDC25V, VDC-25V, VDC6V), gerador de funções (VFGENA, VFGENB), e um componente genérico chamado de caixa-preta (*BLACKBOX*).

O seguinte padrão é seguido para adicionar um componente na ‘*component.list’*:

<tipo de componente>_<nº da placa>_<nº do relé> <nó 1>< nó 2>...<nó ‘N’> <valor>

Usando o resistor da Figura 23 como exemplo:

R_2_7 A B 10k

Onde R é o elemento resistor, 2 é o número da placa de componentes onde o resistor está instalado, 7 é o número do relé no qual o resistor está conectado, A e B são os nós onde o relé conecta o resistor e 10k é valor do resistor.

Se um componente utiliza mais de um relé, estes relés serão separados pelo sinal dois-pontos (:), por exemplo, um curto entre os nós D e G usando o relé 1 da placa 1 e o relé 5 da placa 3:

SHORTCUT_1_1:3_5 D G

A conexão dos instrumentos ou de componentes com três ou mais terminais seguem o mesmo padrão (TAWFIK, 2011). Um exemplo de um amplificador operacional conectado através de cinco relés seria:

OP_4_4:4_5:4_6:4_7:4_14 NC1 D G B NC2 F H NC3 uA741

Os nós NC1, NC2, e NC3 são pinos não conectados do CI, que, somados aos demais nós, totalizam os 8 pinos do amplificador operacional u741.

Os componentes instalados nas *freecards* seguem o mesmo padrão, devendo ser incluídos todos os nós onde pode-se conectar o componente. Um exemplo de um resistor de 1 KΩ instalado em uma *freecard* fica:

R_15_18:15_1 A 0 1k

R_15_18:15_3 A D 1k

R_15_18:15_5 A F 1k

R_15_18:15_7 A H 1k

R_15_4:15_1 D 0 1k

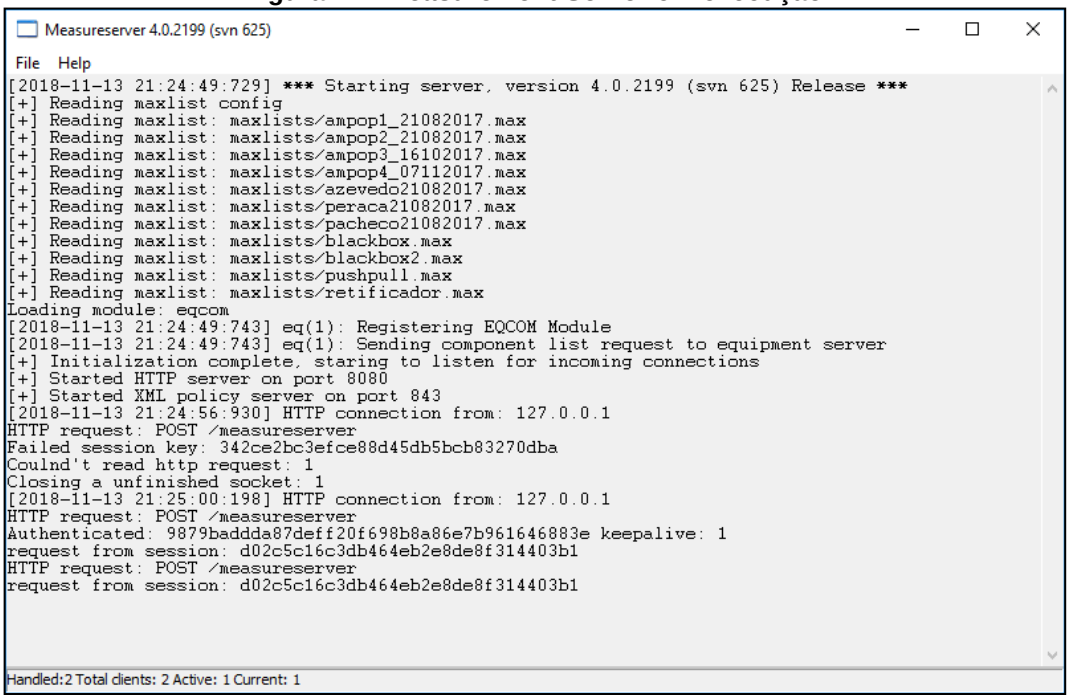
R_15_4:15_5 D F 1k
R_15_4:15_7 D H 1k
R_15_6:15_1 F 0 1k
R_15_6:15_7 F H 1k
R_15_8:15_1 H 0 1k

2.3.4 Measurement Server

O *Measurement* Server é um servidor que age como um “instrutor virtual”, controlando a troca de mensagens entre o servidor *Web* e o *Equipment Server*. Este servidor recebe pacote de dados com um experimento montado pelo usuário no *webserver* e avalia o circuito. Se o experimento for válido, o *Measurement Server* envia o pacote ao *Equipment Server* para executar o experimento. O resultado do experimento retorna ao *Measurement Server* e é repassado para o usuário. Se o experimento não for válido, o *Measurement Server* retorna para o usuário uma mensagem de erro e o experimento não é executado. A Figura 27 mostra a tela principal do *Measurement Server* em execução.

Para determinar se um experimento é valido, o servidor compara o conteúdo do pacote enviado pelo usuário com arquivos chamados ‘*maxlist*’. Nestes arquivos o administrador determina as conexões permitidas e os valores de tensão e corrente disponíveis para os experimentos, tanto para habilitar somente determinados experimentos como para garantir a integridade do laboratório, não permitindo que sejam executados experimentos que possam danificar os equipamentos. É possível ter diversos arquivos **.max* simultaneamente. O servidor fica encarregado de procurar entre os arquivos algum que combine com o experimento do usuário. Mesmo com diversas *maxlist* o sistema nunca irá combinar as informações de diferentes arquivos para validar um experimento.

Figura 27 - *Measurement Server* em execução



Fonte: elaboração própria (2018)

Assim como na '*component.list*', nas *maxlist* os componentes também seguem um padrão para serem adicionados:

<tipo de componente>_<identificador> <nó 1>< nó 2>...<nó 'N'> <valor>

Assim, o resistor da Figura 23 fica:

R_R1 A B 10k

Onde R é o tipo do componente, R1 é um identificador único do componente na *maxlist*, A e B são os nós onde o componente pode se conectar e 10k é o valor do componente.

Os instrumentos que fornecem tensão ou corrente podem ter seus valores limitados através do campo 'valor', como no seguinte exemplo:

VFGENA_FGENA1 A 0 MAX:2.5

VDC+25V_1 H VMAX:25 IMAX:0.5

Neste exemplo, o gerador de funções, pode ser conectado ao nó A e terá sua amplitude máxima limitada em 2,5 V e a fonte de tensão de +25 V pode ser conectada ao nó H, terá sua tensão máxima limitada em 25 V e a corrente máxima limitada em 500 mA (TAWFIK, 2011).

Os componentes que são instalados na *freecard* seguem um padrão diferente. Neles é necessário adicionar o caractere '@' que representa 'grupo'. Assim o resistor de 1 KΩ usado como exemplo anteriormente terá sua representação na *maxlist* da seguinte maneira:

R_R1@1 A 0 1k
R_R2@1 A D 1k
R_R3@1 A F 1k
R_R4@1 A H 1k
R_R5@1 D 0 1k
R_R6@1 D F 1k
R_R7@1 D H 1k
R_R8@1 F 0 1k
R_R9@1 F H 1k
R_R10@1 H 0 1k

Isto garantirá que o componente identificado como pertencente a um grupo só possa ser utilizado uma única vez por experimento. Sem isso, o servidor poderia interpretar que existem 10 resistores de 1 KΩ instalados no sistema.

2.3.5 *Web Server* e *Web Interface*

O *Web Server* hospeda a interface *web* do *VISIR* e o banco de dados em *MySQL* (TAWFIK, 2016). Escrito em *PHP*, o *Web Server* constrói as páginas principais do *VISIR*, como as telas de início e login e também pode acessar o banco de dados e informações relacionados ao usuário, como cadastro no sistema, experimentos salvos, agenda de horários e outros. Este servidor também constrói os *templates* onde será montada a interface do usuário para interação com o sistema.

A interface *web* é onde o usuário interage com o laboratório remoto. O *Web Server*, coletando dados sobre o usuário, gera os arquivos que serão enviados ao navegador do cliente. Estes arquivos são documentos *HTML*, *CSS*, *JavaScript* e *XML*.

Usando um navegador, o usuário se conecta à página *web* (Figura 28) e faz login no laboratório via *Hyper Text Transfer Protocol Secure* (*HTTPS*). Após ser autenticado, pode, então, acessar a bancada de trabalho virtual (Figura 29) onde terá acesso à matriz de contatos virtual e demais instrumentos do sistema, como fonte de tensão (Figura 30), gerador de funções (Figura 31), osciloscópio (Figura 32), multímetro digital (Figura 33) e – clicando no símbolo ‘+’ – uma lista dos componentes disponíveis para uso (Figura 34) (TAWFIK, 2011).

