UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

Vander Elton Maziero

SISTEMA PARA CONTROLE DE CARGA RLC AUTOMATIZADA EMPREGANDO INTERFACE DE COMUNICAÇÃO ETHERNET

Santa Maria, RS
2018

Vander Elton Maziero

SISTEMA PARA CONTROLE DE CARGA RLC AUTOMATIZADA EMPREGANDO INTERFACE DE COMUNICAÇÃO ETHERNET

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Bacharel em Engenharia Elétrica**.

Orientador: Prof. Leandro Michels, Dr.

Santa Maria, RS

2018

Vander Elton Maziero

SISTEMA PARA CONTROLE DE CARGA RLC AUTOMATIZADA EMPREGANDO INTERFACE DE COMUNICAÇÃO ETHERNET

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Bacharel em Engenharia Elétrica**.

Aprovado em 12 de dezembro de 2018:

_____
Prof. Leandro Michels, Dr. (UFSM)
(Orientador)

_____
Prof. Frederico Schaf, Dr. (UFSM)
(Membro Convidado)

_____
Ricardo Jochann Franceschi Bortolini, Me. Eng. (UFSM)
(Membro Convidado)

Santa Maria, RS
2018

AGRADECIMENTOS

Agradeço aos meus pais por me ensinarem o valor e a importância do estudo e da instrução para a minha vida.

Agradeço ao meu orientador, Prof. Leandro Michels, por acreditar no meu trabalho e incentivar para o desenvolvimento do projeto.

Agradeço aos amigos, colegas da Engenharia e INRI, pela convivência que auxiliou na formação de meu caráter como profissional.

Agradeço ao INCT-GD, CAPES, CNPq e FAPERGS pelo apoio financeiro recebido para o desenvolvimento desse trabalho. O presente trabalho foi realizado com apoio do INCT e suas agências financiadoras (CNPq processo 465640/2014-1, CAPES processo n°. 23038.000776/2017-54 e FAPERGS 17/2551-0000517-1).

Seja a diferença que você
quer ver no mundo.

(Mahatma Gandhi)

RESUMO

SISTEMA PARA CONTROLE DE CARGA RLC AUTOMATIZADA EMPREGANDO INTERFACE DE COMUNICAÇÃO ETHERNET

AUTOR: Vander Elton Maziero

ORIENTADOR: Prof. Leandro Michels, Dr.

LUGAR E DATA DA DEFESA: Santa Maria, 12 de Dezembro de 2018

Este trabalho consiste no desenvolvimento de uma interface Ethernet para comunicação entre o firmware de equipamentos microcontrolados e sistemas supervisórios desenvolvidos no software LabVIEW. A interface desenvolvida para a substituição de uma interface USB que opera em conjunto com uma placa de aquisição de dados da National Instruments para automação de um banco de carga resistivo-indutivo-capacitivo (RLC) atualmente em utilização no LabEnsaios/INRI. Essa carga RLC é utilizada em ensaios de inversores para medição do tempo de desconexão dos sistemas anti-ilhamento implementadas em inversores fotovoltaicos conectados à rede baseados na norma ABNT NBR IEC 62116. O sistema implementado consiste em: i) placa de interface desenvolvida para conexão de uma placa de Linux embarcado BeagleBone Black (BBB) a um conjunto endereçável de placas de acionamentos de relés; ii) firmware em Python para controle do BBB; iii) instrumento virtual (VI) para rodar código no software LabVIEW. Resultados experimentais demonstram que o sistema proposto substitui adequadamente o sistema anterior, tendo a vantagem de poder ser empregado em rede associado a múltiplos equipamentos.

Palavras chaves: Redes TCP/IP, Sistemas supervisórios, Sistemas fotovoltaicos conectados à rede, Técnicas anti-ilhamento.

ABSTRACT

AUTOMATED RLC LOAD CONTROL SYSTEM EMPLOYING ETHERNET COMMUNICATION INTERFACE

AUTHOR: Vander Elton Maziero

ADVISOR: Prof. Leandro Michels, PhD.

PLACE AND DATE: Santa Maria, December 12, 2018

This work consists in the development of an Ethernet interface for communication between the firmware of microcontrollable equipment and supervisory systems developed in the LabVIEW software. The interface developed for the replacement of a USB interface that operates in conjunction with a National Instruments data acquisition board for automation of a resistive-capacitive-inductive (RLC) load bank currently in use by LabEnsaios/INRI. This RLC load is used in photovoltaic inverter tests to measure the disconnection time of the anti islanding systems implemented in *on-grid* photovoltaic inverters based on the ABNT NBR IEC 62116 standard. The implemented system consists of: i) interface board designed to connect a BeagleBone Black (BBB) embedded Linux board to an addressable set of relay drive boards; ii) firmware in Python to control the BBB; iii) virtual instrument (VI) to run code in LabVIEW software. Experimental results demonstrate that the proposed system adequately replaces the previous system, having the advantage of being able to be used in network associated to multiple equipments.

Keywords: TCP/IP Network, Supervisory systems, *On-grid* photovoltaic systems, Anti-islanding techniques.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ACK	Acknowledgement
ANEEL	Agência Nacional de Energia Elétrica
API	Application Programming Interface
BBB	BeagleBone Black
CI	Circuito Integrado
CPU	Computador de Placa Única
DAQ	Data Acquisition
eMMC	Embedded Multimedia Card
GD	Geração Distribuída
GPIO	General Purpose Input/Output
IEC	International Electrotechnical Commission
INMETRO	Instituto Nacional de Metrologia
IP	Internet Protocol
JSON	JavaScript Object Notation
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LAN	Local Area Network
MAC	Media Access Control
NBR	Norma Brasileira
PAC	Ponto de Acoplamento Comum
PCI	Placa de Circuito Impresso
RAM	Random Access Memory
RSP	Relay Switch Protocol
SEP	Sistema Elétrico de Potência
SoC	System On a Chip
SPMP	Seguidor de Ponto de Máxima Potência
SSH	Secure Shell
TCP	Transmission Control Protocol
USB	Universal Serial Bus
VHLL	Very High Level Language
VI	Virtual Instrument

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

O crescimento econômico de um país está vinculado ao potencial energético disponível. Nesse cenário, as fontes de energia renováveis tornaram-se uma ótima alternativa, adquirindo destaque nas políticas públicas no setor energético, visando à diminuição da dependência dos derivados de petróleo (TAMANINI, 2017).

O Brasil possui uma vasta gama de recursos naturais que podem ser utilizados para a geração de energia elétrica. A água, utilizada nas hidrelétricas, é o principal. Essa alternativa, entretanto, vem sendo cada vez mais questionada com o passar dos anos, muito em virtude de seu alto custo de implantação acompanhado pelo seu elevado impacto ambiental. Esse contexto permitiu o desenvolvimento e expansão da geração de energia por meio de fontes alternativas, entre elas, a energia solar, obtida através de placas fotovoltaicas (TAMANINI, 2017).

A exploração desse potencial é diretamente dependente do desenvolvimento de tecnologias, que permitam a conversão de energia solar em energia elétrica, de uma forma competitiva com as demais fontes de energia.

Os sistemas de geração de energia fotovoltaicos são divididos em duas categorias: os sistemas conectados à rede (*on-grid*) e os sistemas isolados da rede, ou autônomos (*off-grid*). O foco desse projeto será em sistemas conectados à rede. Apesar da baixa representatividade da energia solar na matriz energética brasileira, sistemas conectados à rede estão em crescente expansão, conforme visto na Figura 1, muito em virtude da aprovação da normativa 482/2012 da ANEEL em 17 de Abril de 2012, e modificada pelas Resoluções Normativas ANEEL no 687/2015 e no 786/2017, que permite aos consumidores instalar painéis fotovoltaicos em suas unidades consumidoras e utilizar o sistema elétrico para injetar o excedente de energia (ANEEL, 2014).

Figura 1 – Potência Instalada (MW) de Geração Distribuída Solar Fotovoltaica no Brasil.

	2012	2013	2014	2015	2016	2017	02 / 2018
Potência Instalada no Ano (MW)	0,4	1,4	2,4	9,5	46,6	117,2	30,6
Potência Instalada Acumulada (MW)	0,0	0,4	1,8	4,2	13,7	60,3	177,4
Total	0,4	1,8	4,2	13,7	60,3	177,4	208,0

Fonte: (SAUAIA, 2018).

Para possibilitar a conexão de centrais fotovoltaicas ao Sistema Interligado Nacional é necessário o uso de inversores, responsáveis por fazer o condicionamento da energia elétrica, visto que, placas solares geram eletricidade em corrente continua com tensão/corrente variáveis com a temperatura e irradiância.

No Brasil, os principais equipamentos fotovoltaicos estão sujeitos à regulamentação que exige a avaliação de um conjunto de requisitos de avaliação da conformidade. A comercialização desses equipamentos somente é possível se os equipamentos foram aprovados em ensaios que avaliam todos os requisitos descritos por essa regulamentação.

1.2 MOTIVAÇÃO

Para atender ao Regulamento de Avaliação de Conformidade do INMETRO que entrou em vigência em 2011, e teve sua última atualização em 2016, no Brasil para inversores fotovoltaicos conectados à rede são necessários entre 76 e 208 ensaios para certificação, sendo este número dependente da faixa de potência, e do número de amostras a ser avaliado de um mesmo modelo. Dessa forma, fica evidente a necessidade de se automatizar os ensaios para reduzir os tempos de ensaio, e

consequentemente os custos de certificação, assim como eliminar os possíveis erros humanos durante esse processo (FIGUEIRA, 2016).

Entre os ensaios que compõem os requisitos de avaliação da conformidade para sistemas e equipamentos para energia fotovoltaica, está o ensaio de anti-ilhamento, disposto na norma ABNT IEC 62116:2012, tendo por objetivo a avaliação do tempo de desconexão do inversor ao ocorrer uma falta na rede. Esse ensaio demanda o uso de uma carga RLC passiva que deve ser ajustada de forma adequada à potência de ensaio. Sendo assim, necessária a associação de resistores, capacitores e indutores, para obter os critérios desejados.

Motivado para tornar este processo, de associação de cargas, mais simples, foi desenvolvido um sistema para determinar a associação de cargas necessárias e fazer a comutação das mesmas, por meio de relés. Um programa para execução dos cálculos das associações adequadas nesse processo foi desenvolvido em LabVIEW. A comunicação desse sistema com os dispositivos externos que controlam as cargas foi projetada para empregar uma placa de aquisição de dados NI USB – 6501, com conexão serial via USB. Contudo, todos os demais equipamentos utilizados no ensaio possuem comunicação por meio de rede Ethernet usando protocolo TCP/IP, o que facilita o acesso desses por meio de outros computadores conectados à rede. Por esse motivo, tendo em vista o desenvolvimento de um laboratório multiusuário que possibilite acesso remoto por diversos computadores a este sistema de ensaio, faz-se necessária a adequação desse sistema a comunicação por meio de rede Ethernet usando protocolo TCP/IP.

1.3 OBJETIVOS

O objetivo deste trabalho é a implementação de um sistema de controle automático por meio de rede Ethernet usando protocolo TCP/IP de uma carga RLC passiva automatizada projetada para a realização de ensaio de anti-ilhamento em inversores fotovoltaicos segundo a norma ABNT NBR IEC 62116:2012. O sistema proposto é constituído de uma placa de controle para monitorar e gerenciar o funcionamento interno do equipamento, um *firmware* em Python implementado em uma placa BeagleBone Black para controle do sistema, e um instrumento virtual (VI) desenvolvido no *software* LabVIEW para execução no computador.

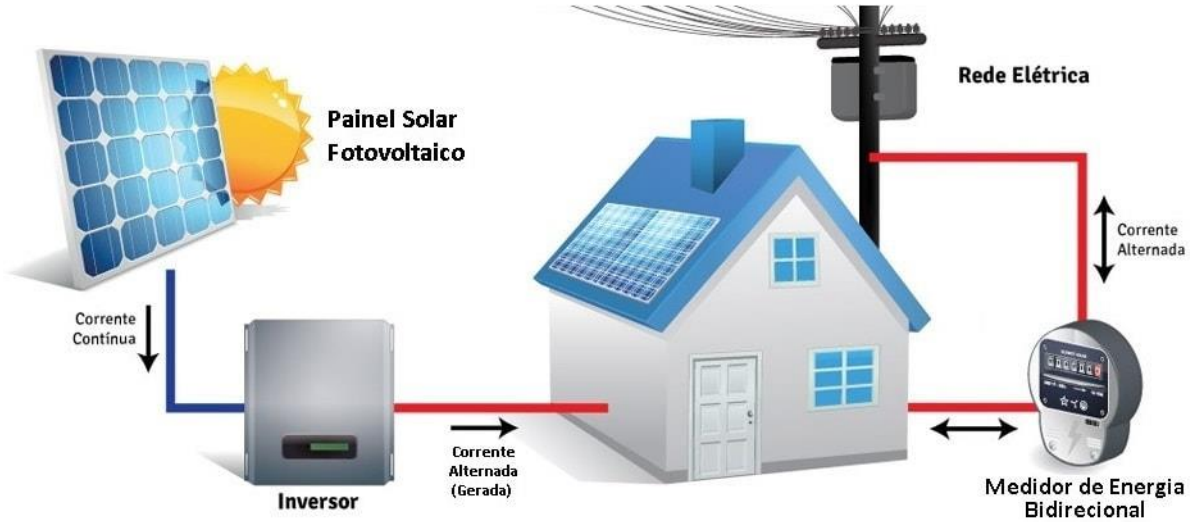
2 FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMAS FOTOVOLTAICOS CONECTADOS À REDE ELÉTRICA

Sistemas fotovoltaicos conectados à rede elétrica, também chamados de sistemas *on-grid*, operam em paralelismo com a rede elétrica. Diferentemente dos sistemas autônomos, os sistemas conectados são empregados em locais já atendidos por energia elétrica. Por não possuírem dispositivos de armazenamento de energia, todo o excedente de energia produzida pelo sistema é injetado na rede elétrica, ou no caso de usinas fotovoltaicas, toda a energia produzida.

Esse sistema é basicamente constituído por um conjunto de módulos fotovoltaicos, que irão gerar a energia, um grupo de inversores, que irão fazer a conversão CC-CA para conexão à rede elétrica, além das caixas de junção e dispositivos de proteção e de medição da energia produzida (TAMANINI, 2017). A Figura 2 ilustra esse tipo de sistema.

Figura 2 – Representação de um sistema fotovoltaico conectado à rede elétrica.



Fonte: (SOLAR, 2017).

A radiação proveniente do Sol incide nos módulos solares, onde ocorre a conversão para energia elétrica. Em seguida, a tensão, produzida em corrente contínua, chegará ao inversor, que fará a conversão e enviará a energia para à rede elétrica.

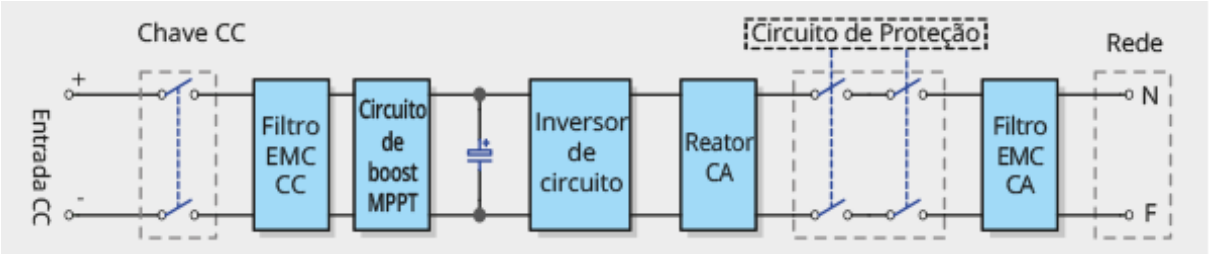
O inversor, além de realizar a conversão CC-CA, tem como função também, controlar a corrente a ser injetada na rede, de modo que tenha o formato senoidal e esteja sincronizada com a frequência da rede, além de atender os requisitos de operação impostos pela concessionária de energia.

2.1.1 Inversores Fotovoltaicos

Como, nos sistemas *on-grid*, a energia elétrica gerada é entregue diretamente à rede, os dispositivos de condicionamento de potência devem se adequar ao modo como a energia está fluindo nas linhas de distribuição, copiando esse padrão e fornecendo o mesmo tipo de grandeza elétrica.

Os inversores fotovoltaicos fazem a conversão da corrente continua, gerada em módulos fotovoltaicos, para corrente alternada, condicionando a conexão destes módulos a rede elétrica, sendo compostos por uma série de blocos, responsáveis por realizar as funções necessárias, conforme Figura 3. Buscando a alta eficiência, esses equipamentos possuem uma série de medidas a serem tomadas, como o seguimento do ponto de máxima potência (SPMP), medidas de segurança para desconexão da rede, mecanismos de anti-ilhamento, entre outras funções (ALMEIDA, 2012).

Figura 3 – Diagrama de inversor utilizado em sistema PV *on-grid*.



Fonte: (JOBESOLAR, 2017).

Para garantir uma uniformidade quanto a esses parâmetros, que um inversor fotovoltaico deve possuir, é necessária uma lista de ensaios a ser realizada. No Brasil estes ensaios são determinados por Regulamentos de Avaliação de Conformidade descritos em Portarias do INMETRO, baseados em normas brasileiras desenvolvidas através da ABNT.

As normatizações e regulamentações criadas, e o crescimento esperado nesse setor, (TIOMNO, 2013) implica em uma necessidade crescente de laboratórios capazes de realizar ensaios de avaliação de conformidade de inversores conectados

à rede elétrica, os quais viabilizam o abastecimento do mercado brasileiro com inversores certificados. Para poder realizar os ensaios com validade para o Programa Brasileiro de Etiquetagem, tais laboratórios devem atender a norma ABNT ISO IEC 17025 e serem reconhecidos pelo INMETRO (INMETRO, 2015).

2.1.2 Regulamentação

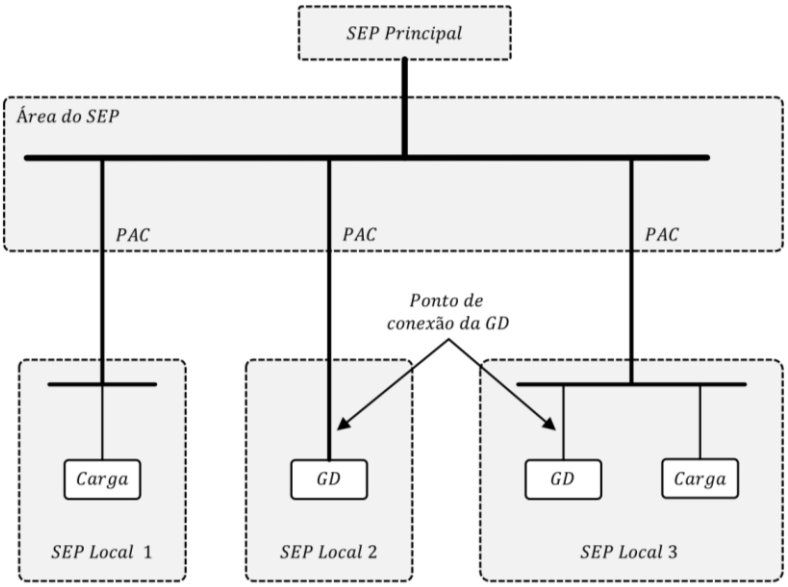
A disseminação da geração distribuída solar fotovoltaica no Brasil teve seu estopim após a publicação da Resolução n° 482, em 2012, pela Agência Nacional de Energia Elétrica (ANEEL). Logo após foi conduzido um processo de normatização dos sistemas fotovoltaicos, em que a Associação Brasileira de Normas Técnicas (ABNT) publicou em 2012 a norma NBR IEC 62116, que estabelece os procedimentos de ensaio para avaliar o desempenho das medidas de prevenção de ilhamento utilizadas em sistemas fotovoltaicos conectados à rede elétrica (ABNT, 2012).

2.1.2.1 ABNT NBR/IEC 62116 - Ensaio anti-ilhamento

Define-se como ilhamento a condição na qual parte da área do SEP permanece energizada através de fontes de geração distribuída, mesmo quando este se encontra eletricamente isolada do restante do sistema. A Figura 4 ilustra a conexão típica dos sistemas de geração distribuída ao sistema elétrico de potência (SILVA, 2016).

O ilhamento não intencional é uma condição indesejada, por apresentar riscos de segurança e, também por gerar mau funcionamento dos equipamentos conectados à rede. De acordo com as normas, os inversores fotovoltaicos devem ser capazes de detectar o ilhamento e cessar sua operação em um curto intervalo de tempo, a partir do instante da formação da ilha, sendo comumente denominada proteção anti-ilhamento.

Figura 4 – Conexão típica de um sistema de geração distribuída ao SEP.

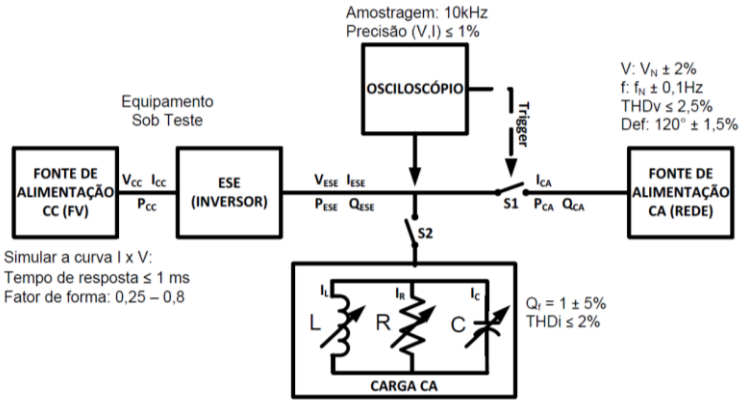


Fonte: Adaptado da IEEE 1547-2003.

A norma ABNT NBR/IEC 62116 é uma tradução da norma IEC de mesma numeração. O texto da norma descreve as características e procedimentos para o teste da função de anti-ilhamento, necessária a todos os inversores conectados à rede elétrica. Na norma são descritos padrões de tempo de desconexão assim como situações de consumo de carga para verificar se existe o equilíbrio entre carga e geração para a ocorrência de uma situação de funcionamento ilhado do sistema na eventual queda da rede elétrica.

A configuração dos equipamentos necessários para a realização do ensaio para detecção de ilhamento de um inversor pode ser observado na Figura 5.

Figura 5 – Configuração dos equipamentos para o ensaio segundo a NBR IEC 62116.



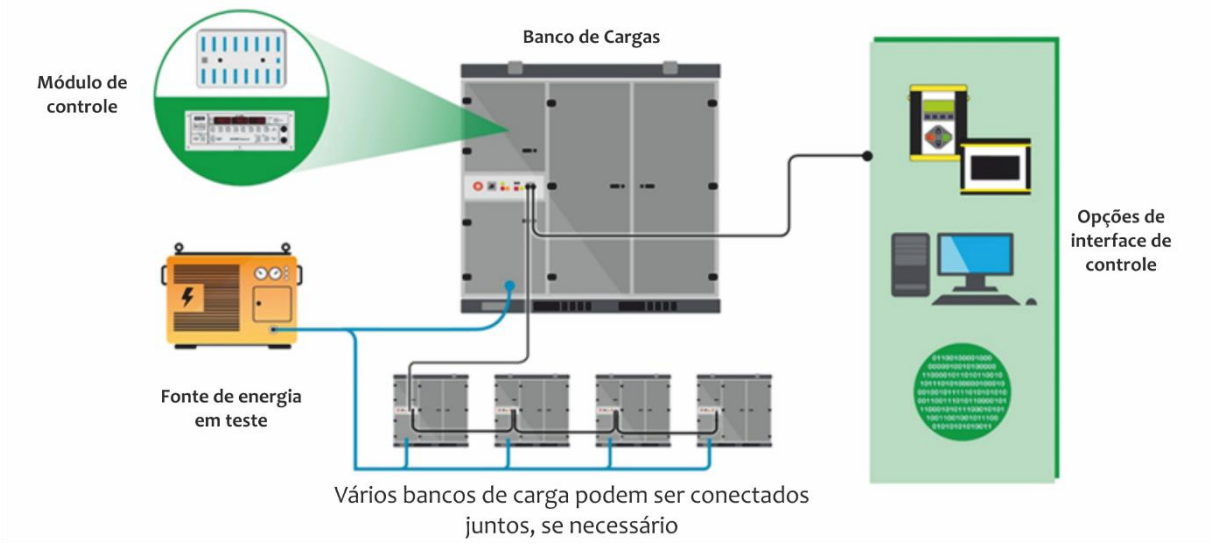
Fonte: Adaptado da ABNT NBR IEC 62116.

2.1.2.2 Banco de cargas RLC

Nos ensaios de proteção contra a situação de operação em ilhamento do inversor é necessário que toda a energia fornecida pelo inversor seja consumida no ponto de conexão com a rede elétrica. Para tanto, é recomendado pela norma ABNT NBR IEC 62116 o uso de um banco de cargas resistiva, indutiva e capacitiva de forma a consumir a energia gerada pelo inversor no ponto de conexão com a rede (ALMEIDA NETO, 2017).

Um banco de cargas combinadas normalmente consiste em elementos resistivos e reativos que permitem simular cargas com fator de potência unitário e não unitário. Na aplicação em questão, a configuração de ensaio possui fator de potência próxima da unitária, mas inclui um circuito LC em paralelo para emular possíveis comportamentos ressonantes do sistema elétrico. O banco consiste em um único conjunto de carga que podem ser comutadas separadamente, sendo somente resistivas, reativas ou ambas ao mesmo tempo, permitindo a variação do fator de potência, logo a capacidade do banco de cargas é dada em (kVA). A Figura 6 traz um arranjo típico de banco de cargas, que possuem módulos de controle e interface com usuário.

Figura 6 – Arranjo típico de banco de cargas para testes.



Fonte: Autor, adaptado de (ASCO POWER, 2018).

2.2 AUTOMAÇÃO DE SISTEMAS PARA ENSAIOS

Atualmente há uma série de equipamentos que possuem interfaces com o usuário, facilitando e automatizando atividades rotineiras. Para o desenvolvimento destas interfaces existem ferramentas voltadas para a interconexão com protocolos comuns aos equipamentos.

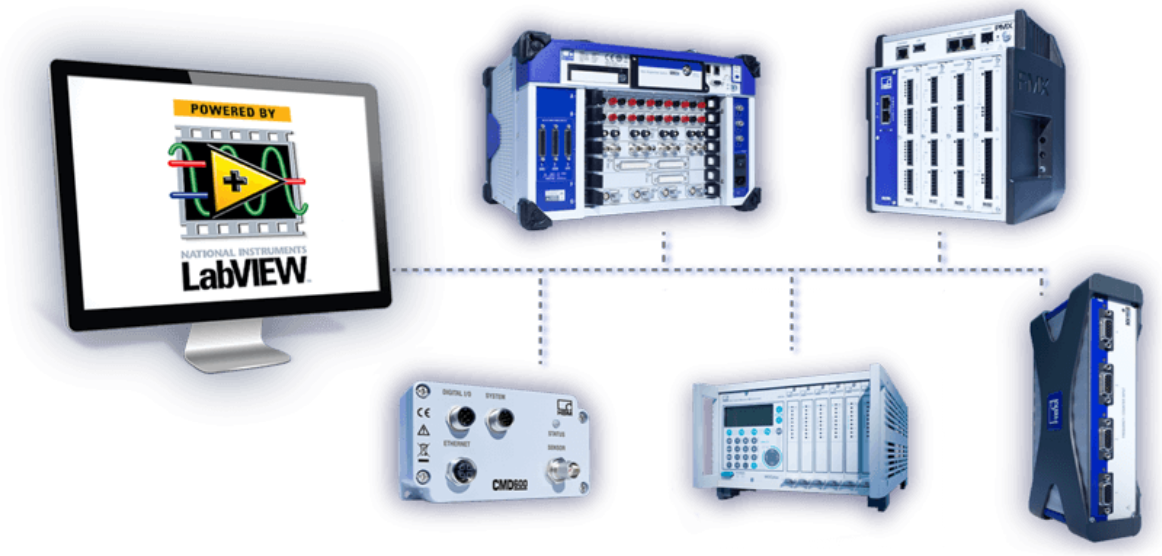
O LabVIEW é um programa muito utilizado no desenvolvimento de interfaces para controle e automação de equipamentos, sendo este o motivo para o uso do mesmo neste trabalho. Além de que, o laboratório onde o sistema será implementado já utiliza outros programas desenvolvidos em LabVIEW e visa padronizá-los.

2.2.1 LabVIEW

O LabVIEW é diferente das usuais linguagens de programação por empregar uma linguagem gráfica conhecida como linguagem G que é composta de muitos nodos conectados. O LabVIEW tem um compilador gráfico aperfeiçoado para maximizar o desempenho do sistema. Isso simplifica o desenvolvimento de aplicações com funções personalizadas para automatizar um dispositivo de aquisição de dados, processar os sinais e exibi-los em interfaces para o usuário (NATIONAL INSTRUMENTS, 2009).

O *software* possui suporte para diversas plataformas, o que facilita sua implementação. Além disso a National Instruments, desenvolvedora do programa, possui diversos dispositivos de aquisição de dados que auxiliam nas funcionalidades do programa, como mostra a Figura 7.

Figura 7 – National Instruments, fabricante do LabVIEW e DAQs.



Fonte: (HBM, 2014).

Dada a sua extensa biblioteca de funções, o LabVIEW excuta as mais diversas tarefas em um ambiente de programação gráfica baseado em um fluxograma intuitivo de blocos funcionais. Contém aplicativos específicos para aquisição, processamento e análise de dados, exibição de resultados e armazenamento.

O ambiente de desenvolvimento do *software* LabVIEW é dividido entre o painel frontal, que permite a interface com o usuário, e o diagrama de blocos, onde é determinado o fluxo de dados e são criadas as funções de controles por meio de blocos. A interação entre os painéis é instantânea, ou seja, ações num painel surtem efeitos no outro, por exemplo, caso um botão seja criado no painel frontal, seu respectivo bloco é criado no diagrama de blocos, o que facilita a conexão entre a interface e a programação gráfica (PIECZOKI, 2015).

Qualquer unidade de funcionamento completo, em LabVIEW, é denominado como um *Virtual Instrument* (VI), conforme (DRUMMOND, 1996). Isso é, qualquer conjunto criado que, recebendo entradas estipuladas, seja capaz de realizar operações, por si só, operar sobre elas, entregando as saídas definidas pode ser considerado como um VI. Isso é, enquadram-se nessa categoria funções, rotinas e programas. Nesse sentido, um VI pode, em si, conter outros VIs, nesse caso denominados SubVIs.

Assim, o desenvolvimento de projetos em LabVIEW envolve principalmente a elaboração de VIs de diversas naturezas. Como um VI pode conter outros em seu

código, diz-se haverem, num projeto, níveis de VIs. Isso é, aqueles que não dependem de outros para funcionarem são ditos de nível mais baixo. Conforme um VI passa a utilizar em seu código outros, seu nível sobe, progressivamente.

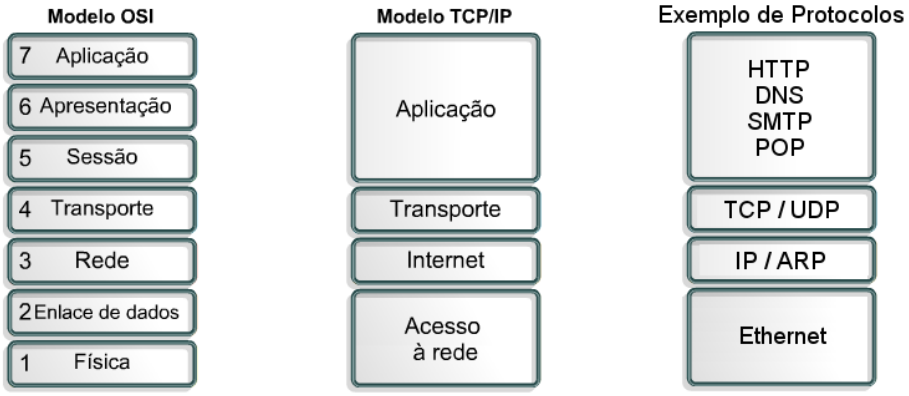
2.2.2 Modelo TCP/IP

Entre os protocolos de comunicação desenvolvidos, o padrão TCP/IP se tornou amplamente utilizado, devido ao fato do seu uso para transferência de páginas *web*, transmissões de e-mail e transferência de arquivos através da Internet.

O TCP/IP é um conjunto de protocolos de comunicação entre computadores em rede, com denominação derivada dos protocolos: TCP – Protocolo de controle de Transmissão e o IP – Protocolo de Interconexão. O protocolo é estruturado em um modelo de camadas, onde cada camada é responsável por algumas tarefas, fornecendo um conjunto de serviços definidos para o protocolo da camada superior.

Conforme pode ser visto na Figura 8, o TCP implementa a camada de transporte do modelo, enquanto que IP é responsável pela camada de rede.

Figura 8 – Modelo de camadas do Protocolo TCP/IP.



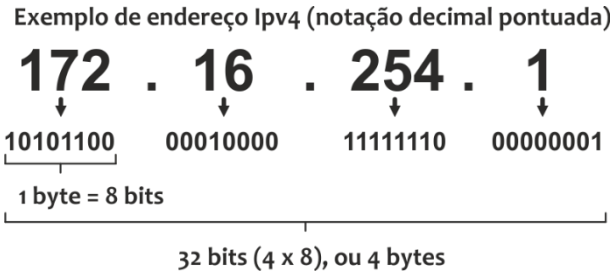
Fonte: (COMER e STEVENS, 1999).

O TCP/IP oferece alguns benefícios, dentre eles, um padrão roteável que é o mais completo e aceito protocolo disponível atualmente. Todos os sistemas operacionais modernos oferecem suporte para o TCP/IP e a maioria das grandes redes se baseia em TCP/IP para a maior parte de seu tráfego. Dada a sua característica multiplataforma, com estrutura para ser utilizada em sistemas operacionais cliente/servidor, permite a utilização de aplicações entre dois pontos distantes.

2.2.2.1 O Protocolo IP

Este protocolo é usado para a entrega de pacotes (também chamados de datagramas IP) com a funcionalidade de endereçamento e roteamento. Atualmente há duas versões de endereçamento, o IPv6 (representado de forma hexadecimal com pontos a cada 2 *bytes*) e o IPv4 (representado de forma decimal com pontos a cada *byte*). A versão 4 estabelece endereços de 32 *bits* (4 *bytes*) conforme Figura 9.

Figura 9 – Representação IPv4.



Fonte: Autor.

Para que pacotes de informação cheguem a seus destinos é necessário definir um caminho, caminho este que segue regras definidas no protocolo IP. Os pacotes deverão sair do computador de origem chegando até o computador de destino, podendo passar por uma ou mais redes para atingir tal objetivo. Segundo (SOARES, LEMOS e COLCHER, 1995) o serviço oferecido pelo IP é sem conexão. Portanto, cada pacote IP é tratado como uma unidade independente que não possui nenhuma relação com qualquer outro pacote. A comunicação é não-confiável, não sendo usados reconhecimentos fim-a-fim ou entre nós intermediários. Nenhum mecanismo de controle de fluxo é empregado, assim como, nenhum mecanismo de controle de erros nos dados transmitidos é utilizado, exceto um *checksum* do cabeçalho que garante que as informações nele contidas, que são usadas pelos *gateways* (roteadores) para encaminhar os pacotes estão corretas. Todos esses mecanismos para tornar a comunicação confiável são implementados em uma camada superior.

No cabeçalho do pacote IP, estão contidas as informações sobre o endereço de origem e destino dos dados encapsulados. O endereço IP é o responsável pela identificação de cada equipamento conectado à rede, assim endereços internos devem ser únicos em suas subredes, possibilitando a ocorrência de uma comunicação entre dois computadores distintos na mesma subrede.