Figura 28 - Página principal do *VISIR*

Fonte: elaboração própria (2018)

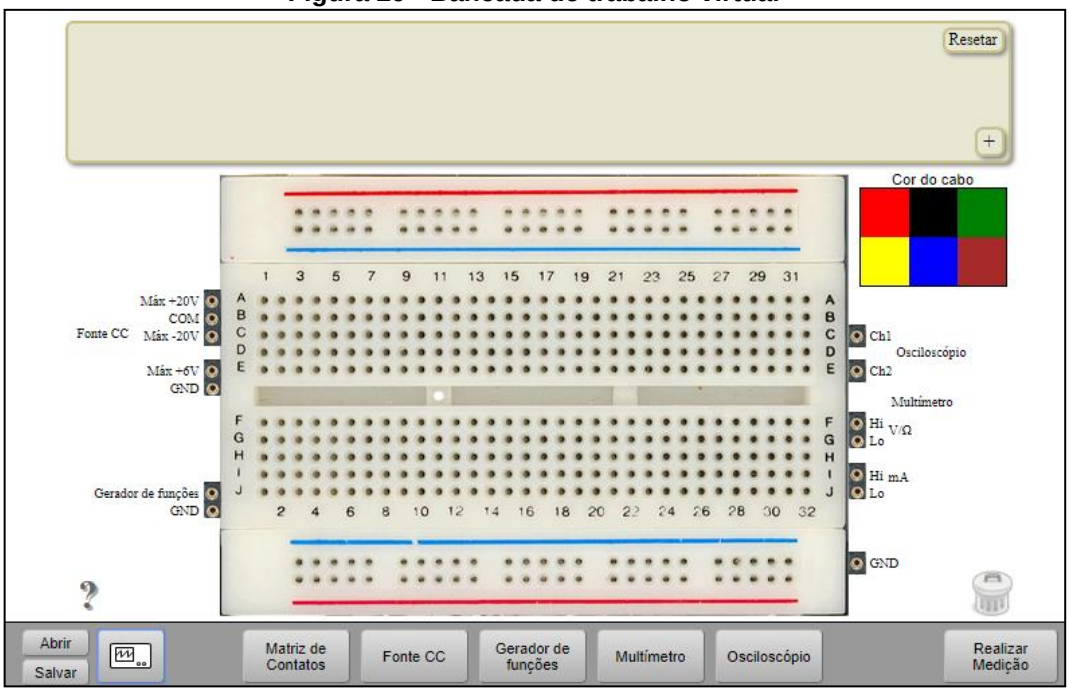
Para todo usuário é atribuído um tipo de conta, com diferentes características, privilégios e propriedades, descritas na Tabela 7.

Tabela 7 - Tipos de contas de usuários

Administrador	• Manipular o conteúdo das páginas, • Criar, atualizar e deletar cursos, • Atribuir professores e instrutores aos cursos, • Modificar e excluir usuários, • Os mesmos privilégios das demais contas.
Professor	• Adicionar ou remover experimentos, • Adicionar ou remover estudantes, • Agendar horários, • Tem os privilégios de estudante
Estudante	• Acesso a cursos em que está matriculado, • Agendar horários.
Convidado	• Conta pública limitada, criada pelo administrador, • Pode ser acessada sem *login*, • Tem acesso limitado a experimentos.

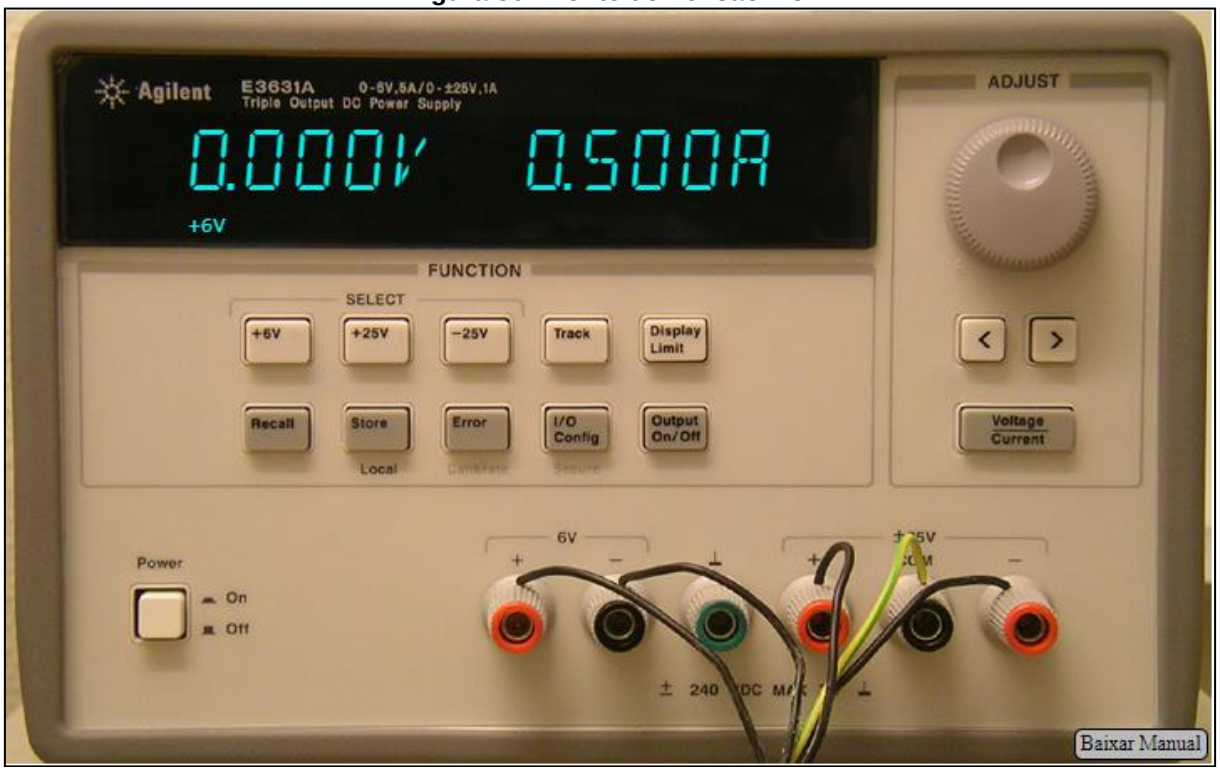
Fonte: adaptado de TAWFIK, 2016

Figura 29 - Bancada de trabalho virtual



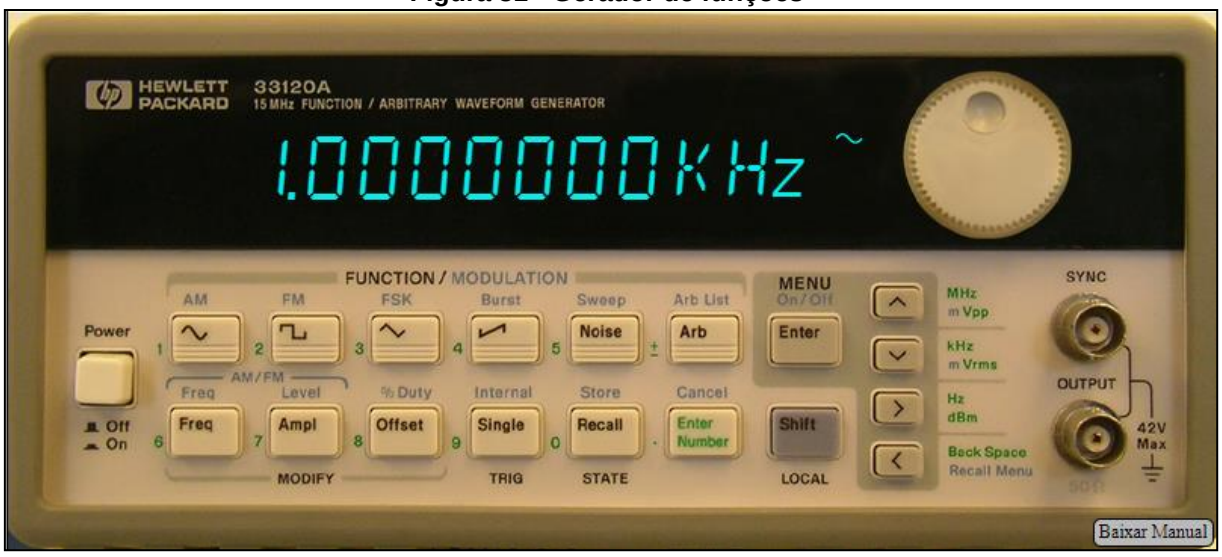
Fonte: elaboração própria (2018)

Figura 30 - Fonte de Tensão DC



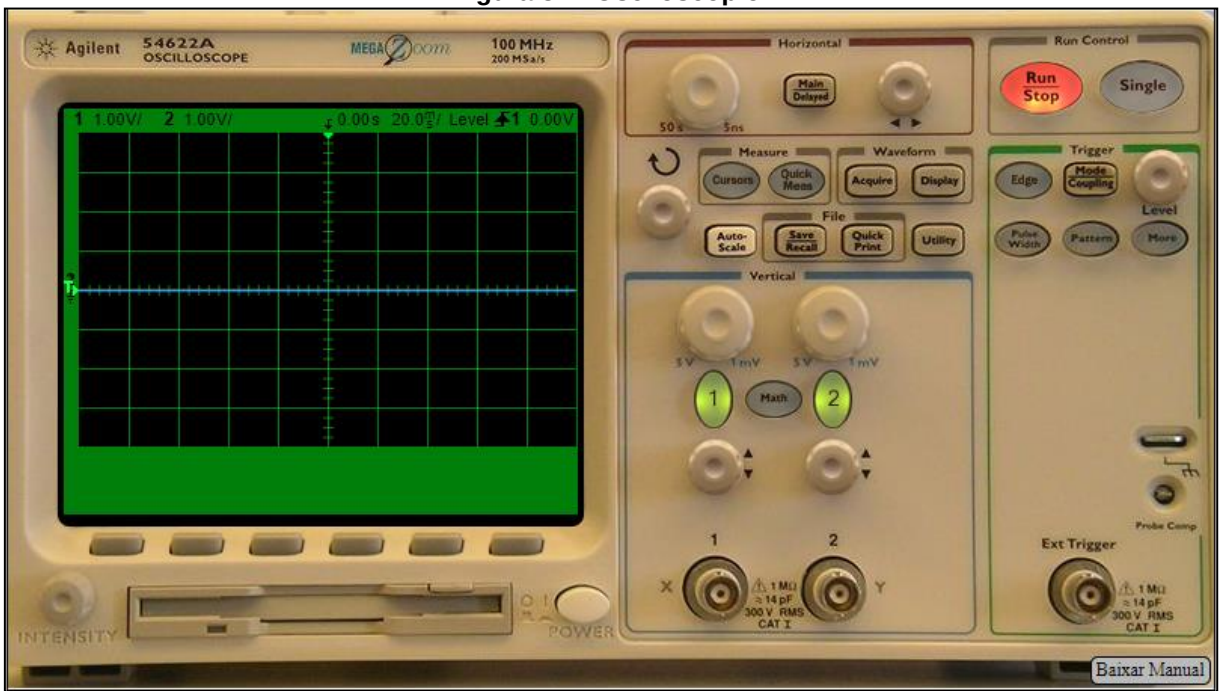
Fonte: elaboração própria (2018)

Figura 31 - Gerador de funções



Fonte: elaboração própria (2018)

Figura 32 - Osciloscópio

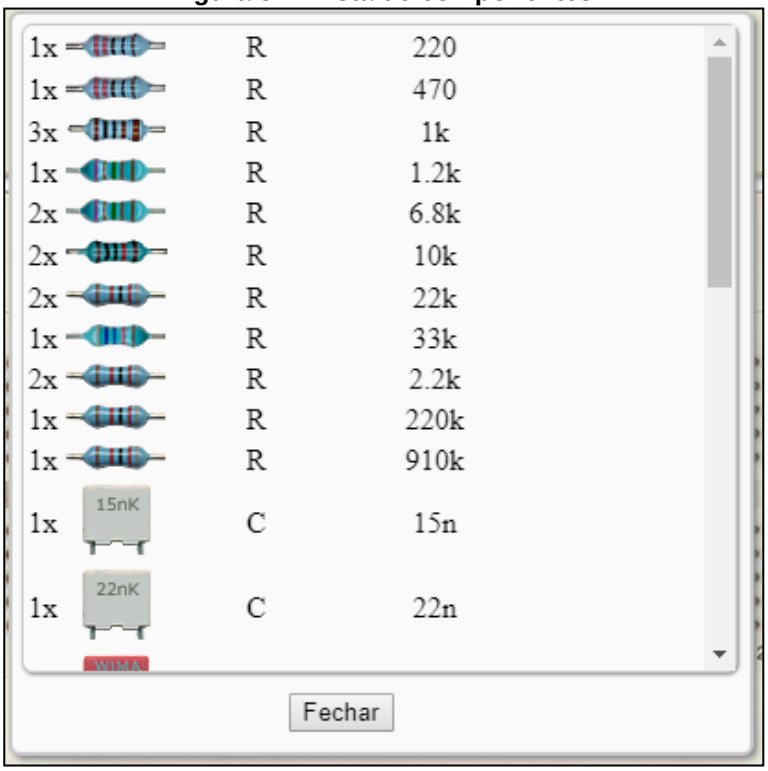


Fonte: elaboração própria (2018)

Figura 33 - Multímetro digital

Fonte: elaboração própria (2018)

Figura 34 - Lista de componentes



Fonte: elaboração própria (2018)

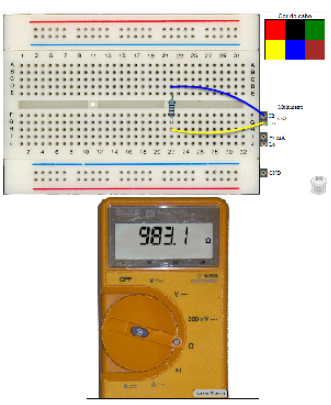
Os componentes e instrumentos disponíveis podem ser limitados de acordo com os privilégios da conta. Um professor pode, por exemplo, disponibilizar somente determinados resistores e habilitar somente o uso do multímetro para um determinado experimento.

Toda manipulação na bancada virtual é executada no computador do próprio usuário, sem ocupar o servidor. Somente quando o usuário clicar em ‘Realizar Medição’ haverá comunicação com o servidor. Quando o usuário realizar uma medição o navegador enviará para o servidor um arquivo *XML* descrevendo todas as conexões e configurações dos instrumentos. Cada instrumento possui uma série de variáveis que serão anexadas ao *XML*, formando um único arquivo que será enviado por *HTTP* via *TCP/IP*. Um exemplo de arquivo *XML* equivalente a um experimento e as configurações dos instrumentos pode ser observado na Figura 35.

No *XML* exibido na Figura 35 é possível observar os elementos que serão enviados na requisição. Entre os marcadores <*circuit_list*> são descritas as conexões dos componentes na tela, em <*multimeter*> são descritas as configurações do multímetro digital. Estes valores serão usados na configuração do *Equipment Server*.

A resposta do servidor retorna também na forma de um arquivo *XML*. A resposta ao experimento da Figura 35 pode ser observada na Figura 36.

Figura 35 - *XML* equivalente a um experimento



```
<protocol version="1.3">
    <request sessionkey="96f94a8243b8dada8284331ccb72f606">
        <circuit>
            <circuitlist>
                W_X DMM_1_2 F22
                W_X A22 DMM_1_1
                R_X A22 F22 1k
                DMM_1 DMM_1_1 DMM_1_2
            </circuitlist>
        </circuit>
        <multimeter id="1">
            <dmm_function value="resistance"></dmm_function>
            <dmm_resolution value="3.5"></dmm_resolution>
            <dmm_range value="-1"></dmm_range>
            <dmm_autozero value="1"></dmm_autozero>
        </multimeter>
    </request>
</protocol>
```

Fonte: elaboração própria (2018)

Figura 36 - Resposta do servidor em formato *XML*

```
<protocol version="1.3">
<response>
<multimeter id="1">
<dmm_function value="resistance"/>
<dmm_resolution value="3.5"/>
<dmm_range value="-1.000000e+000"/>
<dmm_result value="9.831235e+002"/>
</multimeter>
</response>
</protocol>
```

Fonte: elaboração própria (2018)

É possível observar no arquivo de resposta (Figura 36) o campo '*dmm_result*'. Este campo armazena o resultado da medição e é o valor que será formatado e exibido no *display* do instrumento virtual.

Conhecendo o funcionamento de todos os blocos, como é feita a comunicação entre eles e a estrutura dos arquivos, é possível fazer alterações com o objetivo de personalizar o sistema.

Assim, as próximas etapas do trabalho são: descrever a adição de um componente que não está previsto no sistema, o diodo, modificar o multímetro padrão para possibilitar a medição do diodo e inserir um novo multímetro no sistema.

2.4 Adicionando o componente diodo

O diodo é um dispositivo semicondutor que tem como principal característica permitir que a corrente elétrica flua em um sentido e seja bloqueada no sentido reverso. Devido a sua simplicidade e seu vasto uso em projetos eletrônicos, o diodo é um dos primeiros elementos que os alunos da área de eletrônica têm contato.

Para habilitar o uso do diodo no *VISIR* e possibilitar novos experimentos, foi feito um estudo no código-fonte do laboratório buscando compreender o funcionamento dos componentes e instrumentos em todos os servidores.

Além de inserir o componente para uso, seria necessário também habilitar os instrumentos para operarem com o dispositivo, como por exemplo, usar a função ‘diodo’ no multímetro digital.