2.2.2.2 O Protocolo TCP

O protocolo TCP é um serviço de entrega de segmentos de dados, que garante entrega e integridade e, funciona basicamente na conexão lógica entre dois computadores. Nesse tipo de comunicação, ambos os computadores entram em conformidade de como será feito o envio dos segmentos de dados entre si. Quando uma informação é transmitida, mecanismos de verificação de integridade garantem que a informação seja recebida sem erros.

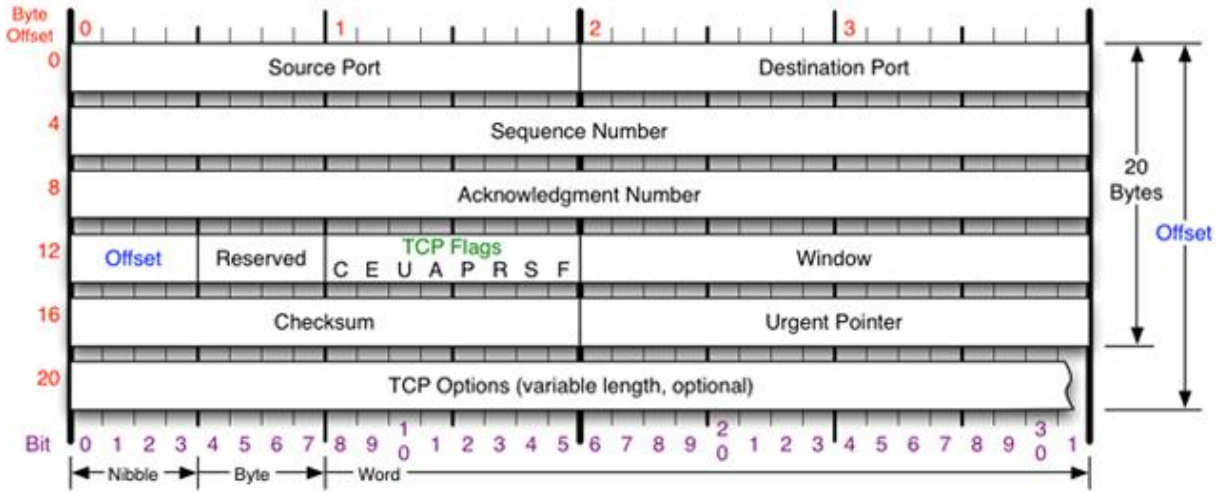
Antes de transmitir as informações, o protocolo TCP estabelece uma comunicação entre os computadores, num processo chamado *three-way-handshake*. Ao final da transmissão das informações, a conexão é fechada pelo mesmo processo.

Para transmitir os dados, o segmento de dados TCP é dividido em segmentos menores, tipicamente 64 *kbytes* (podem ser no máximo 1460 *bytes* para caber em um quadro Ethernet), que são numerados e enviados ao destino. O receptor, recompõe o pacote original, reordenando se preciso segmentos que chegaram fora de ordem e/ou solicitando o envio de segmentos que não chegaram. Cada segmento é verificado por meio de um *checksum* para que não tenha sofrido interferência no meio do caminho por parte do meio físico.

No protocolo TCP ocorre um fluxo de *bytes* não de mensagens. Entidades transmissoras e receptoras do TCP trocam dados na forma de segmentos. Um cabeçalho de segmentos possui 20 *bytes* e é seguido de 0 ou mais *bytes* de dados. O cabeçalho do protocolo TCP, que pode ser visto na Figura 10, contem campos de grande importância para o correto funcionamento de redes baseadas neste modelo. Entre esses existe o campo de 16 *bits* para identificação das portas de fonte e destino, o campo que informa o número de sequência do segmento, o campo que informa o próximo número de segmento que está sendo aguardado, se o ACK está configurado, além do campo de *flags* que identificam ações nas negociações entre *hosts.*

O protocolo TCP especifica três fases durante a conexão: primeiro o estabelecimento da conexão, depois a transmissão e por último o termino da conexão. Havendo uma grande negociação entre *hosts* para garantir a entrega dos segmentos TCP.

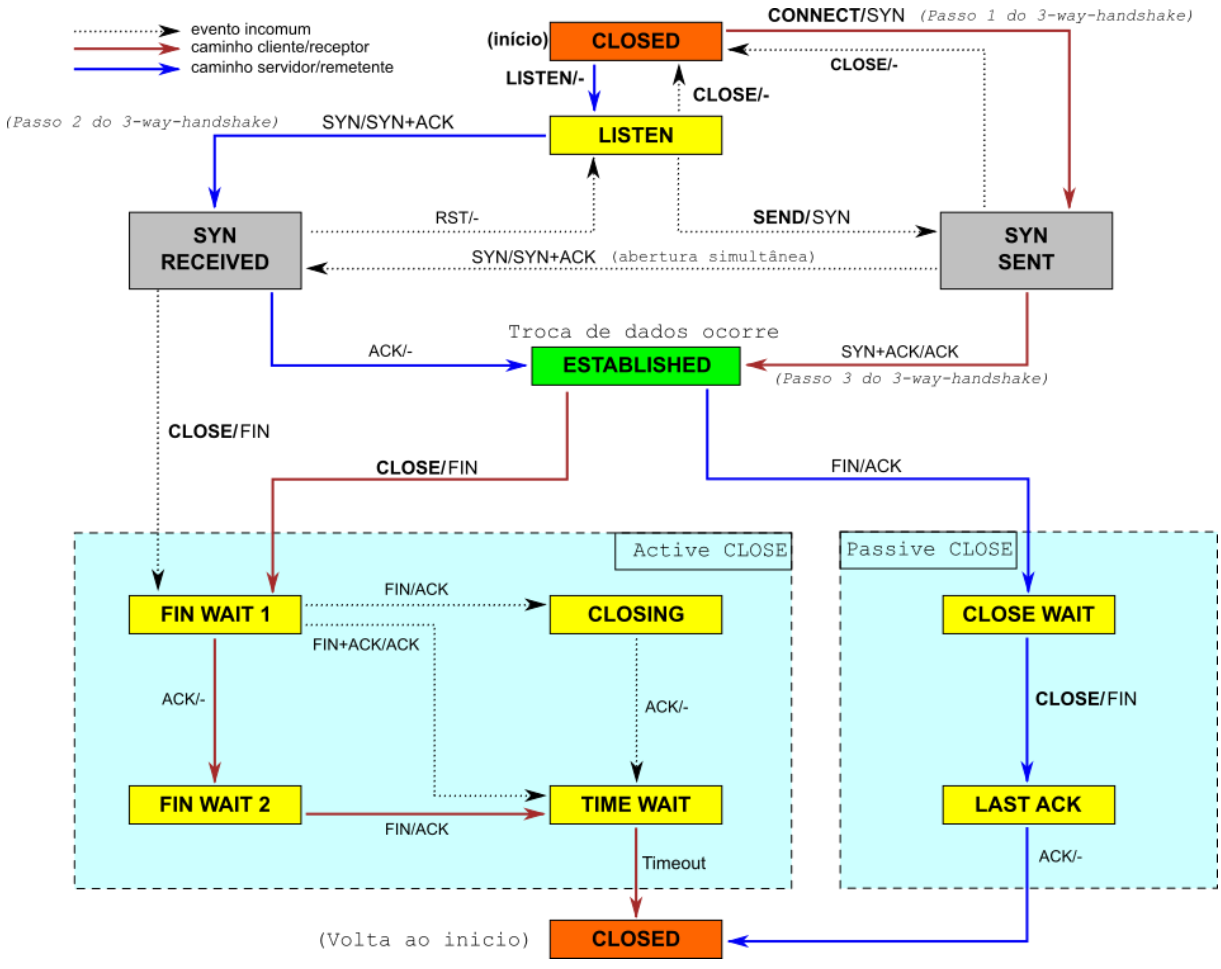
Figura 10 – Estrutura do cabeçalho TCP.



Fonte: (BAXTER, 2017).

O diagrama da Figura 11 ilustra um pouco desta negociação com os estados do transmissor e receptor durante a troca de *flags* para configuração da conexão.

Figura 11 – Diagrama do processo *three-way-handshake*.



Fonte: (PAULEY, 2008).

Fazendo a análise do diagrama, a conexão TCP é estabelecida do seguinte modo:

1. O *host* que inicia a conexão envia um segmento com a *flag SYN* ativada;
2. O *host* de destino, ao receber o segmento, envia outro segmento, com as *flags SYN* e *ACK* ativadas, um número de sequência que identifica o próximo segmento que o *host* enviará e o próximo número de sequência que este *host* espera receber;
3. O *host* que iniciou a conexão, envia um último segmento com a *flag ACK* ativada, o número de sequência que o outro *host* espera e o próximo número de sequência que espera receber.

2.2.2.3 Padrão Ethernet

Ethernet é uma arquitetura de interconexão para redes locais (LAN) baseada no envio de quadros. Ela define cabeamento e sinais elétricos para a camada física, em formato de quadros e protocolos para a subcamada de controle de acesso ao meio (MAC). É baseada na ideia de pontos da rede enviando mensagens, cada ponto tem uma chave de 48 *bits* globalmente única, conhecida como endereço MAC, para assegurar que todos os sistemas em uma Ethernet tenham endereços distintos.

A maioria das instalações modernas de Ethernet usa *switches*, em que a maior vantagem é restringir os domínios de colisão, o que causa menos colisão no meio compartilhado causando uma melhora no desempenho da rede. Redes com *switches* tipicamente seguem uma topologia em estrela, embora elas ainda implementem uma "nuvem" única de Ethernet do ponto de vista das máquinas ligadas.

O modelo TCP/IP não define um meio de acesso a rede específico na sua camada que engloba o meio físico e o enlace de dados, exceto que deve suprir conexão à rede para possibilitar a entrega de pacotes IP. Normalmente é utilizado a padronização Ethernet (IEEE 802.3) ou Wi-Fi (IEEE 802.11) nesta camada.

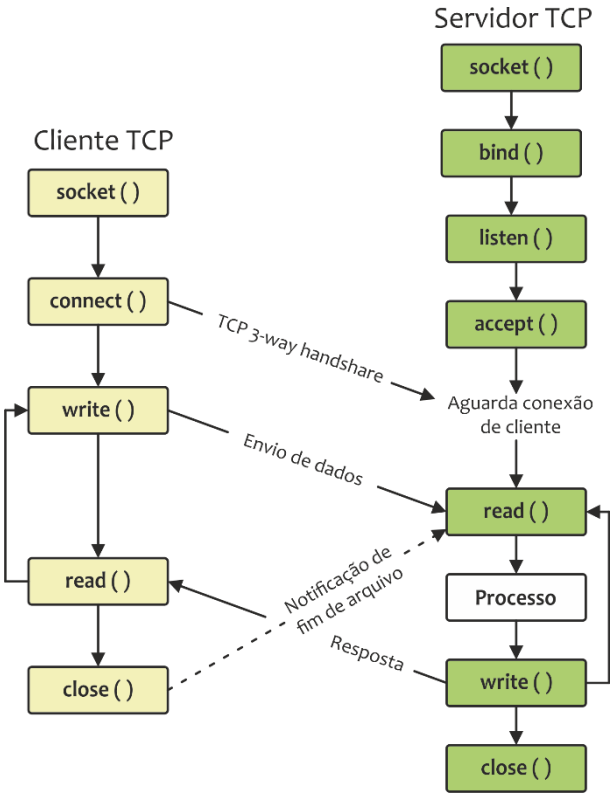
2.2.2.4 Sockets de rede

Um *socket* de rede é o ponto-final de um fluxo de comunicação entre 2 aplicativos através de uma rede, fazendo a interface de programação de aplicativos (API),

normalmente fornecida pelo sistema operacional, que permite que os programas de aplicação controlem e usem a rede.

Uma forma simples de implementar uma estrutura de comunicação cliente/servidor é utilizando *sockets*, que abstraem a camada de rede para que uma aplicação possa se comunicar com outra sem ter que se preocupar com detalhes da pilha TCP/IP que gere a rede abaixo da aplicação. Ou seja, os *sockets* estão entre a camada de transporte e a de aplicações, gerando a interface para comunicação entre aplicações, um exemplo de fluxo de trabalho utilizando *sockets* é visto na Figura 12.

Figura 12 – Fluxo utilizado por sockets em comunicação cliente/servidor.



Fonte: Autor, adaptado de (PERRONE, 2018).

O programa cliente primeiro cria um *socket* através da função *socket ( )*. Em seguida ele se conecta ao servidor através da função *connect ( )* e inicia um *loop* que permite envios (*send( ))* e recebimentos (*recv( ))* com as mensagens específicas da aplicação. É no par *send, recv* que temos a comunicação lógica. Quando alguma mensagem da aplicação diz que é o momento de terminar a conexão, o programa chama a função *close( )* para finalizar o *socket*.

O programa servidor também utiliza a mesma API de *sockets.* Ou seja, inicialmente ele também cria um *socket.* No entanto, diferentemente do cliente, o servidor precisa de um *bind( ),* que associa o *socket* a uma porta do sistema operacional, e depois utiliza o *listen( )* para aguardar novas conexões de clientes nessa porta. Quando um novo cliente faz uma nova solicitação de conexão, a chamada *accept( )* é utilizada para confirmar a conexão. Da mesma forma que no cliente, o servidor fica em *loop* recebendo e enviando mensagens através do par de funções *send( )* e *recv( )*.

2.2.3 Linux Embarcado

Linux Embarcado é a aplicação e uso do kernel Linux em uma placa eletrônica cujo principal elemento é o SoC (*System-on-a-chip*). SoC é um *chip* eletrônico que reúne todos os elementos básicos de um sistema computacional como CPU, memória principal (RAM), memória secundaria (ROM/FLASH) e alguns periféricos. Atualmente um SoC contém internamente diversos periféricos dos mais variados tipos como I2C, SPI, UART, CAN, enfim, a lista pode ser enorme (MACIEL, 2014).

O *kernel* Linux em conjunção com outros *softwares* são escritos na memória *flash* ou outra mídia de armazenamento presente na placa, como, por exemplo, um cartão uSD. Esta combinação forma um sistema operacional completo e funcional.

O mesmo Linux que roda em um supercomputador pode rodar também em uma simples placa eletrônica. O que torna isso possível é que o Linux suporta uma grande variedade de arquiteturas e processadores. Mas nem todas essas arquiteturas são de fato usadas em sistemas embarcados. As mais comumente usadas em sistemas embarcados são ARM, PowerPC e MIPS.

Muitas sub-comunidades mantêm seu próprio *kernel*, com atributos novos e distintos, porém menos estáveis. E em alguns casos, os fabricantes mantêm versões derivadas do Linux oficial para dar suporte para seu *hardware* específico. Como exemplo, a BeagleBone Black que mantém seu próprio *kernel*.

A disponibilidade de *ports* feitos por distribuições como Debian torna a programação mais fácil em qualquer linguagem disponível em um *desktop* Linux. Por exemplo, é possível programar em C, C++, Java, Perl, Python. Muitos projetos têm sido feitos em Python, devido à facilidade de programação e ao número de bibliotecas disponíveis.

2.2.4 Python

Python é uma linguagem de programação de altíssimo nível (VHLL) criada sob o ideal de “Programação de computadores para todos”. Este ideal fez com que o desenvolvimento de Python tivesse sempre em mente a liberdade (código aberto), disponibilidade (compatível com Windows, Linux, Mac, e outra infinidade de sistemas) e principalmente a clareza de sintaxe, que hoje é responsável pela alta produtividade.

É uma linguagem orientada a objetos, um paradigma que facilita entre outras coisas o controle sobre a estabilidade dos projetos quando estes começam a tomar grandes proporções. Mas como a Orientação a Objetos ainda é vista como um paradigma para “experts”, permite que o usuário programe na forma procedural. Python também é uma linguagem altamente modular. Isso significa que provavelmente já existam bibliotecas com funções especificas utilizadas por um programa a ser desenvolvido, economizando tempo de trabalho.

Enfim, a escolha de se trabalhar com Python para o desenvolvimento do programa utilizado neste trabalho está resumida neste paragrafo obtido em um portal para desenvolvedores:

> *Python tem uma curva de aprendizado bastante interessante, permitindo que novos programadores, mesmo os que nunca tenham programado antes, sejam imediatamente produtivos escrevendo scripts procedurais. O programador pode rodar o interpretador como um shell, vendo imediatamente o resultado da saída de cada comando e explorando os recursos da linguagem interativamente.*

Além de possuir bibliotecas previamente desenvolvidas que auxiliam na utilização de *sockets* e criação de *scripts* em sistemas embarcados rodando Linux, como a BeagleBone Black.

2.3 SUMÁRIO

Com a crescente representação de sistemas fotovoltaicos conectados à rede elétrica, no cenário energético brasileiro, foram necessárias ações para regulamentação dos equipamentos utilizados na conexão com o sistema elétrico de potência.

A norma ABNT NBR/IEC 62116 introduziu o ensaio de anti-ilhamento à inversores fotovoltaicos, utilizados para condicionar a ligação de sistemas

fotovoltaicos à rede elétrica. Visto a configuração de equipamentos necessários para a realização deste ensaio, é necessário o uso de cargas no ponto de conexão com a rede, para realizar o consumo de toda a energia fornecida pelo inversor na situação de operação em ilhamento. Assim, um conjunto de cargas pode ser comutadas separadamente de forma automática, fazendo o uso de ferramentas para automatizar este processo.

Na seção 2.2 deste capítulo são descritas ferramentas que auxiliaram no desenvolvimento de uma interface Ethernet para comunicação entre o *firmware* do equipamento controlado e o sistema supervisório.

3 SISTEMA DESENVOLVIDO

Após a estruturação da ideia inicial do projeto motivada pela automação nos processos de ensaios desenvolvidos pelo Laboratório de Ensaios – Setor Fotovoltaico, deu-se início ao estudo do sistema que era utilizado e, por conseguinte as possíveis formas de se automatizar o processo.