Assim, os passos seguidos para permitir o uso do diodo no ambiente virtual foram:

a) configurar o multímetro digital para habilitar a função ‘diodo’ no *Equipment server*;
b) configurar o *Measurement server* para tratar circuitos com o componente diodo;
c) configurar o *Web Server* para disponibilizar o componente para o usuário;
d) integrar todas as modificações e garantir que o sistema consiga tratar o novo componente.

2.4.1 Configurando o *Equipment Server*

Como visto anteriormente, o *Equipment Server* é um servidor escrito em *LabView*, uma linguagem de programação gráfica da *National Instruments*. Para cada placa de instrumento existe um código-fonte que executa as funções especificas de cada módulo.

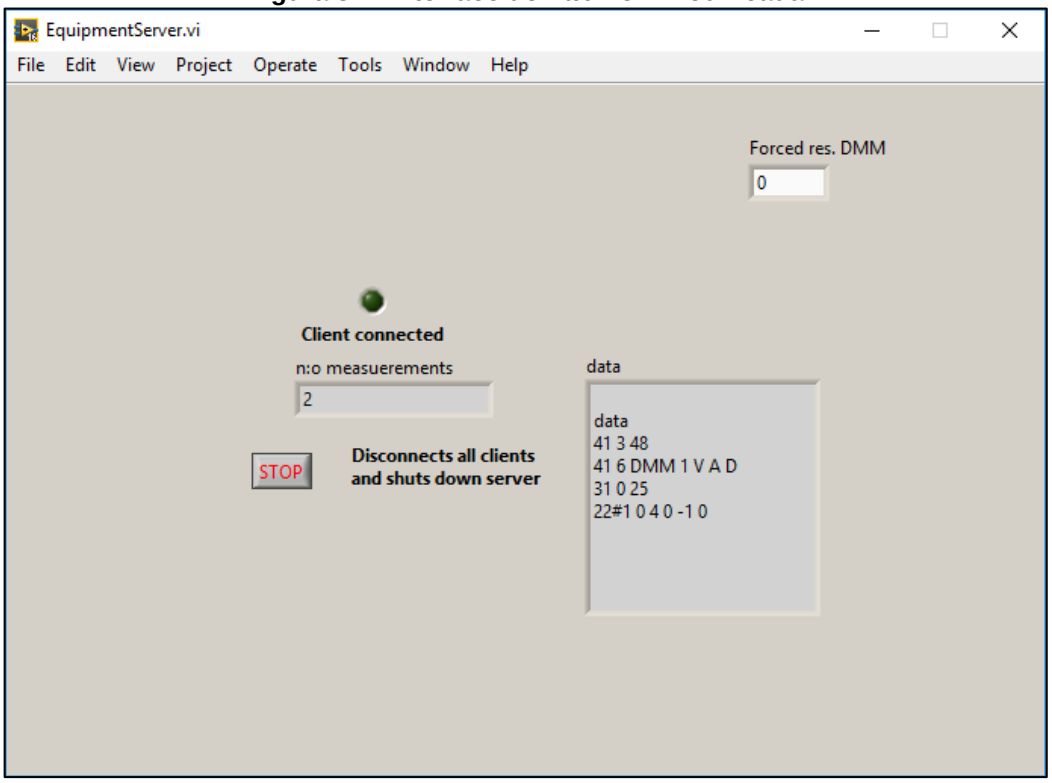
Uma das funções necessárias para se trabalhar com o diodo é a leitura da queda de tensão no componente utilizando o multímetro digital.

Resgatando os parâmetros do multímetro listados na Tabela 5 pode-se observar que o *Equipment Server* já prevê a medição de diodos, bastando passar os parâmetros corretos para o servidor.

Como o instrumento já estava configurado para operar com o teste de diodo, não foi necessário fazer nenhuma alteração no código-fonte do multímetro digital neste servidor.

Mesmo não sendo necessária nenhuma intervenção para o funcionamento do multímetro, foi adicionado à interface do *LabView* (Figura 37) um bloco que permitisse a visualização das requisições que chegavam ao *equipment server*. Esta visualização se mostrou bastante útil para a depuração do sistema.

Figura 37 - Interface do *LabView* modificada



Fonte: elaboração própria (2019)

Ainda no *Equipment Server* era necessário habilitar a matriz de comutação para reconhecer e operar com o diodo.

O arquivo '*EquipmentServer.ini*' armazena informações relacionadas aos instrumentos instalados no servidor e os componentes das matrizes. Por convenção, o diodo foi representado pela letra 'D' e adicionado ao parâmetro '*component type*'. Além do texto de identificação do componente, é necessário inserir, também, o número de terminais deste, como no exemplo a seguir:

Component type = R:2, C:2 ,..., POT:3,D:2

Onde 'R' identifica resistores, 'C', capacitores, 'POT', potenciômetros e 'D', o diodo. Além destes componentes do exemplo, este parâmetro apresenta diversos outros componentes. Quando um novo componente for inserido deve-se verificar este parâmetro e, se necessário, adicionar o novo componente.

Quando um diodo for instalado na matriz de comutação, será necessário adicioná-lo no arquivo *component.list*, inserindo o tipo, número da placa, número do relé, os nós onde o componente pode ser conectado e seu valor. Como exemplo, um diodo 1N4007 instalado entre os nós D e F, no relé 10 da placa 2 será escrito da seguinte maneira na *component.list*:

D_2_10 D F 1N4007

Com estas alterações o *Equipment Server* está pronto para receber instruções relacionadas ao diodo e também pode-se instalar o dispositivo na matriz de comutação. Ainda é necessário configurar a interface do usuário para que se possa utilizar o diodo no navegador e configurar o *Measurement Server* para manipular as mensagens com conteúdo relacionado ao diodo.

2.4.2 Configurando o *Measurement Server*

Para inserir o novo componente no *Measurement Server* é necessário adicioná-lo no arquivo '*componente.types*'. Para isto, basta acrescentar uma nova linha com o identificador D do diodo, o número de terminais e, se for o caso, adicionar as *flags* especiais. As *flags* são representados por letras e podem ser três tipos:

- T – identifica componentes que podem ser rotacionados,
- I – ignora o valor do componente,
- S – para componentes com valores especiais.

Para o diodo, a única que se aplica é a *flag* 'T'.

Assim, o diodo no arquivo '*component.types*' fica:

D 2 T

Com isso o *Measurement Server* pode manipular mensagens que usam o diodo, com o identificador D.

Para que o dispositivo seja disponibilizado para o usuário, ainda é necessário configurar uma *maxlist* que permita seu uso. Para configurar a *maxlist* o componente deve ser instalado no equipamento, usando algum dos relés disponíveis nas matrizes de comutação. Sabendo o nome ou valor do componente, número da placa e o número do relé onde o dispositivo está instalado, pode-se escrever a *maxlist*. Por exemplo, para permitir o uso do diodo 1N4007 entre os nós D e F, deverá ser escrito o seguinte trecho em uma *maxlist*:

D_D1 D F 1N4007

Assim o dispositivo fica disponível para o usuário dentro do contexto da *maxlist* onde for inserido.

2.4.3 Configurando o *Web Server*

A última etapa de configuração dos servidores para permitir o uso do diodo é configurar o servidor *web* para apresentar o diodo como um componente utilizável e permitir que o usuário construa circuitos usando o dispositivo.

Para que o diodo apareça como um elemento que pode ser usado nos experimentos, é necessário adicionar uma imagem que represente o dispositivo. Esta imagem deve ter fundo transparente e seu tamanho deve ser compatível com a escala dos demais componentes. Após isso, deve-se adicionar o diodo no arquivo '*library.xml*'. Neste arquivo são inseridos os componentes que estarão disponíveis na interface *web*. Neste *XML* também são descritos os parâmetros do componente utilizados nas requisições do *Web Server* e alguns parâmetros relacionados à apresentação e manipulação do componente no navegador. A Figura 38 mostra o trecho do arquivo *library.xml* onde é inserido um diodo.

Na Figura 38 podem ser observadas alguns marcadores. Os campos da <*component* > identificam o tipo de componente e o valor. Os campos dentro de <*rotation* > definem a imagem associada ao componente (*image*), os pontos centrais x e y (ox e oy) da imagem e a rotação (*rot*). Os marcadores <*pins* > e <*pin* > definem o ponto de contato do elemento com as linhas e colunas da matriz. Os valores de x e y estão em pixels e são relacionados aos valores de x e y da imagem do componente. É baseado na posição desses pontos sobre a imagem da matriz de contatos que o servidor define as conexões de todos os elementos utilizados nos experimentos.

Figura 38 – Descrição do diodo na *library.xml*

```
<component type="D" value="1N4007" pins="2" qtde="1x">
    <rotations>
        <rotation ox="-27" oy ="-7" image="d_1n4007.png" rot="0">
            <pins><pin x="26" y="0" /><pin x="-26"  y="0" /></pins>
        </rotation>
        <rotation ox="-7" oy ="-27" image="d_1n4007.png" rot="90">
            <pins><pin x="0" y="26" /><pin x="0" y="-26" /></pins>
        </rotation>
        <rotation ox="-27" oy ="-7" image="d_1n4007.png" rot="180">
            <pins><pin x="-26" y="0" /><pin x="26" y="0" /></pins>
        </rotation>
        <rotation ox="-7" oy ="-27" image="d_1n4007.png" rot="270">
            <pins><pin x="0" y="-26" /><pin x="0" y="26" /></pins>
        </rotation>
    </rotations>
</component>
```

Fonte: elaboração própria (2018)

Após inserir a imagem e editar o arquivo *library.xml*, o servidor permitirá que o usuário utilize o componente diodo nos experimentos.

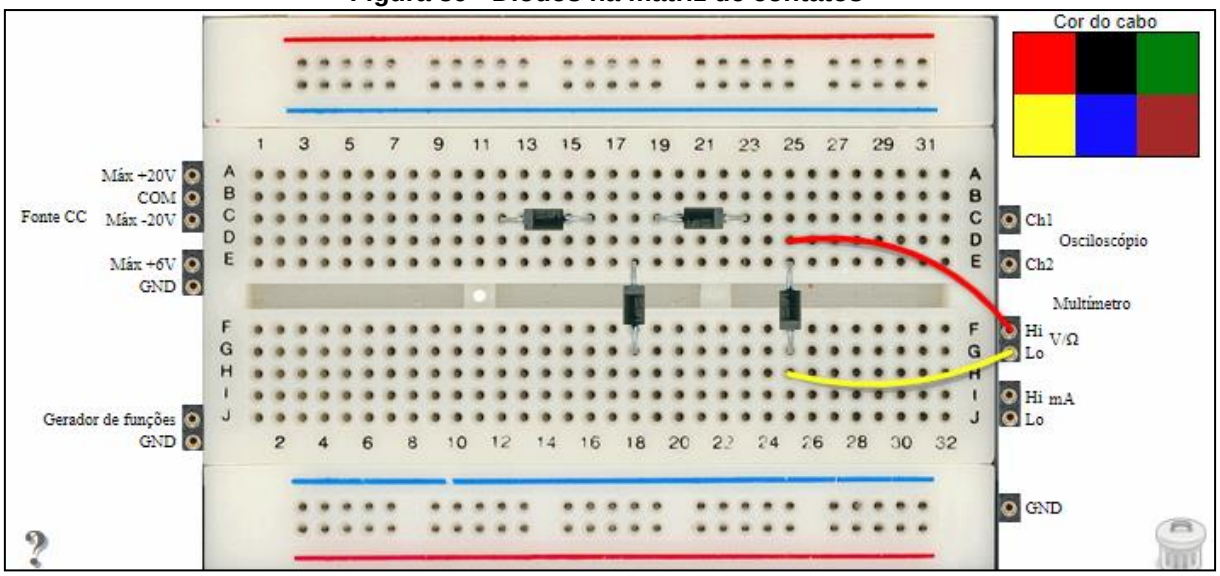
O multímetro padrão da interface do *VISIR* não prevê o uso do teste de diodo. Por isso, é necessário alterar o código-fonte para adicionar esta função ao instrumento. Esta alteração é feita modificando o arquivo '*flukemultimeter.js*'.

A posição do cursor equivalente ao teste de diodo no multímetro padrão do *VISIR* é definida como '*off*', ou desligado. Para habilitar o teste, a posição foi alterada para '*diode*' – valor passado ao *Equipment Server* – e, para exibir o valor corretamente, a variável '*range*' é configurada com o valor '1'.

A Figura 39 mostra o componente posicionado com rotações diferentes – habilitadas pelo marcador <*rotation rot*> – e a Figura 40 mostra o resultado da medição da queda de tensão no diodo, usando o multímetro digital.

Com esse teste foi confirmado que é possível inserir novos componente no *VISIR* e que os instrumentos que já estão instalados podem ser alterados. O diodo e o teste de diodos no multímetro padrão, antes inexistentes no sistema, agora são elementos que podem ser usados no *VISIR* instalado no DAELN.

Figura 39 - Diodos na matriz de contatos



Fonte: elaboração própria (2018)

Figura 40 - Medição da queda de tensão no diodo

Fonte: elaboração própria (2018)

2.5 Adicionando um novo multímetro – Minipa ET-2042D

Uma das vantagens de um laboratório virtual em relação a um simulador é que ele apresenta um ambiente similar ao que o usuário encontra em uma bancada real, onde os instrumentos apresentam a mesma interface de um equipamento real. Embora o *VISIR* já apresente os instrumentos básicos de uma bancada para estudos de eletrônica, estes instrumentos não são os mesmos disponíveis nos laboratórios do DAELN. Seria interessante se, para um melhor aproveitamento do laboratório remoto, os equipamentos apresentassem uma interface similar ao que é encontrado em salas de aula.

Para verificar a possibilidade de inserir e disponibilizar novos instrumentos para os usuários do *VISIR* foi escolhido o multímetro digital Minipa ET-2042D (Figura 41), que é um instrumento bastante utilizado pelos alunos do DAELN.

Figura 41 - Multímetro Minipa ET-2042D

Fonte: ELETROPEÇAS (2019)

Este multímetro apresenta um *display* de 3,5 dígitos e chave seletora com 30 posições, que permitem medir: resistência, tensão AC e DC, corrente AC e DC, capacitância, frequência, temperatura, hFE, teste de linha viva e teste de diodo.

2.5.1 Revisão do *Equipment Server* e *Measurement Server*

Verificando o código-fonte do multímetro do *VISIR* e a Tabela 5, pode-se observar que as funções necessárias para replicar o multímetro ET-2042D já estavam implementadas e para acessá-las bastava passar os parâmetros corretos nas requisições ao *Equipment Server*. Por isso, não foi necessário fazer nenhuma alteração no *Equipment Server*.

Também não foi necessária nenhuma intervenção no *Measurement Server*. Este servidor é responsável por receber e encaminhar as requisições vindas do *Web Server* e não faz distinção de qual instrumento foi selecionado pelo usuário.

2.5.2 Configurando o *Web Server*

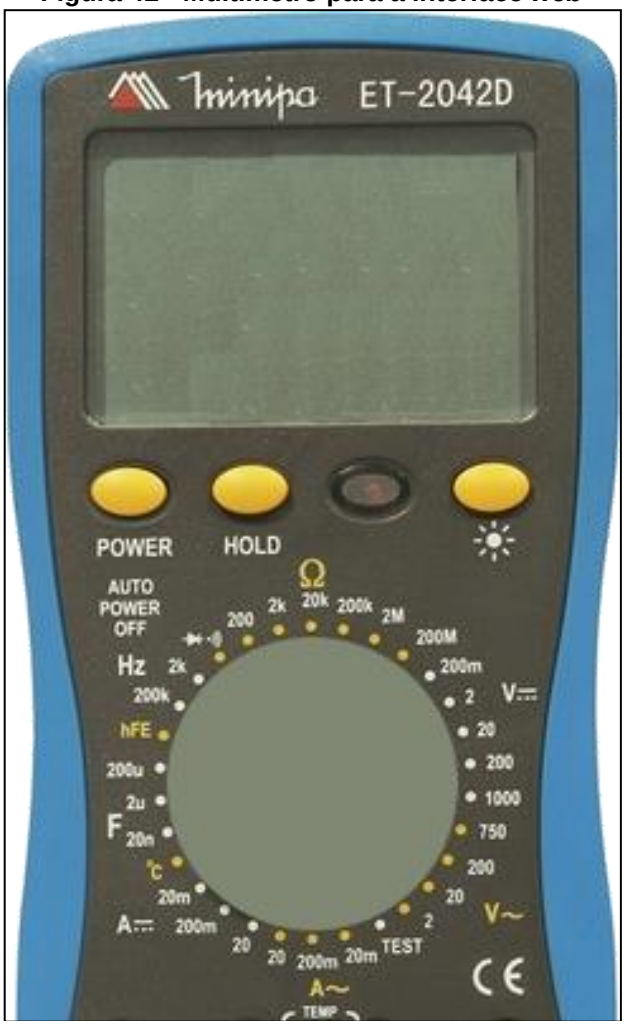
Dentre todos os servidores do laboratório virtual, o *Web Server* foi o que sofreu o maior número de alterações.

Neste servidor é criada a interface onde o usuário interage com o sistema e configuradas as mensagens que serão passadas aos demais servidores por requisições dos usuários. Para isso, são usadas as linguagens *HTML*, *CSS* e *JavaScript*.

Para criar o multímetro, foram usadas fotos de um multímetro real. Dessas imagens foram separados o corpo do instrumento e a chave seletora, que serão tratados como imagens separadas.

O corpo do instrumento (Figura 42) é uma imagem estática que serve como grade para os demais elementos visuais, como display e chave seletora.

Figura 42 - Multímetro para a interface *web*



Fonte: elaboração própria (2018)

O display onde são mostrados os resultados das medições é um campo de texto em *HTML*, com propriedades configuradas para imitar o *display* do multímetro real.

A chave seletora (Figura 43) é uma imagem de tamanho e posição fixos, mas que é possível rotacionar através do uso de funções em *JavaScript*. Baseado no ângulo de rotação da imagem é possível determinar o modo do multímetro escolhido pelo usuário.