O desenvolvimento do sistema de automação para a carga RLC foi projetado juntamente a outros dispositivos previamente desenvolvidos, responsáveis por realizar a comutação das cargas e fazer a medição das saídas de fontes trifásicas utilizadas nos ensaios.

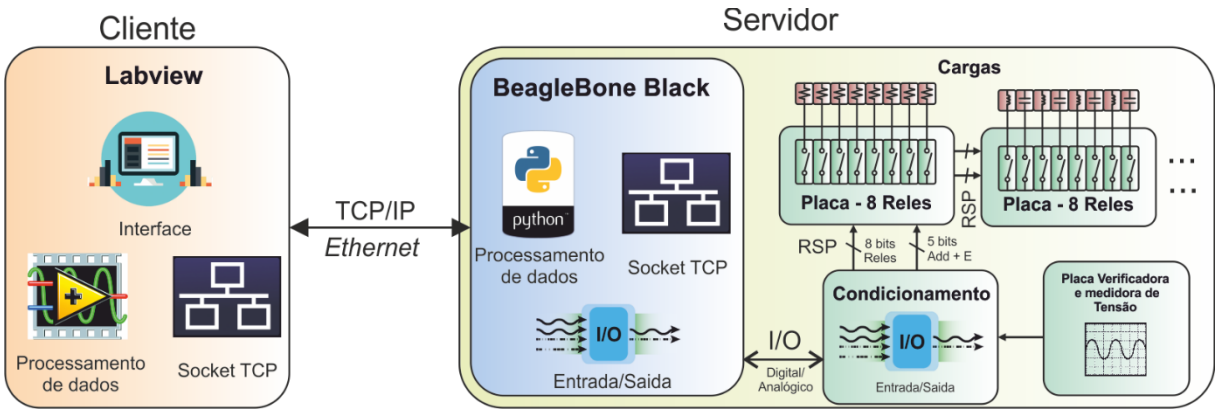
3.1 METODOLOGIA E MATERIAIS UTILIZADOS

Inicialmente optou-se por utilizar a plataforma de desenvolvimento BeagleBone Black para implementar o controle central da carga RLC, responsável por realizar a comunicação entre a interface de controle desenvolvida e os dispositivos responsáveis por controlar as comutações. Essa escolha foi baseada, em já se possuir a plataforma, contudo não haver um material que traga informações sobre a mesma, visando assim a elaboração de conteúdo para uso posterior desta plataforma de desenvolvimento em projetos futuros.

Dado que o sistema desenvolvido não requer de armazenamento de dados, já que os comandos enviados pela interface LabVIEW deveriam ocorrer instantaneamente, optou-se por realizar a comunicação por meio de *sockets* TCP/IP, fazendo a comunicação direta entre as aplicações. E devido a simplicidade de se implementar *sockets* em Python, foi escolhida esta linguagem para desenvolver o *script* executado no servidor.

Assim, a estrutura de comunicação foi definida, utilizando sockets para envio de dados entre cliente (aplicação LabVIEW) e servidor (aplicação Python), conforme pode ser visto na Figura 13.

Figura 13 – Estrutura de comunicação e fluxo de dados.

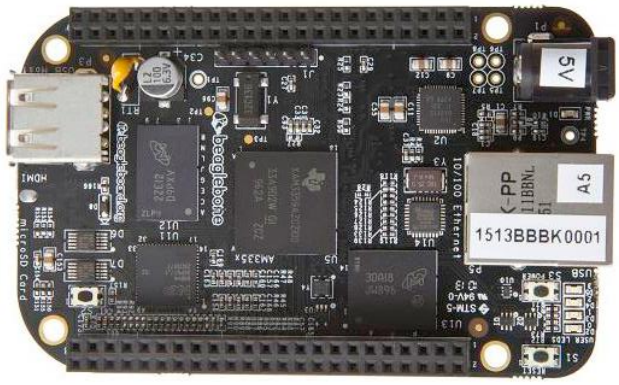


Fonte: Autor.

3.1.1 BeagleBone Black

A solução de *hardware* utilizada neste projeto para executar uma distribuição de Linux Embarcado foi a BeagleBone Black Rev C, ilustrada na Figura 14, que é uma plataforma de desenvolvimento de *hardware* livre, que disponibiliza uma infinidade de funcionalidades, sendo um computador de placa única (CPU), com todos os componentes necessários para seu funcionamento situados em uma única placa de circuito impresso (COELHO, 2017).

Figura 14 – BeagleBone Black.



Fonte: (BEAGLEBOARD, 2018).

Diferentemente de outras placas de desenvolvimento de sistemas com especificações semelhantes, a BeagleBone foi desenvolvida primariamente como uma ferramenta de prototipagem rápida e desenvolvimento de projetos eletrônicos devido ao seu *hardware open source* e grande quantidade de interfaces I/O (COLEY, 2012).

A Tabela 1 mostra as principais características e especificações da plataforma.

Tabela 1 – Especificações da BeagleBone Black Rev C.

Processador	Sitara XAM3359AZCZ100, 1GHz, 2000MIPS
***Engine* gráfica**	SGX530 3D, 20M Polígonos/s
Memória SDRAM	512MB, DDR3L, 800MHz
***Onboard* Flash**	4GB, MMC embarcado de 8 bits
Fontes	Mini USB, USB ou entrada DC (5V)
Ethernet	10/100 RJ45
Pinos	VCC 5V e 3.3V, 19 GPIO (entrada/saída digital, 3.3V), 7 entradas analógicas (1.8V), 5 PWM, 4 UART, 4 timers, entradas digitais configuráveis
Conector SD/MMC	Micro SD, 3.3V

Fonte: (BEAGLEBOARD, 2018).

A BeagleBone Black possui dois conectores de expansão, com 46 pinos cada, tornando-a um computador adequado para coleta de dados e controle de dispositivos, onde a maioria dos pinos estão conectados ao SoC. Desses pinos, 65 podem ser utilizados como entradas e saídas digitais para propósitos gerais (GPIO), selecionando o modo 7, para configurar um pino do SoC, é necessário entender seu funcionamento. A distribuição dos pinos pode ser vista no apêndice A, que apresenta um resumo das informações do processador presente na BleagleBone Black[1].

Cada pino digital possui até 8 modos diversos, que implementam funções especificas. As alterações na configuração dos pinos devem ser realizadas no *device tree* da BeagleBone Black. Uma característica interessante é a possibilidade de alterar a definição dos pinos do SoC via *device tree* em tempo de execução (*device tree overlay*). Esta funcionalidade é toda implementada no *kernel* por um subsistema chamado Capemgr. A ideia principal é possibilitar a conexão de módulos adicionais de *hardware* ao sistema (capes) sem precisar alterar o *device tree* original da placa.

Quando o assunto é redes integradas, a BBB não apenas possui uma conexão Ethernet na própria placa, mas também já tem incluídas e disponíveis todas as ferramentas básicas de rede presentes nos pacotes do Linux. Podem ser usados serviços como FTP (Protocolo de Transferência de Arquivos), Telnet (protocolo de comunicação baseado em texto interativo bidirecional), SSH (Secure Shell – protocolo

[1] Disponível na seção “Control Module” do Technical Reference Manual do AM3359.

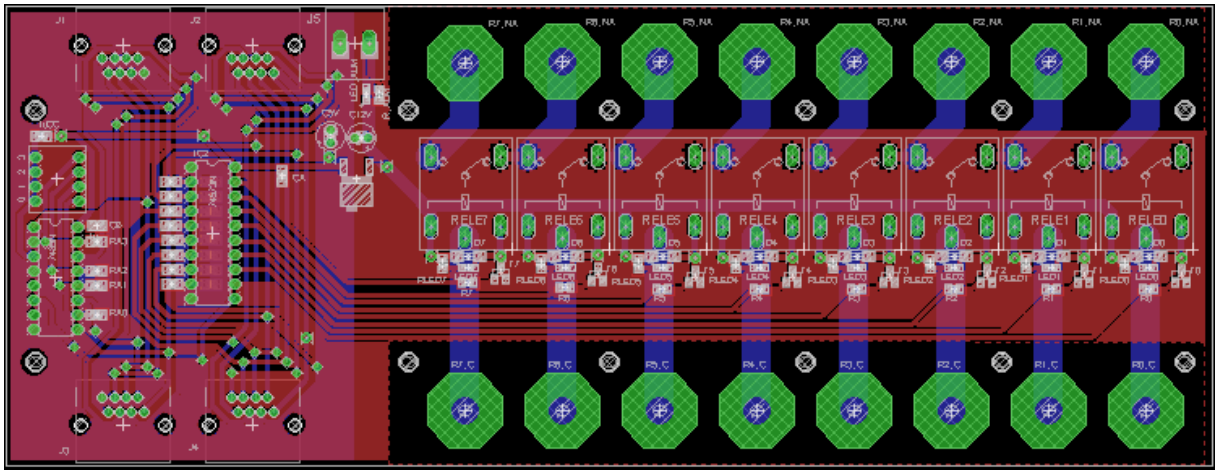
de rede criptográfico), ou até mesmo hospedar um servidor *web* próprio na placa (RICHARDSON, 2013).

Inicialmente foi necessário realizar a configuração da BeagleBone, conforme as necessidades do projeto em desenvolvimento. O primeiro passo foi a verificação do sistema operacional a ser utilizado. No caso a BeagleBone Black vem originalmente com a distribuição Angstrom pré-instalada na eMMC, esta foi alterada pela distribuição Debian 9.4 – Stretch[2], que já vem compilada com as ferramentas necessárias para o desenvolvimento do projeto. No Apêndice B é descrito com detalhes as etapas necessárias para a gravação da imagem e configuração da BeagleBone.

3.1.2 Placa comutadora de relés

A placa ED_8_Relés vista na Figura 15, desenvolvida por (PIETTA, 2017), permite a comutação de até oito cargas de forma isolada, e foi projetada baseada em relés eletromecânicos. O sistema foi projetado para possibilitar o uso de 15 placas simultâneas, a partir de um único barramento, conectado as placas em série, a partir dos seus conectores RJ45 de entrada e saída.

Figura 15 – Placa 8 relés para comutação de cargas.



Fonte: (PIETTA, 2017).

O seu funcionamento baseia-se no endereçamento de sinais com os estados de relés, e isso é feito por meio de dois conectores RJ45, que recebem o endereçamento por um conector e os sinais de comutação por outro. O endereço de cada placa no

[2] Disponível em debian.beagleboard.org/

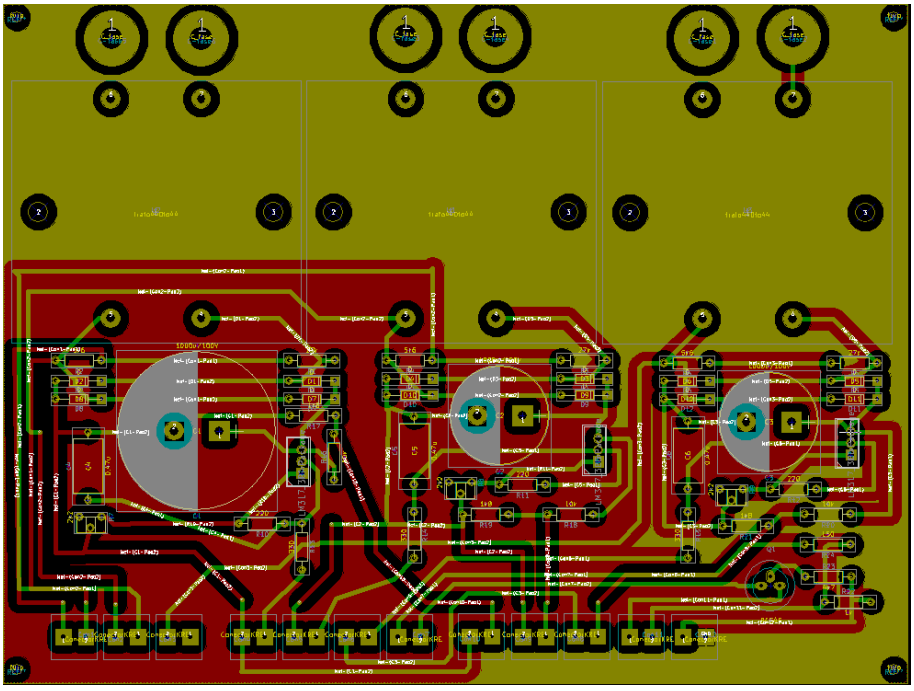
sistema pode ser ajustado manualmente por um *dip switch* de quatro *bits*. Assim, quando os sinais de endereçamento enviados ao comparador de magnitude (CI 7485) presente na placa forem iguais aos configurados manualmente, é gerado um sinal de *clock*, que habilita a mudança de estado do CI 74573, composto de 8 *latches* do tipo D com saída paralela, para os sinais de comutação recebido no respectivo conector. Dessa forma, é realizada a comutação dos relés.

Uma alteração no projeto desta placa foi realizada para evitar problemas de funcionamento da mesma devido ao sincronismo necessário dos sinais de endereçamento para o correto funcionamento do comparador de magnitude. Detectou-se que se um dos sinais tiver um atraso maior que os outros, ocorre um erro no endereçamento. A solução empregada foi a utilização de um dos sinais auxiliares não aproveitados, para habilitar a comparação do CI 7485.

3.1.3 Placa de medição de tensão

A placa verificadora e medidora de tensão, desenvolvida por (LENZ, 2018), Figura 16, é parte do desenvolvimento do sistema para a automatização do banco de cargas RLC.

Figura 16 – Placa verificadora e medidora de tensão.



Fonte: (LENZ, 2018).

É conectada em paralelo com a entrada de uma fonte trifásica no banco de cargas. Juntamente a placa de aquisição informa a existência de tensão, sendo essencial para que não haja comutação de relés sob carga, enquanto houver tensão remanescente no banco de carga RLC.

3.1.4 Carga RLC

A Tabela 2 apresenta os valores de potência dos elementos da carga RLC para tensões de ensaio fase-neutro de 220 V. Esses valores são utilizados como base nos cálculos para determinação da comutação das cargas, segundo a potência necessária aos equipamentos ensaios.

Tabela 2 – Potência ativa e reativa disponível no banco de cargas.

	Indutor (Var)							
	8	7	6	5	4	3	2	1
R	12118	6340	3583	1605	809	409	201	0
S	12213	6353	3165	1605	803	428	196	0
T	12010	6341	3180	1563	800	404	202	0

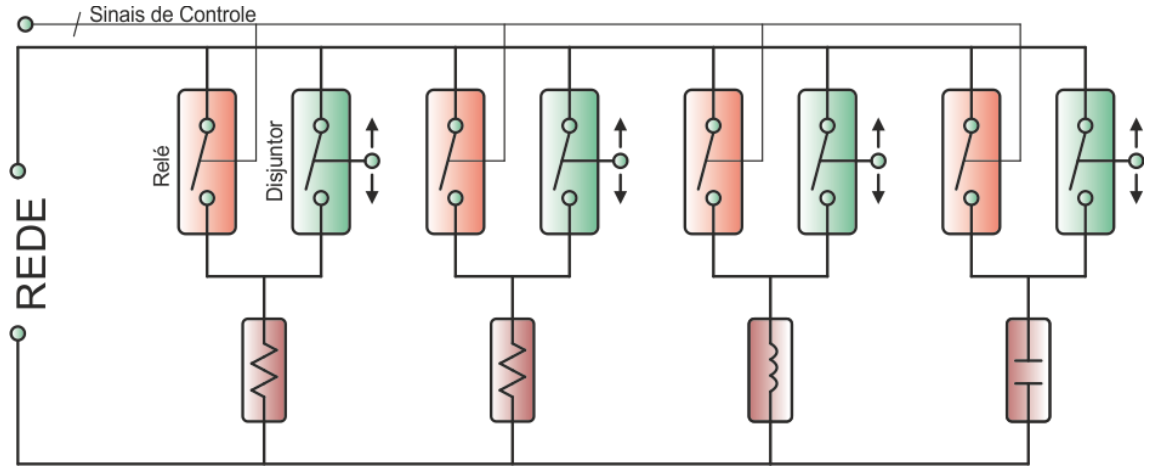
	Resistor (W)														
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
R	10 M	7747	3706	2053	1026	507	248	119	61	34	17	9	4	2	1
S	10 M	7642	3764	2061	1025	508	253	119	60	35	17	10	4	2	1
T	10 M	7683	3729	2052	1024	509	250	118	61	35	17	8	5	2	1

	Capacitores (Var)															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
R	9195	4764	2350	1164	584	295	180	74	40	19	9	4,6	2,2	1,3	0,6	0,3
S	9195	4722	2357	1155	582	294	180	73	40	20	9	4,6	2,3	1,2	0,6	0,3
T	9195	4712	2355	1162	587	292	188	77	40	20	9	4,3	2,3	1,3	0,8	0,3

Fonte: Autor.

O banco de cargas RLC trifásico é formado por três bancos monofásicos de cargas semelhantes, compostos atualmente por 15 resistores, 8 indutores e 16 capacitores conectados em paralelo, e possui uma disposição de forma que possibilite a comutação das cargas através de disjuntores (manualmente) ou relés (através dos sinais enviados), conforme Figura 17.

Figura 17 – Arranjo de cargas e dispositivos de comutação.



Fonte: Autor.