Figura 43 - Chave seletora do multímetro

Fonte: elaboração própria (2018)

Usando o multímetro padrão já instalado no *VISIR* como referência, foram criados os arquivos *JavaScript* e *CSS* do novo multímetro. O *HTML* do instrumento é armazenado como uma variável - *var* tpl - dentro do arquivo '*minipamultimeter.js*', por isso não existe um arquivo com a extensão '*.html*' para o instrumento. Nessa estrutura *HTML* armazenada na variável são definidos alguns marcadores *HTML*, com classes e identificadores próprios, que serão usados para manipular os elementos no navegador. É neste *HTML* também que são referenciadas as imagens que serão carregadas no navegador e o *link* para o manual de instruções do instrumento.

No arquivo '*minipamultimeter.css*' (Apêndice A) são definidas as propriedades visuais do multímetro. Configurando as propriedades vinculadas às classes ou identificadores do *HTML*, são definidos o tipo e tamanho de fontes do *display* e a visibilidade e posição de todos os elementos da página. Usando essas propriedades foi possível sobrepor a imagem da chave seletora e do texto do *display* sobre a imagem do corpo do instrumento. Também foi configurada a visibilidade de elementos que só devem ser visíveis em determinados casos, como o sinal 'AC' quando forem medidos sinais em corrente alternada ou o sinal de negativo (-) que só deve ser visível quando a leitura apresentar valores negativos.

O documento ‘*minipamultimeter.js*’ é o principal arquivo do instrumento. Além de carregar a variável que armazena o *HTML* ele também é responsável pela lógica e interação da página.

Neste arquivo há uma função que verifica o ângulo de rotação da imagem da chave seletora e associa este ângulo a um dos 30 modos do multímetro. Sabendo o modo selecionado, pode-se definir os valores das variáveis que serão passados aos demais servidores quando o usuário fizer uma requisição.

Para definir os parâmetros de cada um dos 30 modos possíveis, foi criado uma estrutura *switch-case* onde para cada posição da chave seletora será atribuído um valor para as variáveis *range* e *function* – através da função *SetMode(mode)* -, que serão anexadas ao arquivo *XML* que será passado aos demais servidores. As variáveis *resolution* e *autozero* podem ter valores iguais para todas as opções. Isto não terá impacto no funcionamento do instrumento. Dentro da estrutura *switch-case* ainda são reconfigurados alguns elementos visuais, como os sinais de AC ou negativo.

A Tabela 8 associa a posição da chave seletora com o ângulo da imagem e os valores que serão armazenados nas variáveis *mode* e *range*.

Com isso, o instrumento está configurado para enviar todas as requisições necessárias para fazer quase todos os tipos de medição que o multímetro ET-2042D permite. Por uma limitação de h*ardware* foram deixadas de fora as medições de teste de linha viva, temperatura e hFE.

Após fazer uma requisição, o usuário recebe uma resposta do servidor em um arquivo *XML*. O *Web Server* deve tratar esse arquivo e extrair dele as informações relevantes para a exibição dos resultados da medição. Retomando a Tabela 6, pode-se observar um parâmetro chamado ‘*result*’. Esse parâmetro armazena o resultado da medição feita pelo *Equipment Server*.

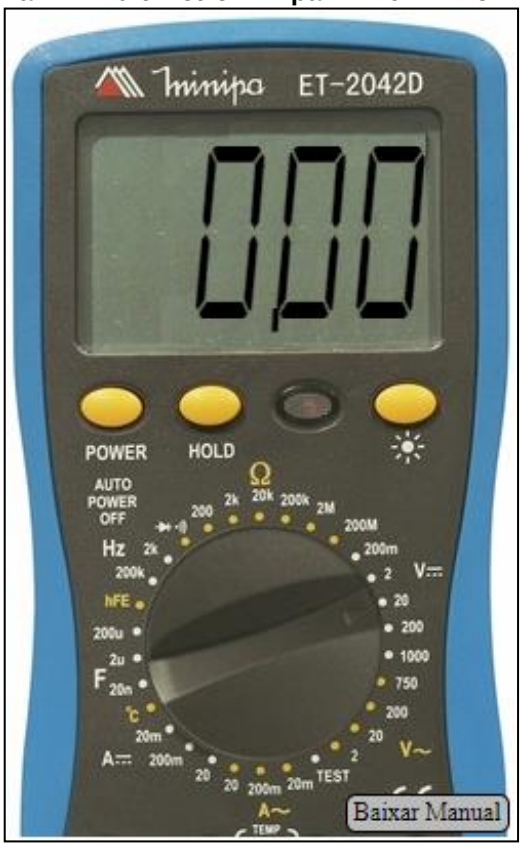
Este resultado deve ser colocado no display do instrumento respeitando o formato do *display* original. Resgatando a Figura 36, pode-se observar que o valor armazenado na variável *dmm_result* – 9,831235e+002 – não está num formato semelhante ao que pode ser exibido no *display* do multímetro. Para tratar isso, foi criada uma segunda estrutura *switch-case* que, baseada no parâmetro *range*, adequa o resultado para um formato que respeite o *display* original, considerando o número máximo que pode ser exibido e a posição do dígito decimal. Assim o resultado final da interface do multímetro ET-2042D ficou como ilustrado na Figura 44.

Tabela 8 - Opções da chave seletora

Chave seletora	Ângulo da imagem	*mode*	*range*
200	246	*resistance*	200
2k	258	*resistance*	2×10^3
20k	270	*resistance*	20×10^3
200k	282	*resistance*	200×10^3
2M	294	*resistance*	2×10^6
200M	306	*resistance*	200×10^6
200m	318	*dc volts*	2×10^{-3}
2	330	*dc volts*	2
20	342	*dc volts*	20
200	354	*dc volts*	200
1000	6	*dc volts*	1000
750	18	*ac volts*	750
200	30	*ac volts*	200
20	42	*ac volts*	20
2	54	*ac volts*	2
TEST	66	*off*	
20m	78	*ac current*	20×10^{-3}
200m	90	*ac current*	200×10^{-3}
20	102	*ac current*	20
20	114	*dc current*	20
200m	126	*dc current*	200×10^{-3}
20m	138	*dc current*	20×10^{-3}
ºC	150	*off*	
20n	162	*capacitance*	20×10^{-9}
2u	174	*capacitance*	2×10^{-6}
200u	186	*capacitance*	200×10^{-6}
hFE	198	*off*	
200k	210	*frequency*	200×10^3
2k	222	*frequency*	2×10^3
DIODE	234	*diode*	1

Fonte: elaboração própria (2018)

Figura 44 - Multímetro Minipa ET-2042D no *VISIR*



Fonte: elaboração própria (2018)

Durante a conversão, parte do resultado é truncado e perde-se a precisão do número. Essa perda não tem impacto significativo no valor medido, considerando o número de casas decimais do instrumento. A Figura 45 ilustra este comportamento do display mostrando a mesma medição de um resistor de 10 kΩ usando escalas diferentes. Nas escalas de 200 Ω e 2 kΩ o instrumento mostra ‘1. ‘, indicando que houve estouro na escala selecionada. Em 20 kΩ e 200 kΩ a leitura apresentada está adequada. Em 2 MΩ, a leitura mostra alguma imprecisão e em 200 MΩ a leitura está errada por ter sido escolhida uma escala inadequada. Este comportamento é muito semelhante ao obtido quando as medições são feitas usando o multímetro real, como pode ser visto na Figura 46.

Figura 45 - Medição de resistor de 10KΩ em diferentes escalas – VISIR

Fonte: elaboração própria (2018)

Figura 46 - Medição de resistor de 10kΩ em diferentes escalas – instrumento real

Fonte: elaboração própria (2018)

Os outros modos do multímetro apresentaram o comportamento esperado. A Figura 47 apresenta a resposta do multímetro para medições de tensão DC, tensão AC, corrente DC, corrente AC, capacitância, frequência e teste de diodo. As configurações dos instrumentos para cada medição são:

a) tensão DC: +10 V;
b) tensão DC: -5 V;
c) tensão AC: 10 V_{pp} (3,535 V_{rms}), f = 500 Hz;
d) corrente DC: +10 V, R = 1 KΩ;
e) corrente DC: -5 V, R = 1 KΩ;
f) corrente AC: 10 V_{pp} (3,535 V_{rms}), R = 1 KΩ, f = 500 Hz;
g) capacitância: C = 0,1 uF;
h) frequência: 10 V_{pp} (3,535 V_{rms}), f = 500 Hz;
i) diodo: D = 1N4007.

Figura 47 - Medições com o multímetro ET-2042D

Da esquerda para direita, de cima para baixo: a) tensão DC +10 V, b)tensão DC -5 V, c) tensão AC, d) corrente DC +10 V, e) corrente DC -5 V, f) corrente AC, g) capacitância, h) frequência e i) teste de diodo.
Fonte: elaboração própria (2018)

Com estes resultados pode-se afirmar que é possível adicionar novos instrumentos no laboratório remoto. A performance obtida com o multímetro virtual é adequada para se validar a proposta de inserir um novo instrumento no sistema.

3 CONCLUSÃO

O trabalho desenvolvido na plataforma *VISIR* trouxe uma oportunidade única aos envolvidos no projeto. Considerando a pouca quantidade de informações disponíveis relacionadas à tecnologia do laboratório, pode-se considerar que boa parte do trabalho foi um exercício de engenharia reversa, onde, partindo de um ambiente funcionando corretamente, fez-se um movimento de fora pra dentro, com o intuito de compreender e se apropriar da tecnologia.

Quando o laboratório foi instalado no IFSC, os envolvidos ainda desconheciam as possibilidades de intervir no laboratório, mas a possibilidade de alterar o ambiente, principalmente os instrumentos na interface *web*, já era cogitada.

Um dos objetivos deste trabalho era compreender o funcionamento do *VISIR*, tanto *software* quanto *hardware*, e com a revisão da bibliografia e análise do funcionamento de todos os servidores, pode-se criar material de apoio para dar suporte aos outros objetivos e, talvez, para futuros projetos.

Também foi proposta a inserção de um novo componente e um novo instrumento no laboratório. Com a adição do diodo e do multímetro Minipa ET-2042D e seu correto funcionamento, foi confirmado que estas alterações são possíveis e viáveis. A documentação gerada para descrever como adicionar novos elementos pode servir de referência para modificar os demais instrumentos ou adicionar novo componentes.

As alterações descritas neste texto foram aplicadas ao laboratório instalado no IFSC e, até o encerramento deste trabalho, não foram reportados defeitos críticos. O novo multímetro foi, inclusive, adotado como instrumento padrão para o sistema.

Assim, acredita-se que os objetivos do trabalho foram alcançados com êxito.

Como sugestão de futuros trabalhos, pode-se atualizar toda a interface *web*, construindo páginas responsivas e usando ferramentas *web* mais modernas. Pode-se, também, adicionar novas fontes de tensão, gerador de funções e osciloscópio para que o laboratório remoto apresente os mesmos instrumentos usados em laboratórios reais.

REFERÊNCIAS

ABED – Associação Brasileira de Educação a Distância. **Censo EAD.BR**: Relatório Analítico da Aprendizagem a Distância no Brasil 2015. Curitiba: InterSaberes, 2016. Disponível em: <http://www.abed.org.br/site/pt/midiateca/censo_ead/1395/2016/09/censoead.br_-_2015/2016>. Acesso em: 24 de abril de 2017.

ALVES, Gustavo R. et al. **Spreading remote lab usage a system – a community – a federation.** International Conference of the Portuguese Society for Engineering Education (CISPEE). 2016, Portugal: IEEE, 2016.

Aprender PHP **Operadores no PHP**. 2010. Disponível em: <http://aprenderphp.com.br/artigo/operadores-no-php/>. Acesso em: 16 de junho de 2019

BRASIL. **Lei n. 9.394**, de 20 de dezembro de 1996. Estabelece as diretrizes e bases da educação nacional. Diário Oficial da União, Brasília, DF, 23 dez. 1996. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/L9394.htm>. Acesso em: 07 de maio 2017.

_____. **Decreto n. 5.622**, de 19 de dezembro de 2005. Regulamenta o art. 80 da Lei n o 9.394, de 20 de dezembro de 1996, que estabelece as diretrizes e bases da educação nacional. Diário oficial da União, Brasília, DF, 20 dez. 2005. Disponível em: <http://portal.mec.gov.br/seed/arquivos/pdf/dec_5622.pdf>. Acesso em: 07 de maio de 2017.

_____. **Portaria n. 4.059 de dezembro de 2004**. Diário Oficial da União, Brasília, DF, 13 dez. 2004. Disponível em: <http://portal.mec.gov.br/sesu/arquivos/pdf/nova/acs_portaria4059.pdf>. Acesso em: 07 de maio de 2017.

_____. **Portaria n. 1.428 de dezembro de 2018**. Diário Oficial da União, Brasília, DF, 31 dez. 2018. Disponível em: <http://www.in.gov.br/materia/-/asset_publisher/Kujrw0TZC2Mb/content/id/57496468/do1-2018-12-31-portaria-n-1-428-de-28-de-dezembro-de-2018-57496251>. Acesso em: 30 de julho de 2019.

PROTOCOL Specification Version 4.1. s.d.

DS, Kiran **TCP 3-Way Handshake**. 2014. Disponível em: <http://kirandsnetworkingandlinux.blogspot.com/2014/02/tcp-3-way-handshake.html>. Acesso em: 15 de junho de 2019

ELETROPEÇAS Comercial Eletrônica **Multímetro Digital Minipa ET-2042D**. 2019. Disponível em < https://www.eletropecas.com/Produto/multimetro-digital-minipa-et-2042d>. Acesso em 30 de julho de 2019.

GUSTAVSSON, Ingvar **VISIR Relay Switching Matrix Version 4.1 – User's Manual**. 2016.

KOZIEROK, Charles M. **The TCP/IP Guide**. 2005. Disponível em: <http://www.tcpipguide.com/>. Acesso em: 30 de novembro de 2018.

LARSEN, Ronald W. **LabVIEW for Engineers.** New Jersey: Pearson Education, 2011.

MDN Web Docs **Tecnologia para desenvolvedores web**. 2019a. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/>. Acesso em: 15 de junho de 2019.

_____. **CSS: Cascading Style Sheets**. 2019b. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/CSS>. Acesso em: 15 de junho de 2019.

_____. **JavaScript**. 2019c. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/javascript>. Acesso em: 15 de junho de 2019.

_____. **HTTP**. 2019d. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP>. Acesso em: 15 de junho de 2019.

National Instruments **Conceitos básicos de LabVIEW**. 2019. Disponível em: <http://www.ni.com/getting-started/labview-basics/pt/environment>. Acesso em: 16 de junho de 2019.

RELLE – Ambiente de aprendizagem com Experimentos remotos **Laboratórios**. 2019. Disponível em <http://relle.ufsc.br/labs>. Acesso em 30 de julho de 2019.

SCHLICHTING, Luis. C. M. et al. **Remote Laboratory**: Application and usability. Technologies Applied to Electronics Teaching (TAEE), 2016, Espanha: IEEE, 2016.

TAWFIK, Mohamed et al. **VISIR**: Experiences and Challenges. International Journal of Online Engineering (iJOE). 2012.

TAWFIK, Mohamed **VISIR Installation and Start-up Guide V.1.** Electric, Electronic and Control Engineering Department. 2011, Espanha: UNED, 2011.

The PHP Group **PHP Manual**. 2019. Disponível em: <http://php.net/>. Acesso em: 16 de junho de 2019.

TOTTY, Brian et al. **HTTP: The Definitive Guide.** Sebastopol: O'reilly Media, 2009. 656 p.

UNIVERSIA. **Educação a distância é a que mais cresce no Brasil, segundo censo do MEC**. Notícias Universia Brasil, 2016. Disponível em: <http://noticias.universia.com.br/destaque/noticia/2016/02/22/1136578/educacao-distancia-cresce-brasil-segundo-censo-mec.html>. Acesso em: 24 de abril de 2017.

VISIR **Laboratório remoto**. 2019. Disponível em: <https://visir.florianopolis.ifsc.edu.br/visir/index.php/pt>. Acesso em: 17 de junho de 2019.

W3Schools **XML Tutorial**. 2019. Disponível em: <https://www.w3schools.com/xml/default.asp>. Acesso em: 15 de junho de 2019.

WEBLAB Deusto **Darchimedes-demo**. 2019. Disponível em <https://weblab.deusto.es/weblab/labs/Aquatic%20experiments/darchimedes-demo/>. Acesso em 30 de julho de 2019.

World Wide Web Consortium (W3C) **EXTENSIBLE MARKUP LANGUAGE (XML)**. 2008. Disponível em: <https://www.w3c.org/TR/xml/>. Acesso em: 30 de novembro de 2018.