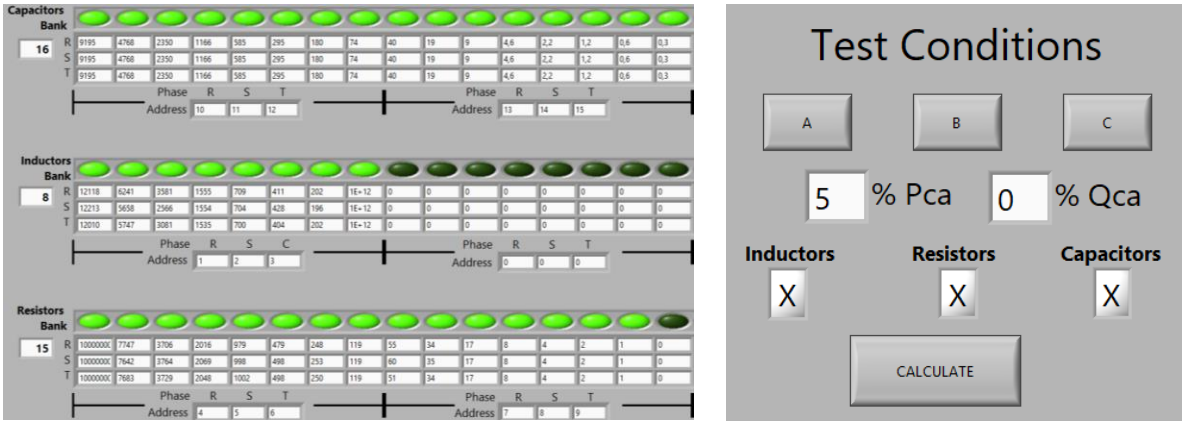
3.1.5 Supervisório LabVIEW para controle da carga RLC

O *software* desenvolvido previamente, por (LENZ, 2018), responsável pelo controle da carga RLC, foi feito em quatro partes:

1. Coleta de dados iniciais e seleção dos parâmetros do ensaio;
2. Determinação das potências ativa, reativa e aparente do ensaio;
3. Determinação da combinação dos elementos da carga RLC empregados no ensaio;
4. Comutação e protocolo de comunicação entre computador e a carga RLC.

Através deste supervisório é possível editar a configuração de cargas, e selecionar os parâmetros de ensaio, conforme visto na Figura 18.

Figura 18 – Painéis supervisório LabVIEW, a) Configuração de Cargas; b) Parâmetros de Ensaio.



a) b)

Fonte: Autor.

3.2 PLACA I/O PARA CONDICIONAMENTO E ENVIO DE SINAIS

Além do desenvolvimento das aplicações responsáveis pela troca de dados, foi necessário o desenvolvimento de um dispositivo para condicionamento e proteção dos sinais controlados pela plataforma embarcada, devido a limitação de corrente e tensão de alguns pinos. Sinais com níveis de tensão maiores que 1,8 V podem danificar as entradas analógicas do sistema embarcado. Além de que, os pinos digitais não devem fornecer correntes superiores a 3 mA, acarretando em possíveis danos.

Portanto foi desenvolvida uma placa, nomeada Cape RLC_Load, responsável por isolar os pinos digitais de saída, utilizando *buffers* para limitar a corrente necessária para controle das placas de comutação. Nesta mesma placa foi desenvolvido um sistema de condicionamento para as entradas analógicas, responsáveis pela leitura de tensão das fases de uma fonte trifásica.

Além disso, foram alocados conectores empilháveis, para conexão com o sistema embarcado e, conectores RJ45, para facilitar a conexão com os dispositivos previamente desenvolvidos, que serão responsáveis pela comunicação com as placas de comutação conectadas as cargas RLC.

O desenvolvimento destes circuitos é mostrado a seguir.

3.2.1 Circuito de aquisição e condicionamento de sinais de tensão

As entradas analógicas da BeagleBone Black aceitam valores entre 0 e 1.8 V, assim sendo necessário condicionar os sinais adquiridos para que fiquem nesta faixa de tensão, já que este circuito recebe sinais de tensão, por meio de um conector RJ45, em uma faixa de 0 a 5 V, provindos da placa responsável pela verificação e medição de tensão.

Assim, foram utilizados um amplificador operacional *rail-to-rail* LM6132 e um diodo Schottky BAT54 na saída do circuito de instrumentação para limitar a tensão nas entradas analógicas da BeagleBone.

Quando o sinal de saída for superior a tensão necessária para polarização do diodo Schottky, em torno de 270 mV, este passa a conduzir, limitando a tensão de saída. Se a tensão na saída for inferior a -270 mV, um diodo conduz para o plano terra, já se este valor for superior a 1,8 V, outro diodo conectado a tensão de referência de 1,63 V passa a conduzir. Esta tensão de referência é obtida a partir de um divisor de tensão, Equação (1), seguido por seguidor de tensão, conectado a referência de

1,8 V, disponibilizada pela BeagleBone ao circuito analógico. Desta forma o sinal de saída fica limitado entre -0,2 e 1,8 V, atendendo os requisitos necessários.

$$1{,}63 = \frac{R_2}{R_1 + R_2} \cdot 1{,}8\ ;\ R_1 = 1k\Omega\ ; R_2 = 10k\Omega \tag{1}$$

Como a faixa do sinal adquirido vai de 0 a 5 V, foi utilizado um divisor de tensão, para condicionar o sinal de entrada, fazendo a regulação de uma faixa de 5 V, para 1,63 V, conforme Equação (2). Sabendo que o circuito é replicado para as três fases medidas.

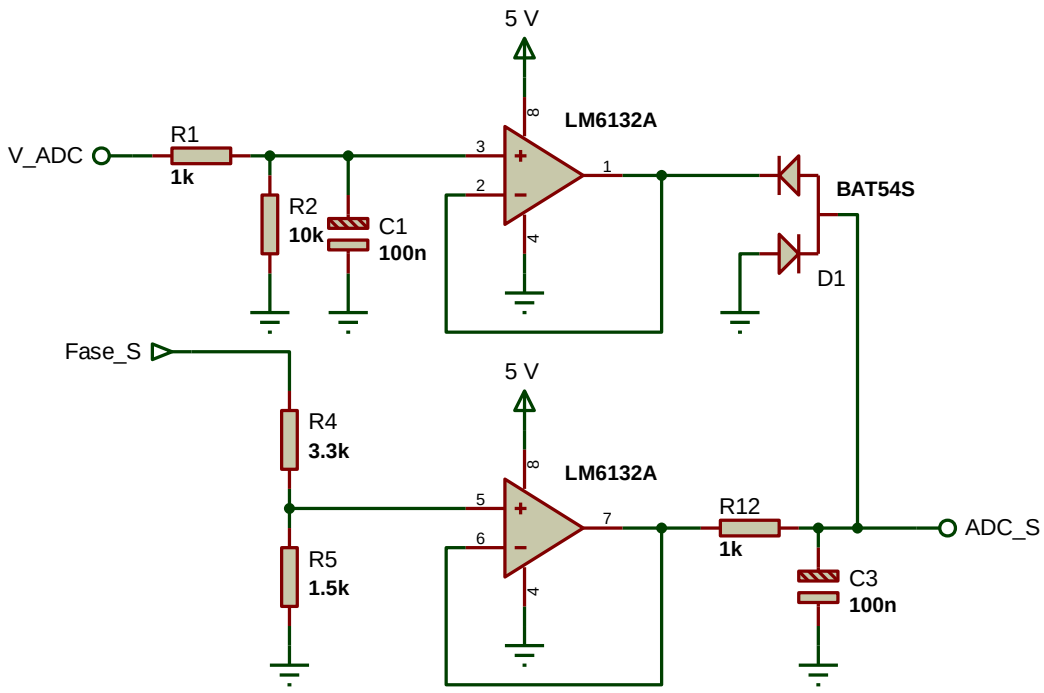
$$1{,}63 = \frac{R_5}{R_4 + R_5} \cdot 5\ ;\ R_4 = 1{,}5k\Omega\ ; R_5 = 3{,}3k\Omega \tag{2}$$

O circuito ainda possui um filtro de anti-*aliasing*, composto por um filtro RC passa-baixa, com frequência de corte de 1590 Hz, calculada conforme Equação (3).

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 1k\Omega \cdot 100nF} = 1590\ Hz \tag{3}$$

Assim o circuito de aquisição foi montado de acordo com o esquema mostrado na Figura 19.

Figura 19 – Circuito de condicionamento dos sinais de tensão.

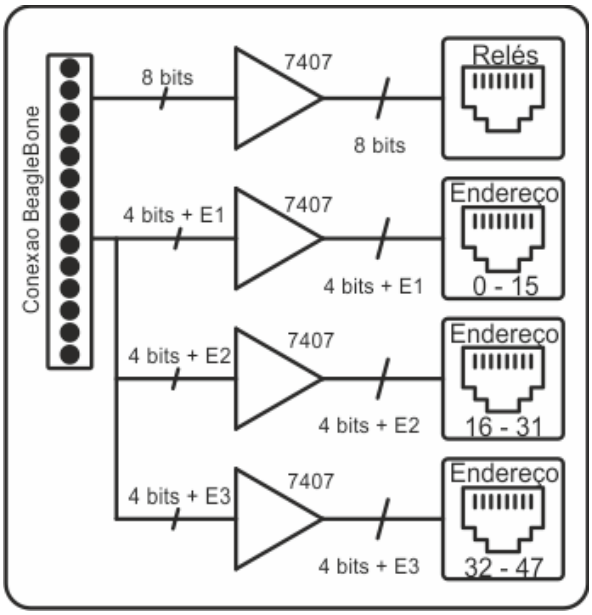


Fonte: Autor.

3.2.2 Circuito para conexão com placa comutadora de 8 relés

Essa placa é conectada ao dispositivo de comutação com 8 relés. Para a proteção dos pinos digitais de saída da BeagleBone Black, que não deve fornecer mais que 3 mA de corrente, os sinais passam por um *buffer* 7407, que limita a corrente suprida. Estes sinais são enviados para a placa de comutação de relés através dos conectores RJ45. O circuito, Figura 20, ainda apresenta resistores *pull-down* de 10 kΩ na saída da BeagleBone e *pull-up* após o *buffer*.

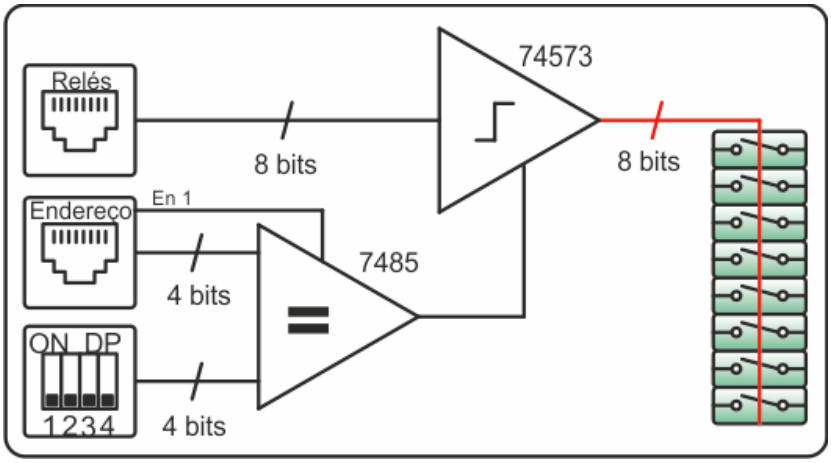
Figura 20 – Circuito para conexão com placa ED_8_RELES.



Fonte: Autor.

Assim é possível enviar os sinais a placa ED_8_RELES, com o funcionamento básico resumido na Figura 21, que mostra os 13 sinais necessários para o seu funcionamento, sendo 8 relativos aos estados de cada relé presente na placa, 4 correspondentes ao endereçamento da placa, feito por um comparador binário, e o sinal que habilita a comparação.

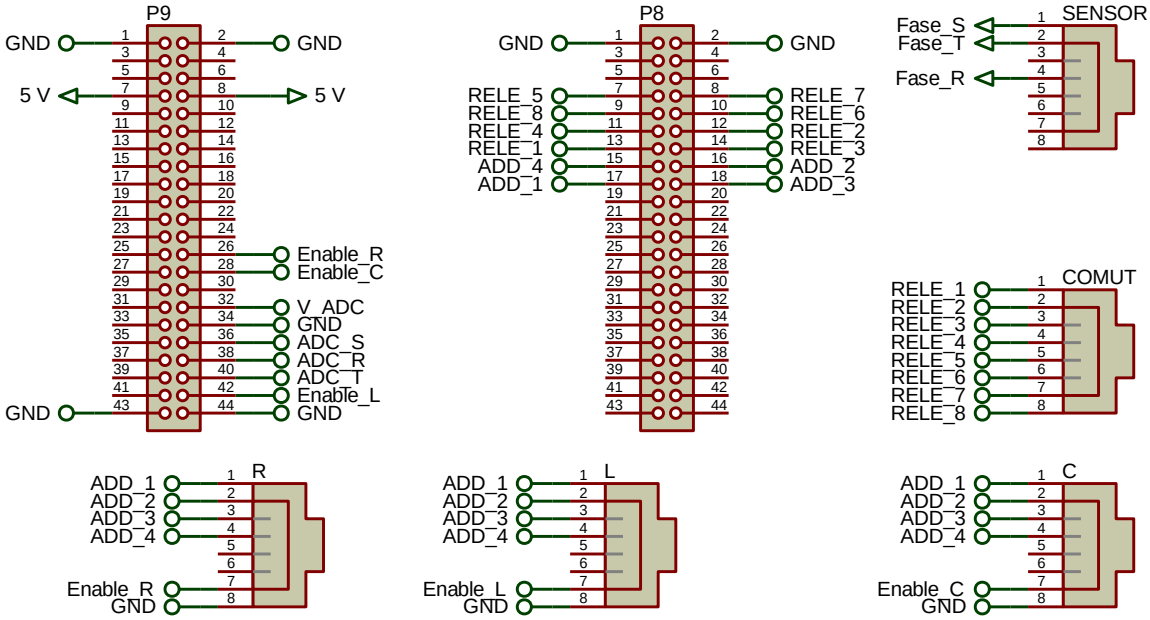
Figura 21 – Funcionamento da PCI 8 relés.



Fonte: Autor.

A Figura 22 mostra os conectores utilizados pela placa desenvolvida, responsável pelo encaminhamento dos sinais processados pelo sistema embarcado.

Figura 22 – Conectores do cape RLC_Load.

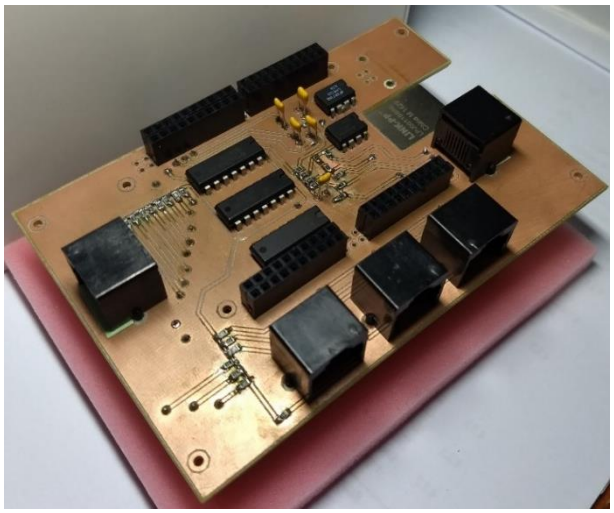


Fonte: Autor.

3.2.3 Placa desenvolvida

A placa desenvolvida para realizar a conexão com as placas de comutação e de medição de tensão, além da conexão com a BeagleBone Black é apresentada na Figura 23.

Figura 23 – Foto da placa desenvolvida, Cape RLC_Load.



Fonte: Autor.

Deve-se tomar cuidado ao adicionar outros sinais não previstos no projeto desta PCI, pois a mesma não foi projetada para receber níveis de tensão superiores aos que são suportados pela BeagleBone Black.

Os componentes necessários para o desenvolvimento desta placa são apresentados na Tabela 3.

Tabela 3 – Componentes utilizados no cape RLC_LOAD.

Componente	**Valor**	**Package**	**Quantidade**
LM6132	-	DIL08	2
BAT54S	-	SOT23-D9	3
7407	-	DIL14	3
Resistor	1 kΩ	0805_RES	4
Resistor	10 kΩ	0805_RES	31
Resistor	3.3 kΩ	0805_RES	3
Resistor	1.5 kΩ	0805_RES	3
Capacitor	100 nF	CAP10	5
Pin 1x10	-	CON10_2X10_US_FCI	8
RJ45	-	RJ45 GLX-S-88M	5
USB B	-	CON4_1X4_USB_B_AM	1

Fonte: Autor.

3.3 DESENVOLVIMENTO DE SCRIPT PYTHON PARA SERVIDOR

A linguagem Python foi escolhida para o desenvolvimento do *script* referente a execução do servidor por parte da BeagleBone, devido ao suporte de bibliotecas dessa linguagem, tanto para controle dos pinos digitais do sistema embarcado, quanto o trabalho com *sockets*.