_____. **HTML 5.2**. 2017. Disponível em: <https://www.w3.org/TR/html/>. Acesso em: 15 de junho de 2019.

APÊNDICE A – CÓDIGO-FONTE MINIPAMULTIMETER.JS

```
"use strict";
var visir = visir || {};

visir.MinipaMultimeter = function(id, elem)
{
	//alert("O instrumento selecionado está em desenvolvimento, por favor selecione
	outro.");
	var dmm = this;

	this.range = 1; // escala selecionada no cursor. Diferente de _range, que configura o
	pxi-4072

	this._elem = elem;
	this._result = "";
	this._resolution = "3.5";
	this._cursor = "";
	this._lastMode = "20kR";

	visir.MinipaMultimeter.parent.constructor.apply(this, arguments)

	var imgbase = "instruments/minipamultimeter/";
	if (visir.BaseLocation) imgbase = visir.BaseLocation + imgbase;

	var tpl = '<div class="minipadmm">\
	<img src="' + imgbase + 'minipa_main.png" width="243" height="400" draggable="false"
	/>\
	<div class="display">\
		<div class="dmm_value" id="value">1.   </div>\
		<div class="mode">AC</div>\
		<div class="signal">-</div>\
	</div>\
	<div class="rot">\
		<div class="top vred">\
			<img src="' + imgbase + 'minipa_knob.png" alt="handle" width="120"
			 height="120" />\
		</div>\
	</div>\
	<div class="manual_link"><a
		href="https://portal.if.usp.br/labdid/sites/portal.if.usp.br.labdid/files/ET-
		2042D-1101-BR.pdf" target="_blank">%downloadManual%</a></div>\
	</div>';

	tpl = tpl.replace(/%downloadManual%/g, visir.Lang.GetMessage("down_man"));
	elem.append(tpl);


	dmm.SetCursor("20kR");

	var top = elem.find(".top");
	setRotation(top, 0);

	var handle = elem.find(".rot");

	function handleTurn(elem, deg)
	{
		if (!visir.Config.Get("readOnly"))
		{
			deg = ( deg - deg % 12 ) + 6;

			if (deg <= 360 || deg >= 0)
			{
				setRotation(top, deg);
```

```
                        var rotFuncMap = {246: "200R", 258: "2kR", 270: "20kR", 282: "200kR"
                                        , 294: "2MR", 306: "200MR", 318: "200mVdc", 330:
                                        "2Vdc", 342: "20Vdc", 354: "200Vdc", 6: "1000Vdc",
                                        18: "750Vac", 30: "200Vac", 42: "20Vac", 54: "2Vac",
                                        66: "test", 78: "20mIac", 90: "200mIac", 102:
                                        "20Iac", 114: "20Idc", 126: "200mIdc", 138:
                                        "20mIdc", 150: "celsius", 162:
                                        "20nF", 174: "2uF", 186: "200uF", 198: "hfe", 210:
                                        "200KHz", 222: "2KHz", 234: "diode"
                        };

                        var cursor = rotFuncMap[deg];
                        dmm.SetCursor(cursor);
                        dmm.UpdateDisplay();

                        return deg;
                  }
            }
            return undefined; // don't set a new rotation
      }

      handle.turnable({ offset: 90, turn: handleTurn })
      dmm.UpdateDisplay();
}

extend(visir.MinipaMultimeter, visir.Multimeter)

visir.MinipaMultimeter.prototype.Test = function() {}

visir.MinipaMultimeter.prototype.SetCursor = function(cursor) {
      this._cursor = cursor;
      this._range = -1;

      this._elem.find(".mode").toggle(false);
      $('.mode, .signal').hide();


      switch(this._cursor){
            // "2kR", 270: "20kR", 282: "200kR" , 294: "2MR", 306: "200MR"
            case "200R":
                  this.range = 200;
                  this.SetMode("resistance");
                  break;
            case "2kR":
                  this.range = 2e3;
                  this.SetMode("resistance");
                  break;
            case "20kR":
                  this.range = 20e3;
                  this.SetMode("resistance");
                  break;
            case "200kR":
                  this.range = 200e3;
                  this.SetMode("resistance");
                  break;
            case "2MR":
                  this.range = 2e6;
                  this.SetMode("resistance");
                  break;
            case "200MR":
                  this.range = 200e6;
                  this.SetMode("resistance");
                  break;

            case "200mVdc":
                  this.range = 2e-3;
```

```
            this.SetMode("dc volts");
            break;
        case "2Vdc":
            this.range = 2;
            this.SetMode("dc volts");
            break;
        case "20Vdc":
            this.range = 20;
            this.SetMode("dc volts");
            break;
        case "200Vdc":
            this.range = 200;
            this.SetMode("dc volts");
            break;
        case "1000Vdc":
            this.range = 1000;
            this.SetMode("dc volts");
            break;

        case "2Vac":
            this._elem.find(".mode").toggle(true);
            $(".mode").show();
            this.range = 2;
            this.SetMode("ac volts");
            break;
        case "20Vac":
            $(".mode").show();
            this.range = 20;
            this.SetMode("ac volts");
            break;
        case "200Vac":
            this.range = 200;
            $(".mode").show();
            this.SetMode("ac volts");
            break;
        case "750Vac":
            $(".mode").show();
            this.range = 750;
            this.SetMode("ac volts");
            break;

        case "test":
        case "celsius":
            this.SetMode("off");
            break;

        case "20mIdc":
            this.range = 20e-3;
            this.SetMode("dc current");
            break;
        case "200mIdc":
            this.range = 200e-3;
            this.SetMode("dc current");
            break;
        case "20Idc":
            this.range = 20;
            this.SetMode("dc current");
            break;

        case "20mIac":
            $(".mode").show();
            this.range = 20e-3;
            this.SetMode("ac current");
            break;
        case "200mIac":
            $(".mode").show();
```

```
                  this.range = 200e-3;
                  this.SetMode("ac current");
                  break;
            case "20Iac":
                  $(".mode").show();
                  this.range = 20;
                  this.SetMode("ac current");
                  break;

            case "20nF":
                  this._range = 100e-9;
                  this.range = 20e-9;
                  this.SetMode("capacitance");
                  break;
/*          case "2uF":
                  this._range = 0.00001;
                  this.range = 2e-3;
                  this.SetMode("inductance");
                  break;
*/
            case "2uF":
                  this._range = 0.00000001;
                  this.range = 2e-6;
                  this.SetMode("capacitance");
                  break;
            case "200uF":
                  this._range = 0.000001;
                  this.range = 200e-6;
                  this.SetMode("capacitance");
                  break;

            case "hfe":
                  this.SetMode("off");
                  break;

            case "2KHz":
                  this.range = 2e3;
                  this.SetMode("frequency");
                  break;
            case "200KHz":
                  this.range = 200e3;
                  this.SetMode("frequency");
                  break;

            case "diode":
                  this._range = 1;
                  this.range = 1;
                  this.SetMode("diode");
                  break;
            // end switch

      }
}

visir.MinipaMultimeter.prototype.UpdateDisplay = function() {
      this._elem.find(".display").toggle(true);
      var out = this.formatResult();

      if (isNaN(out)) {
            out = "1.   ";
      }     else {      }

      if(this._cursor == "test" | this._cursor == "hfe" | this._cursor == "celsius"){
            out = "000";
      }
      if(out == 0){
```

```
            $(".signal").hide();
      }
      this._elem.find(".dmm_value").toggle(true).text(out);

}

visir.MinipaMultimeter.prototype.ReadResponse = function(response) {
      visir.MinipaMultimeter.parent.ReadResponse.apply(this, arguments)
      this.UpdateDisplay();
}

visir.MinipaMultimeter.prototype.formatResult = function(){
      // ajusta o valor que será exibido no display

      var result = this.GetResult();
      var range = this.range;
      result = result.toPrecision(4);

      if(this._lastMeasure != this._mode){
            result = 0;
      }

      if(result < 0){
            $(".signal").show();
      }
      else{
            $(".signal").hide();
      }

      result = Math.abs(result);
      // Se o resultado for maior que a escala, exibe o valor "1.000"
      if(result >= range) {
            $(".signal").hide();
            return "1.   ";
      }

      switch(range){
            case 1:
                  result *= 1000;
                  result = result.toFixed(0) + ".";
                  break;
            case 200e6:
                  result /= 1e6;
                  result = result.toFixed(1);
                  break;
            case 2e6:
                  result /= 1e6;
                  result = result.toFixed(3);
                  break;
            case 200e3:
                  result /= 1000;
                  result = result.toFixed(1);
                  break;
            case 20e3:
                  result /= 1000;
                  result = result.toFixed(2);
                  break;
            case 2e3:
                  result /= 1000;
                  result = result.toFixed(3);
                  break;
            case 1000:
                  result /= 1000;
                  result = result.toFixed(0) + ".";
                  break;
            case 750:
```

```
                result = result.toFixed(0) + ".";
                break;
          case 200:
                result = result.toFixed(1);
                break;
          case 20:
                result = result.toFixed(2);
                break;
          case 2:
                result = result.toFixed(3);
                break;
/*        case 2e-3:
                result /= 1e-3;
                result = result.toFixed(3);
                break;
*/        case 20e-3:
                result /= 1e-3;
                result = result.toFixed(2);
                break;
          case 200e-3:
                result /= 1e-3;
                result = result.toFixed(1);
                break;
          case 200e-6:
                result /= 1e-6;
                result = result.toFixed(1);
                break;
/*        case 20e-6:
                result /= 1e-6;
                result = result.toFixed(2);
                break;
*/        case 2e-6:
                result /= 1e-6;
                result = result.toFixed(3);
                break;
          case 20e-9:
                result /= 1e-9;
                result = result.toFixed(2);
                break;
          default:
                result = result.toFixed(3);
                break;
     }
     result = ("00000" + result).slice(-5);

     if(result[4] === "."){
          result = result.slice(1,4);
     }

     if(result[0] !== "1"){
          result = result.slice(-4);
     }
     return result;
}
```