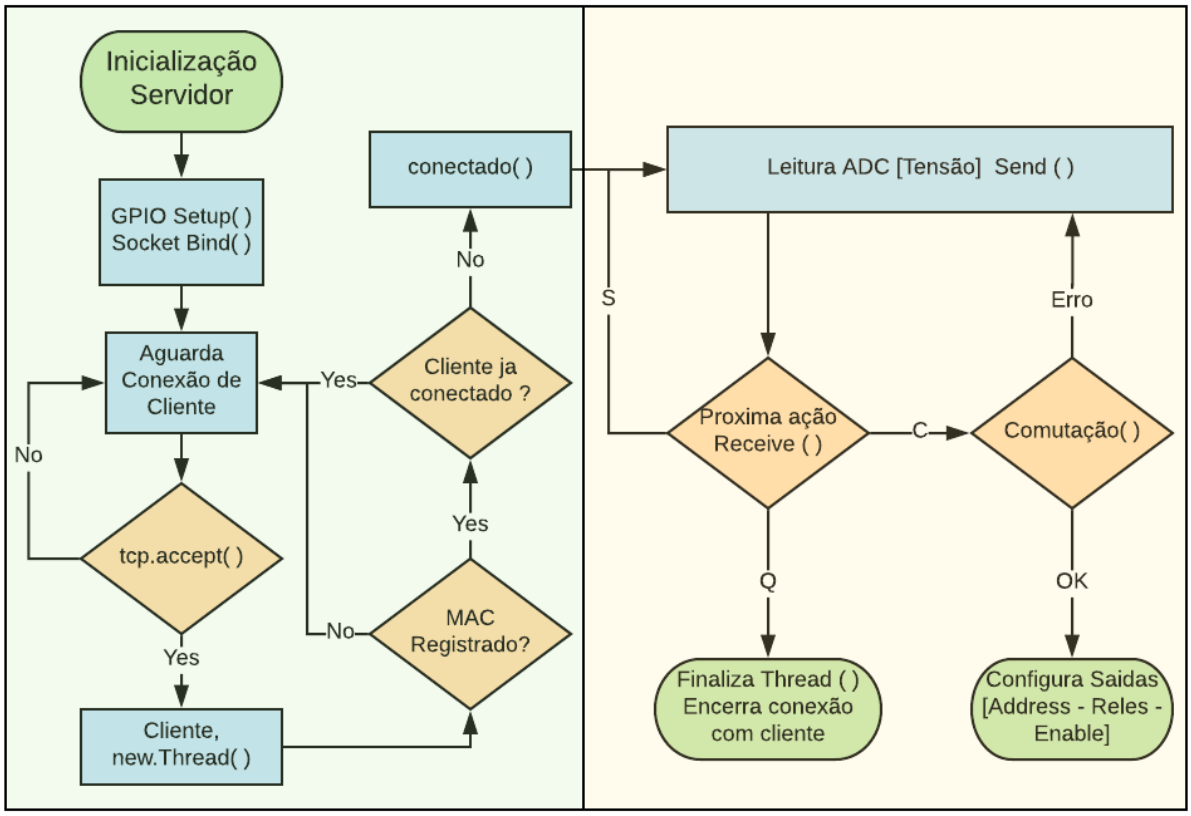
Primeiramente foram definidas as bibliotecas necessárias ao correto funcionamento das funções realizadas no servidor. Após importar as bibliotecas, são definidos e inicializados os pinos de saída digitais e as entradas analógicas, conforme apresentados nos conectores referentes a cape RLC_Load, essas funções são facilitadas com o uso da biblioteca Adafruit_BBIO, que já possui as instruções necessárias, para configurar o *device tree overlay* na BeagleBone. E por fim, segue a criação do *socket* TCP, em que é configurada a porta para recepção das solicitações de conexão por parte dos clientes TCP.

Ao concluir a inicialização, o programa aguarda a conexão de um cliente, não executando nenhuma outra ação. Caso um cliente solicite conexão é aberto um processo paralelo ao principal, que executa a verificação dos requisitos para conexão (endereço MAC de cliente registrado e a não existência de cliente já conectado, para não permitir conflito de dados).

Após isso o cliente é considerado conectado e pode enviar os comandos. O servidor inicia um laço, que alterna envios (variáveis lidas nas entradas analógicas, formato *double*) e recebimentos de dados (*string*), referente a próxima ação a ser executada, podendo ser a comutação, desconexão ou execução normal do programa.

O fluxograma da Figura 24, resume o funcionamento do servidor, representando o fluxo de dados entre as funções executadas. O *script* não será apresentado, devido a futuro registro de propriedade.

Figura 24 – Fluxograma do *script* executado no servidor.



Fonte: Autor.

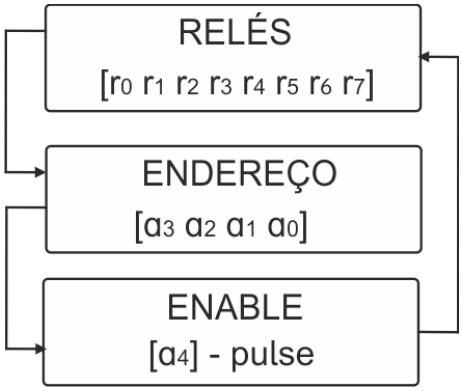
O processo de comutação é o principal objetivo deste trabalho. Assim foi desenvolvido um protocolo especifico para comunicação entre o servidor e as placas de comutação, a seguir é dada uma ênfase em seu funcionamento.

3.3.1 Protocolo RSP (*Relay Switch Protocol*)

Este protocolo define os 13 sinais necessários ao sistema de comunicação entre servidor e placas de comutação, além de definir a ordem de envio dos sinais, e tempo de pulso desses sinais.

Na Figura 25 é apresentado o ordenamento dos sinais enviados, conforme o protocolo RSP.

Figura 25 – Laço de execução do protocolo RSP.

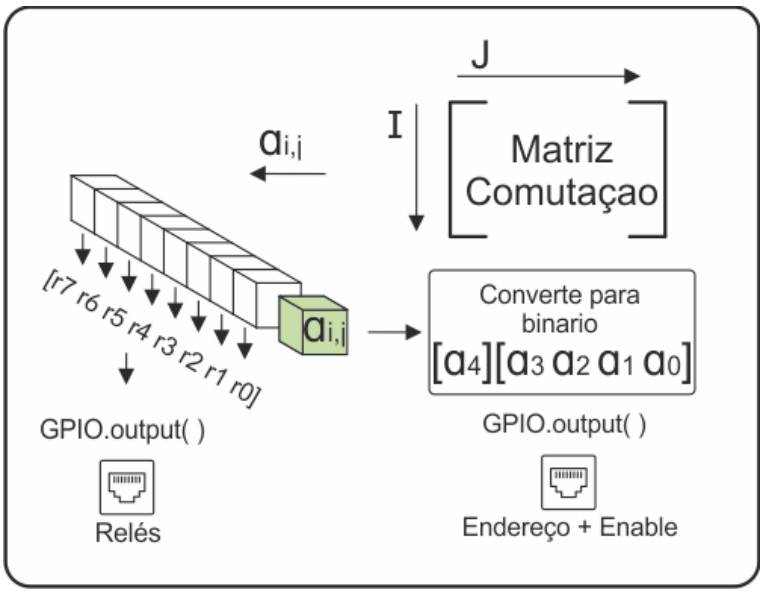


Fonte: Autor

Quando o cliente LabVIEW solicita a comutação, ocorre o envio de uma matriz de comutação, que por sua vez, foi codificada em JSON. Assim é necessário fazer a decodificação da mesma após o recebimento da mensagem TCP, obtendo-se uma matriz de 3 dimensões que contém as informações sobre endereçamento e comutação.

Após receber a matriz de comutação, e fazer a verificação, ocorre o processo descrito na Figura 26, em que as posições referentes aos endereços são convertidas em valores binários de 5 *bits*, e o respectivo vetor de comutação de cada posição é configurado nas portas de saída.

Figura 26 – Processo de configuração dos dados recebidos.



Fonte: Autor.

Como as placas de 8 Relés possuem somente 4 *bits* para endereçamento, e o endereço “0000” é considerado como *standby*, o que permite 15 endereços diferentes, foi utilizado o quinto *bit* da conversão, o mais significativo, para habilitação dos sinais endereçados ao respectivo conector RJ45. Assim permitindo endereçar três series de placas com 4 *bits*, possibilitando agora 45 placas diferentes, além de que, ao se utilizar esse *bit* para habilitar os demais, eles são configurados no mesmo intervalo de tempo, extinguindo possíveis erros de endereçamento ocorridos devido à sincronia dos sinais de endereçamento no comparador de magnitude.

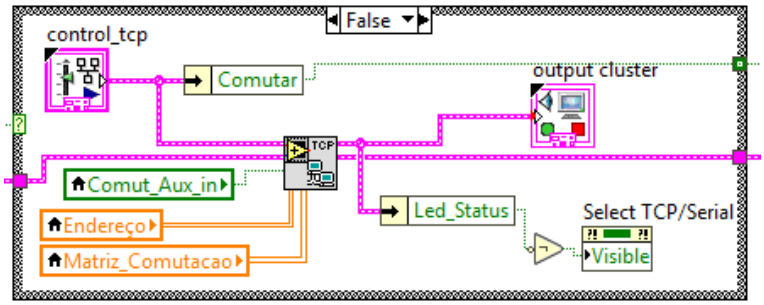
Para tornar o processo mais eficiente, quando o endereço definido é o de *standby*, automaticamente o *script* passa para a próxima posição da matriz, até que as 27 posições tenham sido completadas, seguindo o laço de execução do protocolo RSP.

3.4 DESENVOLVIMENTO DE VI LABVIEW PARA COMUNICAÇÃO TCP/IP

Visando alterar o protocolo de comunicação utilizado, foi desenvolvido um VI especifico para obter os parâmetros calculados anteriormente, que determinam a combinação dos elementos da carga RLC empregados no ensaio, e fazer o envio destes, por meio do protocolo TCP/IP, para o servidor responsável pelo encaminhamento dos dados.

O diagrama de blocos visto na Figura 27, mostra os elementos necessários para o funcionamento do VI, baseado em uma máquina de estados, controlada basicamente por um bloco de controles presentes no painel principal, e dados atualizados por um *buffer* no *loop* de execução do programa.

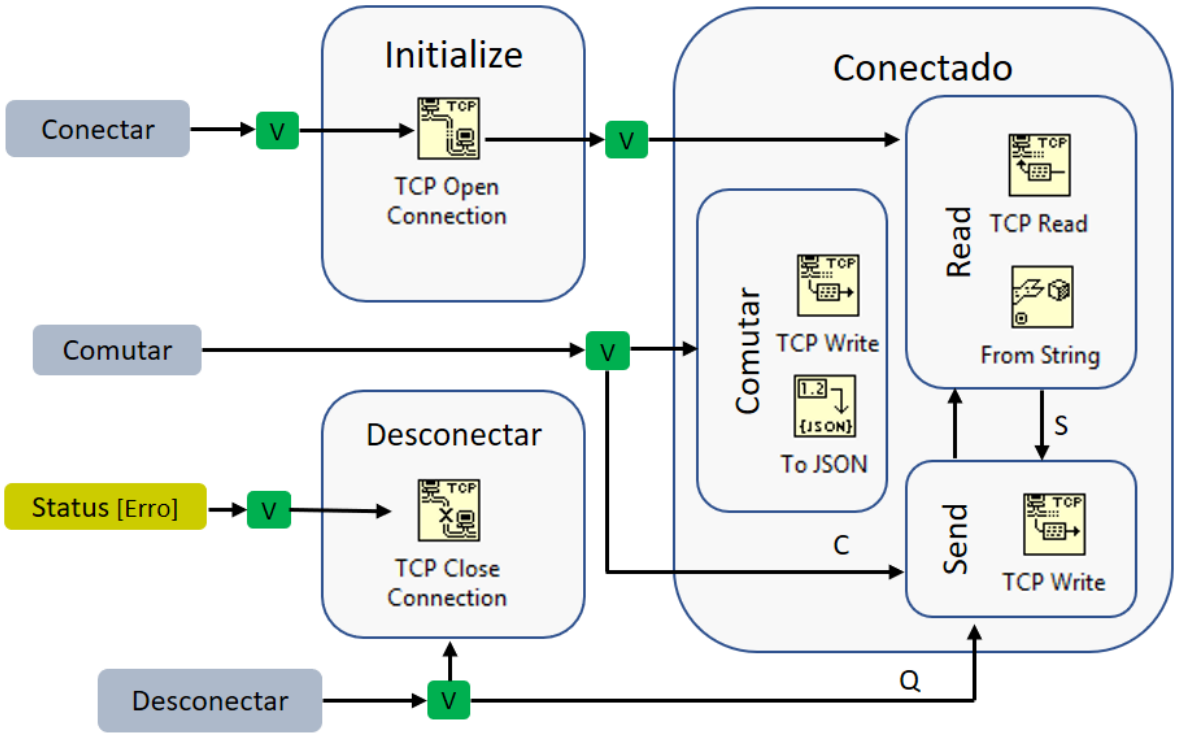
Figura 27 – Diagrama de blocos adicionado ao vi principal.



Fonte: Autor.

Para melhor entendimento da máquina de estados, presente no diagrama de blocos TCP, o processo foi resumido no fluxograma Figura 28, que apresenta os estados e condições detalhados para o funcionamento do *script*.

Figura 28 – Fluxograma de funcionamento do subVI TCP/IP implementado.



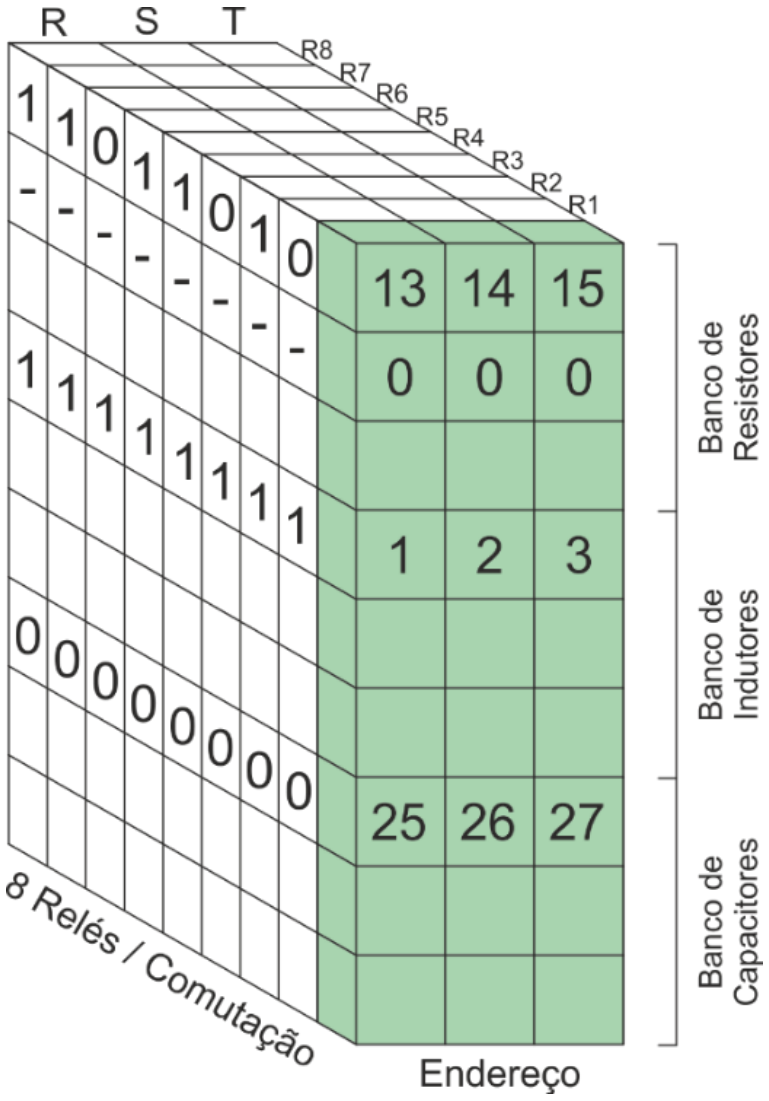
Fonte: Autor.

Por *default*, após o processo ser inicializado, a máquina de estados aguarda a alteração do estado do botão conectar, para tentar conexão com o servidor, caso o servidor esteja *online*, e o endereço MAC do cliente esteja registrado no servidor, ocorre a conexão. No processo, em que servidor e cliente estão conectados, ocorre a troca de dados e é possível solicitar a comutação das cargas ao servidor, pressionando comutar. Caso for solicitada a desconexão, ou o *status* de conexão for falso, é passado para o estado de desconexão, onde o *socket* aberto incialmente é encerrado, e volta-se para a inicialização.

Quando é solicitada a comutação de cargas, o programa LabVIEW faz o envio de uma matriz de comutação que contém 27 posições possíveis, e cada uma dessas posições contém um endereço e um vetor de comutações correspondente, conforme visto na Figura 29.

O endereço independe da posição na matriz e é definido na configuração de cargas presente no programa LabVIEW. Sendo que, por padrão definido no programa, cada coluna desta matriz representa uma fase; as 3 primeiras linhas, posições referentes a cargas resistivas, as 3 seguintes cargas indutivas e por fim as cargas capacitivas.

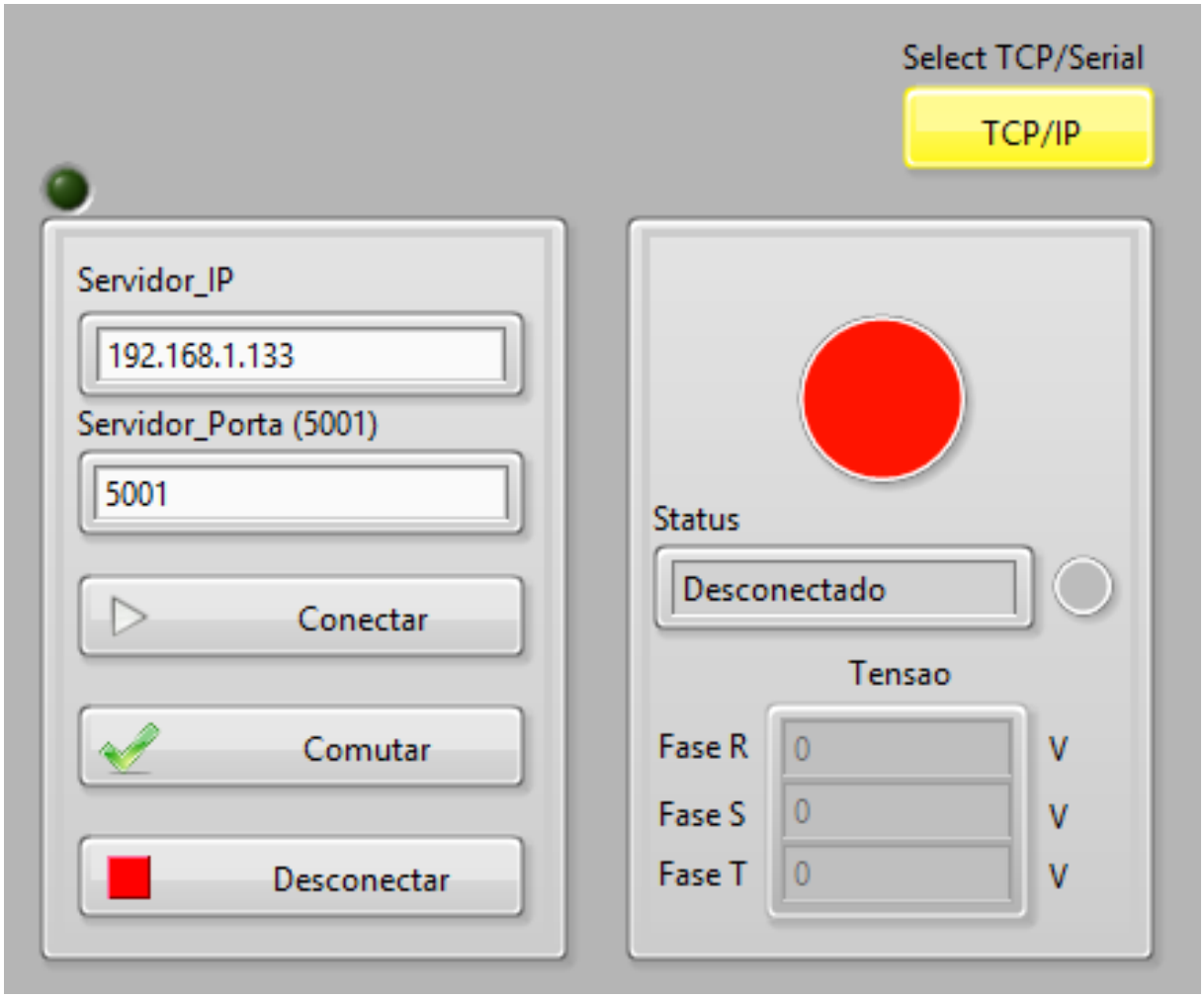
Figura 29 – Matriz de Comutação enviada.



Fonte: Autor.

Para facilitar o controle das ações foi adicionada um ambiente, Figura 30, a interface principal do programa, que permite a seleção do sistema de comunicação, TCP/IP ou Serial, e possui os indicadores de *status* de conexão, e campos para controle.

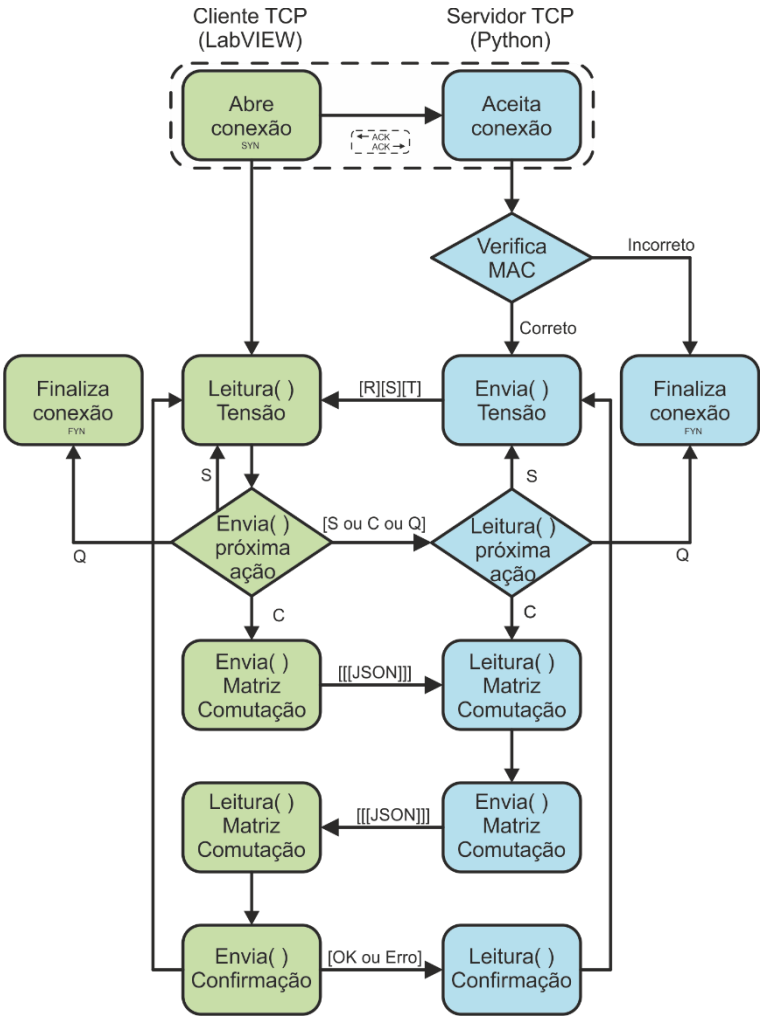
Figura 30 – Interface para controle através do modelo TCP/IP.



Fonte: Autor.

Após apresentar tanto o cliente, como o servidor TCP é exibido um resumo do protocolo de comunicação adotado, conforme Figura 31, que apresenta os dados trocados entre as aplicações desde a abertura da conexão até o encerramento da mesma.

Figura 31 – Protocolo de comunicação cliente/servidor.



Fonte: Autor.

Inicialmente o cliente faz uma requisição de conexão, ao enviar a *flag* SYN para o servidor, se *online* irá enviar uma resposta com a *flag* ACK, confirmando a conexão, e por fim o cliente também envia uma resposta com a *flag* ACK configurada. Agora que ambos estão conectados, por padrão definido no protocolo TCP, irão fazer a verificação da conexão por meio de *flags* a cada envio de dados. Na primeira etapa, foi implementada a verificação do endereço MAC do cliente por parte do servidor, caso este não seja conhecido do servidor a conexão é finalizada. Em seguida, o protocolo entra em um laço, em que o servidor faz o envio das variáveis de tensão adquiridas, em formato *double*, e após o cliente recebe-las envia uma *string* com a próxima ação, selecionada entre manter a leitura de tensão, realizar a comutação ou finalizar a comutação. Caso a comutação for selecionada, o cliente LabVIEW fara o envio da matriz de comutação, o servidor irá recebe-la e fazer o reenvio para verificação por

parte do cliente se está correta, caso afirmativo, o cliente envia uma *string* que direciona o servidor a comutação, caso contrário segue a execução normal, sem realizar a comutação.

3.5 SUMÁRIO

Este capitulo apresentou o desenvolvimento de uma interface de comunicação Ethernet, entre o sistema supervisório e o equipamento controlado, substituindo a atual interface de comunicação, projetada para empregar uma placa de aquisição de dados NI USB – 6501, com conexão serial via USB.

O sistema implementado consiste em uma placa de interface e aquisição de dados, desenvolvida para conexão de uma placa BeagleBone Black com Linux embarcado, a um conjunto endereçável de placas de acionamentos de relés. O *firmware* para controle do BBB foi desenvolvido em Python, e executa um servidor que permite a comunicação, via *socket* TCP/IP, do instrumento virtual (VI) desenvolvido no *software* LabVIEW para execução no computador, podendo ser empregado em rede associado a múltiplos equipamentos.

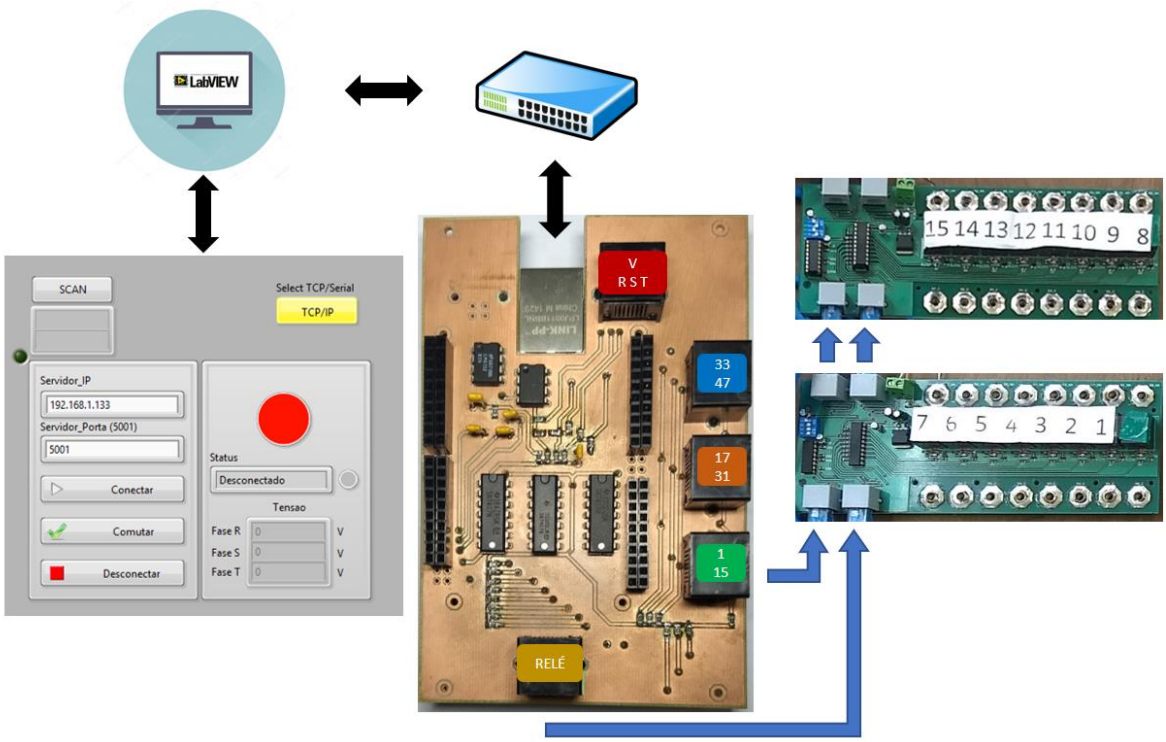
4 VALIDAÇÃO EXPERIMENTAL

Um protótipo experimental foi implementado no Laboratório de Ensaios Fotovoltaicos do Instituto de Redes Inteligentes (INRI) para validação prática do sistema de comunicação Ethernet proposto.

Os sistemas desenvolvidos neste trabalho foram testados integrados aos subsistemas previamente desenvolvidos, e adaptado ao software desenvolvido por (LENZ, 2018), com o auxílio de uma bancada de testes, para futura instalação junto ao sistema de ensaio de anti-ilhamento.

Utilizando um Switch, para criar uma rede local, foi realizada a conexão, via Ethernet, entre as aplicações desenvolvidas. O esquemático, apresentado na Figura 32, ilustra a montagem da bancada de testes utilizada para validação experimental do protótipo.

Figura 32 – Esquemático da bancada de testes, utilizado para validação experimental.



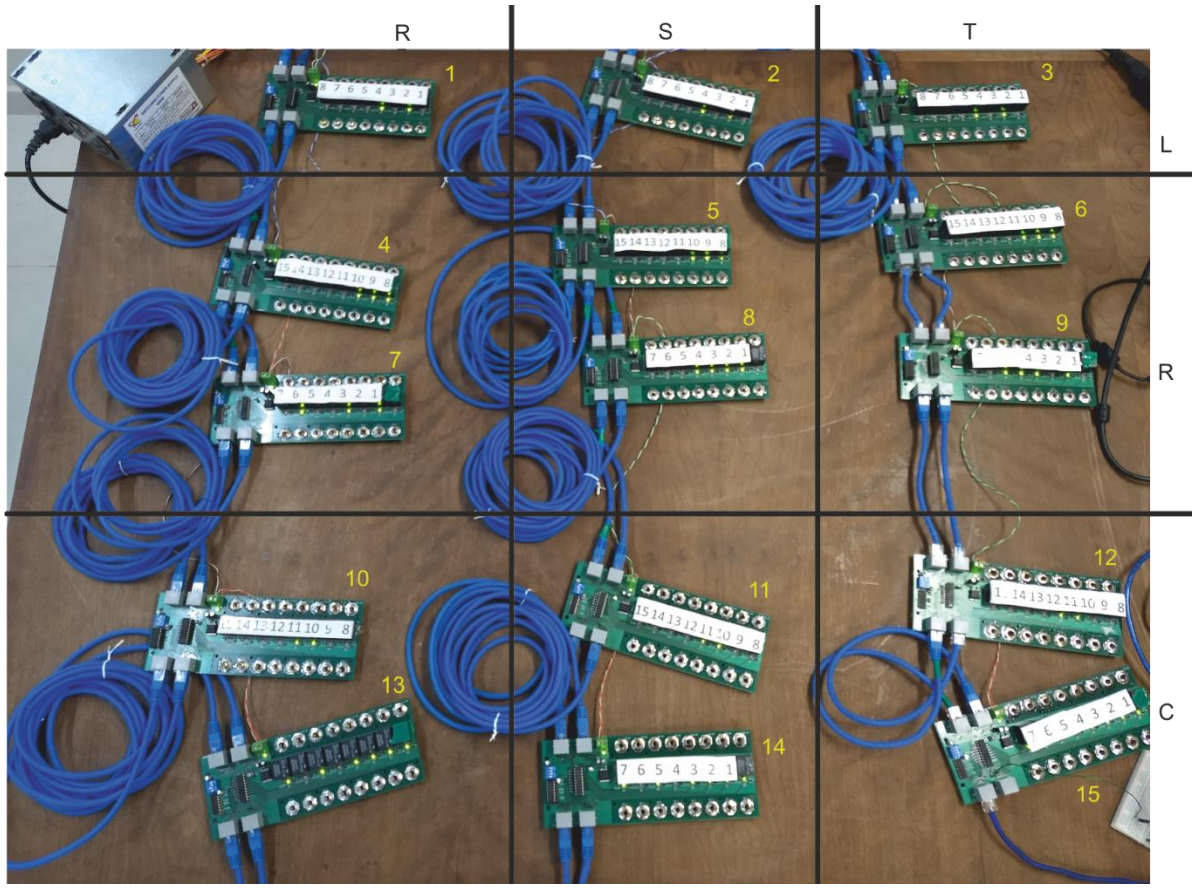
Fonte: Autor.

A comunicação entre as aplicações (LabVIEW e Python) é realizada através de sockets TCP, descritos previamente, enquanto que a comunicação com as placas de comunicação utiliza o protocolo RSP, desenvolvido neste trabalho.

4.1 COMUNICAÇÃO DO SERVIDOR COM PLACAS DE COMUTAÇÃO

Para realizar a validação do sistema implementado foram utilizadas 15 placas de comutação conectadas em série, dispostas conforme Figura 33, sendo que em um dos lados, estas foram conectadas a cape RLC_Load, responsável por enviar os sinais do servidor, e na outra ponta foram analisados os sinais, para verificar possíveis atenuações. Essa verificação foi realizada com o auxílio de uma placa Arduino UNO, atuando como analisador lógico.

Figura 33 – Foto da bancada de testes implementada.

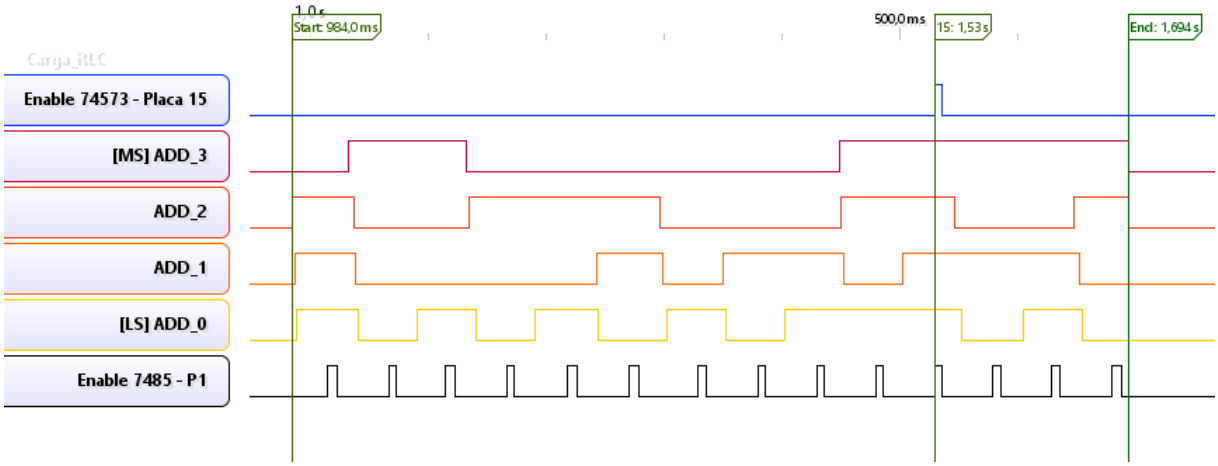


Fonte: Autor.

Devido a limitação, de 6 canais, imposta por esse analisador, foram medidos somente, os sinais de endereçamento, enquanto que os sinais de comutação foram verificados visualmente por meio dos relés, dada a disposição das placas de comutação.

Na Figura 34, é mostrado um ciclo completo de comutação, obtido com o analisador digital. O período mostrado tem duração inferior a 1 segundo em que são verificados os sinais de endereçamento e os pulsos que habilitam a comparação. Nesse caso, foi verificado o sinal de saída do comparador, na placa com endereço 15, para validar o processo, sendo possível verificar que o funcionamento foi correto, pois habilitou a comutação dos relés corretamente. Os 4 *bits* de endereçamento (ADD) iniciam pelo endereço 7, conforme definido no sistema supervisório, e passam por todos os endereços de 1 a 15.

Figura 34 – Análise de sinais em um ciclo de comutação.



Fonte: Autor.

Durante essa fase de testes também foi utilizada uma fonte CC gerando os sinais de 0 a 5 Volts ao circuito de aquisição das tensões, que tem o intuito de evitar comutações sob carga. O funcionamento foi adequado, não permitindo a comutação do sistema, quando houve à verificação de uma tensão maior que os limites definidos. Ressaltando que a leitura de 0 a 5 Volts equivale ao circuito de entrada da placa Cape_RLC, que por sua vez é conectada a placa de medição, para tensões de ensaio fase-neutro de 220 V.

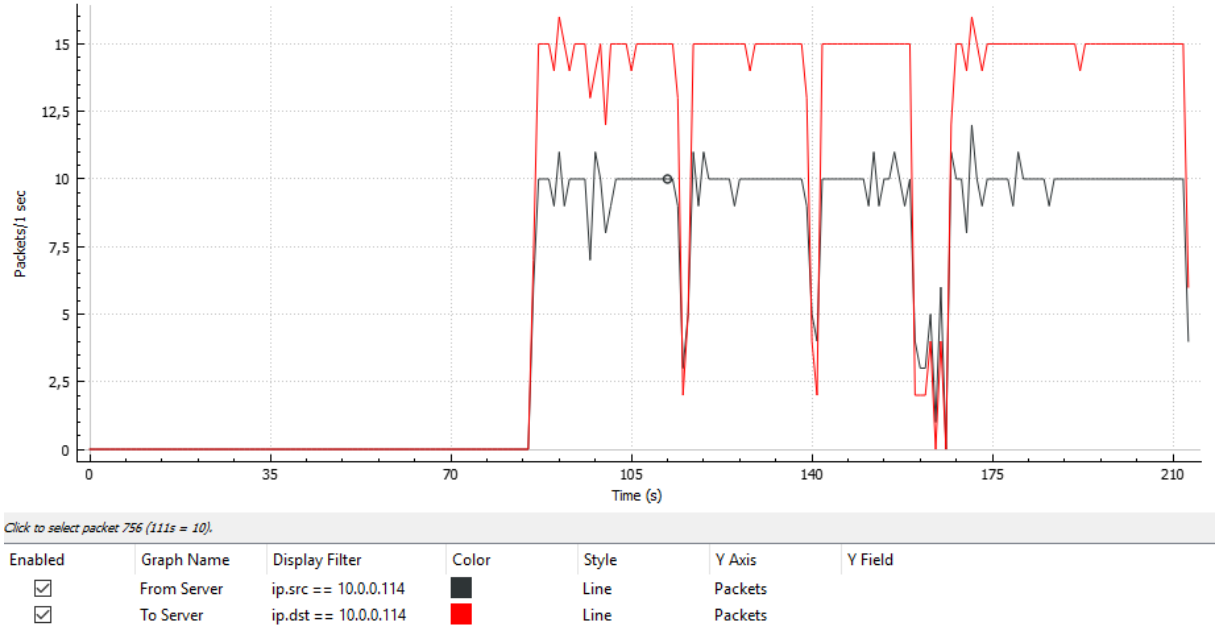
4.2 PROTOCOLO DE COMUNICAÇÃO

Como no laboratório existem diversos equipamentos conectados a uma rede local, para evitar conflitos de endereçamento lógico, e consequentemente a sobrecarga da rede, foram configurados IPs estáticos em cada equipamento. Logo, a BeagleBone Black também foi configurada com um IP fixo na rede. Assim foi definido

o endereço para comunicação dos clientes LabVIEW, com o servidor de controle da carga RLC.

Para observar o comportamento do protocolo de comunicação entre o programa LabVIEW e a BeagleBone, e a interferência no trafego da rede foi utilizada a ferramenta Wireshark, que permite a monitoração e visualização dos pacotes que trafegam pela rede. O resultado do monitoramento é mostrado na Figura 35.

Figura 35 – Pacotes enviados e recebidos pelo servidor.



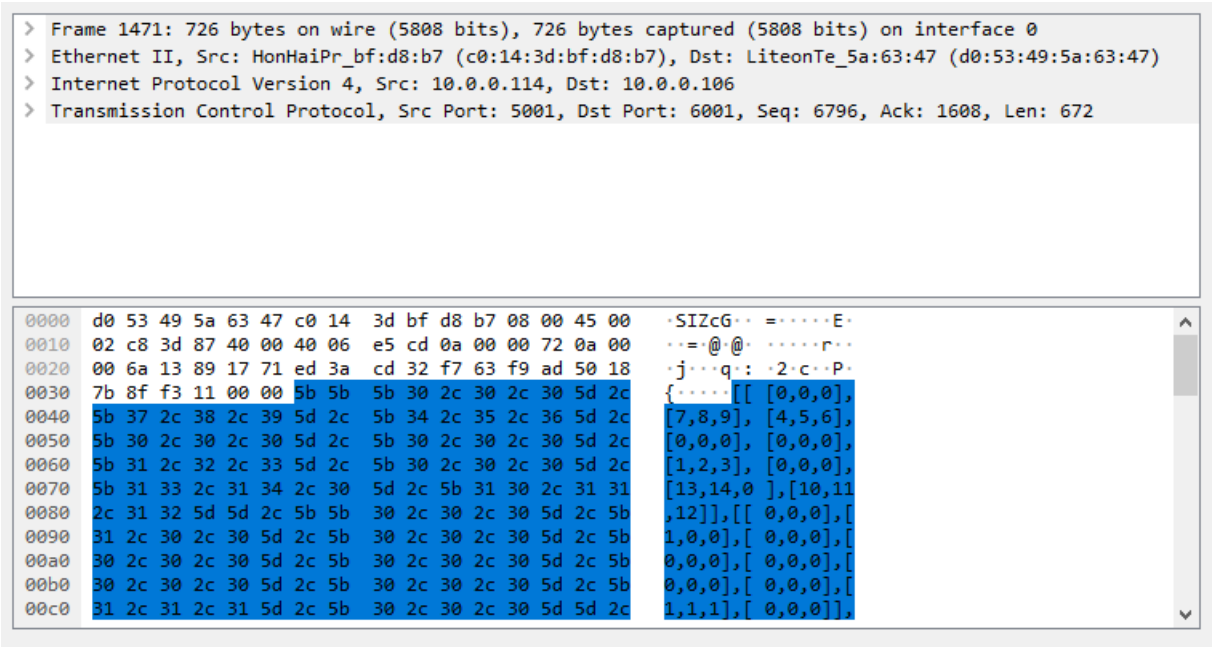
Fonte: Autor.

Neste gráfico é mostrado o número de pacotes por segundo recebidos pelo servidor, em vermelho, e enviados pelo mesmo, em preto. Observa-se que nos momentos em que ocorrem comutações há uma redução do número de pacotes trafegados, dado que o protocolo aguarda o encerramento do processo de comutação para reiniciar o envio de dados. No restante do tempo, a troca de dados é mantida praticamente constante. Visto que estes pacotes possuem tamanho máximo de 1024 *bytes*, constata-se que a carga máxima na rede utilizada por este protocolo é de 15 kb/s, o que interfere pouco no trafego máximo da rede local.

Além disso, devido ao *keep alive* que foi implementado na aplicação LabVIEW, caso nenhum pacote Ethernet seja recebido em um intervalo de 5 segundos, o mesmo finaliza a conexão.

A Figura 36 mostra um quadro Ethernet, que encapsula os dados de comutação. O endereço “10.0.0.114”, representa o endereço de origem, ou seja, o cliente LabVIEW, que realizou o envio do pacote. Já o endereço “10.0.0.106” e a porta 5001 representam, respectivamente, o endereço utilizado pelo protocolo IP e a porta configurada no protocolo TCP empregados para o recebimento dos dados enviados pelo cliente ao servidor.

Figura 36 – Frame enviado com matriz de comutação.



Fonte: Autor.

O *frame* enviado possui 726 *bytes*, sendo 672 de dados, ou seja, a matriz de comutação é contida em um único quadro Ethernet.

4.3 SUMÁRIO

Os testes experimentais foram realizados para validar a comunicação tanto entre servidor e placas de comutação, como entre servidor e interfaces supervisórias. As análises realizadas com o auxílio do software *Wireshark* e analisador digital, demonstraram que os resultados obtidos são adequados à necessidade da aplicação.

5 CONCLUSÃO

A implementação de um sistema para automatizar um dos processos necessários para o ensaio de anti-ilhamento envolveu a integração de vários subprojetos que abordaram pontos específicos do sistema. O presente projeto é complementar aos anteriores, tendo o objetivo de melhorar a funcionalidade de um sistema já existente.

O protocolo de comunicação desenvolvido se mostrou aplicável a outros processos que envolvam controle de variáveis. Comparado com o protocolo que era utilizado anteriormente, baseado em interface USB, este se mostra vantajoso, por poder ser acessado através da internet. Dessa forma, dispensa-se a conexão física direta com o computador responsável pelo controle, o que permite a qualquer computador que esteja na mesma rede e possua o programa apropriado realizar comunicação com o servidor, e consequentemente o controle da carga RLC.

O uso da BeagleBone Black com Linux embarcado mostrou-se uma solução adequada para a comunicação via Ethernet. Constata-se que a solução proposta pode ser expandida a outros processos e ensaios realizados no laboratório, uma vez que a solução proposta é genérica e aplicável a diferentes processos.

Após essas considerações pode-se afirmar que foi possível alcançar o objetivo proposto. Destaca-se que os conhecimentos necessários para a implementação de um sistema como esse que depende da integração entre o *software* e o *hardware.*

O projeto desenvolvido se mostrou funcional e atingiu o objetivo proposto, mas há diversas melhorias que podem ser implementadas. Tais como:

- Modularização das entradas analógicas, permitindo adicionar outras variáveis de controle ao processo, como a verificação da temperatura interna da carga RLC, medição da corrente, para calcular a potência por fase;
- Criação de interface, para uso de um visor *touch-screen*, junto a BeagleBone Black, que permita averiguar informações diretamente na carga RLC;
- Melhoria no protocolo de comunicação, incluindo mais redundâncias para garantir o correto envio dos comandos.

REFERÊNCIAS

ABNT. **ABNT NBR IEC 62116: Procedimento de ensaio de anti-ilhamento para inversores de sistemas fotovoltaicos conectados à rede elétrica**. Rio de Janeiro, Brasil. 2012.

ALMEIDA NETO, J. C. D. S. **Avaliaçao de conformidade de inversores para micro e mini geraçao fotovoltaica: a implementaçao da NBR 16150 e NBR IEC 62116**. Sao Paulo. 2017.

ALMEIDA, M. **Qualificaçao de sistemas fotovoltaicos conectados a rede**. USP. Sao Paulo. 2012.

ANEEL. **Micro e minigeraçao distribuida. Sistemas de compensaçao de Energia Elétrica.** Brasilia. 2014.

ASCO POWER. Asco Power Technologies. **Load Bank Basics**, 2018. Disponivel em: <https://www.ascopower.com/en-us/resource/our-resources/articles/load-bank-basics/>. Acesso em: 02 Novembro 2018.

BAXTER, M. Insecure Lab. **Packet Header Analysis**, 2017. Disponivel em: <http://www.insecure.in/packet_header_analysis.asp>. Acesso em: 29 Outubro 2018.

BEAGLEBOARD. **BeagleBone Black**, 2018. Disponivel em: <http://beagleboard.org/black>. Acesso em: 02 Outubro 2018.

COELHO, P. H. A. **Estudo de tempo de resposta do GPIO da BeagleBone Black**. Uberlandia. 2017.

COLEY, G. **BeagleBone Rev A6 System Reference Manual**. Dallas. 2012.

COMER, D.; STEVENS, D. L. **Interligação em Rede com TCP/IP**. 6. ed. Rio de Janeiro: Prentice Hall, 1999.

DRUMMOND, J. **PHY 406 - Microprocessor Interfacing Techniques**. University of Toronto. Toronto. 1996.

FIGUEIRA, H. H. **Sistema automatizado para ensaio de inversores fotovoltaicos fotovoltaicos conectados à rede em acordo com normatizaçao brasileira**. UFSM. Santa Maria. 2016.

HBM. **Drivers para conectar Sistemas de Aquisição de Dados da HBM ao LabVIEW**, 2014. Disponivel em: <https://www.hbm.com/pt/4404/drivers-para-conexao-de-sistemas-daq-hbm-ao-LabVIEW/>. Acesso em: 05 Dezembro 2018.

INMETRO. **Portaria N° 118**. Rio de Janeiro. 2015.

JOBESOLAR. **Inversor ksg 1.5kw SM**, 2017. Disponivel em: <http://jobesolar.com.br/produtos/inversores-fotovoltaicos/inversor-ksg-1-5kw-sm/>. Acesso em: 29 Novembro 2018.

LENZ, C. D. S. **Relatorio de Projeto de Hardware: Placa Verificadora e medidora de Tensão**. UFSM. Santa Maria. 2018.

MACIEL, F. Softwarelivre. **O que é Linux Embarcado**, 24 Maio 2014. Disponivel em: <http://softwarelivre.blog.br/2014/05/24/o-que-e-linux-embarcado/>. Acesso em: 12 Outubro 2018.

NATIONAL INSTRUMENTS. **Instrumentação Virtual**, 2009. Disponivel em: <http://www.ni.com/white-paper/4752/pt/#toc3>. Acesso em: 30 Setembro 2018.

PAULEY, M. **TCP State Diagram**, 2008. Disponivel em: <https://en.wikipedia.org/wiki/File:Tcp_state_diagram.png>. Acesso em: 30 Novembro 2018.

PERRONE, F. BUCKNELL UNIVERSITY. **Interprocess Communication:** TCP sockets, 2018. Disponivel em: <http://www.eg.bucknell.edu/~cs315/Spring12/wordpress/?page_id=187>. Acesso em: 22 Novembro 2018.

PIECZOKI, R. **Instrumentos Virtuais Baseados no LabVIEW e Placa de Som**. UPF. Passo Fundo. 2015.

PIETTA, L. P. J. **Relatorio de projeto de Hardware: Placa com 8 relés e endereçamento digital**. UFSM. Santa Maria. 2017.

RICHARDSON, M. **Getting Started with BeagleBone:** Linux-powered Electronic Projects with Python and JavaScript. Sebastopol: Maker Media, Inc., 2013.

SAUAIA, R. L. **Geração Distribuída Solar Fotovoltaica:** Benefícios Líquidos ao Brasil. Seminario Internacional de Micro e Minigeraçao Distribuida. Brasilia: ANEEL. 2018.

SILVA, H. T. D. **Estudo sobre a interaçao de metodos anti-ilhamento para sistemas fotovoltaicos conectados à rede de distribuiçao de baixa tensao com multiplos inversores**. USP. Sao Paulo. 2016.

SOARES, L. F. G.; LEMOS, G.; COLCHER, S. **Redes de Computadores:** das LANS, MANS e WANS as redes ATM. Rio de Janeiro: Campus, 1995.

SOLAR, U. **Sistemas Fotovoltaicos Conectados à Rede**, 2017. Disponivel em: <http://universosolar.com/sistemas-fotovoltaicos-conectados-a-rede/>. Acesso em: 20 Novembro 2018.

TAMANINI, L. C. **Projeto de Sistemas Fotovoltaicos conectados à Rede Elétrica**. UEL. Londrina. 2017.

TIOMNO, M. T. **Plano Decenal de Expansao de Energia 2023.** Rio de Janeiro: EPE. 2013.

TRETER, M. E. **Desenvolvimento de um sistema de aquisiçao de dados e monitoramento para diagnostico de centrais de geraçao forovoltaica**. UFSM. Santa Maria. 2016.

ZILLES, R.; MACEDO, W. N. **Sistemas fotovoltaicos conectados à rede elétrica.** USP. Sao Paulo. 2012.

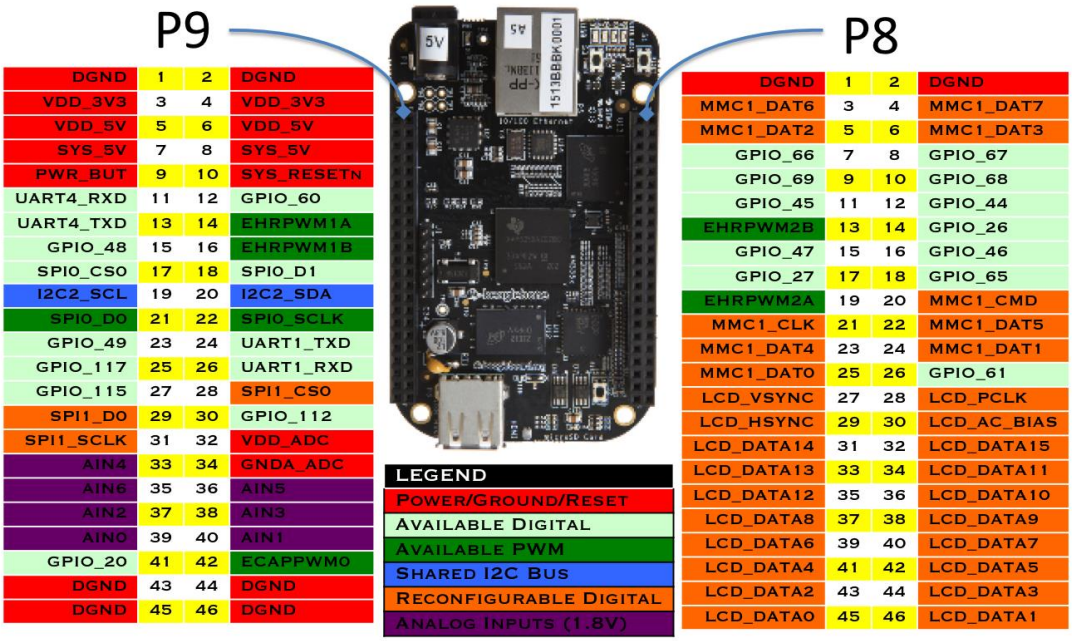
APÊNDICE A – CONFIGURAÇÃO DOS PINOS DA BEAGLEBONE BLACK

Os pinos de entrada e saída da BeagleBone Black são configuráveis de diversas formas possíveis, cada configuração diferente representa um *device tree* diverso. O *device tree* original da BeagleBone Black é apresentado na Figura 37, sendo esta a configuração dos pinos de expansão da placa durante o *boot* do sistema. Caso a aplicação demande uma outra configuração, essa deve ser sobrescrita por outro *device tree*.

É possivel alterar os arquivos (terminados em .dtb) que compõem o *device tree*, no diretório /boot/dtbs, contudo seria necessário reiniciar o sistema após cada mudança, assim um caminho mais flexível é através da utilização de *device tree overlays*, que sobrescrevem as configurações do *kernel* com o sistema sendo executado, através do capemgr.

No caso deste trabalho, como foram utilizadas as bibliotecas Adafruit_BBIO, que já possuem as instruções necessárias, permitindo sobrescrever o *device tree*, durante a execução, com as funções de setup( ). Foi mantido o *device tree* original da BeagleBone Black durante o *boot* do sistema.

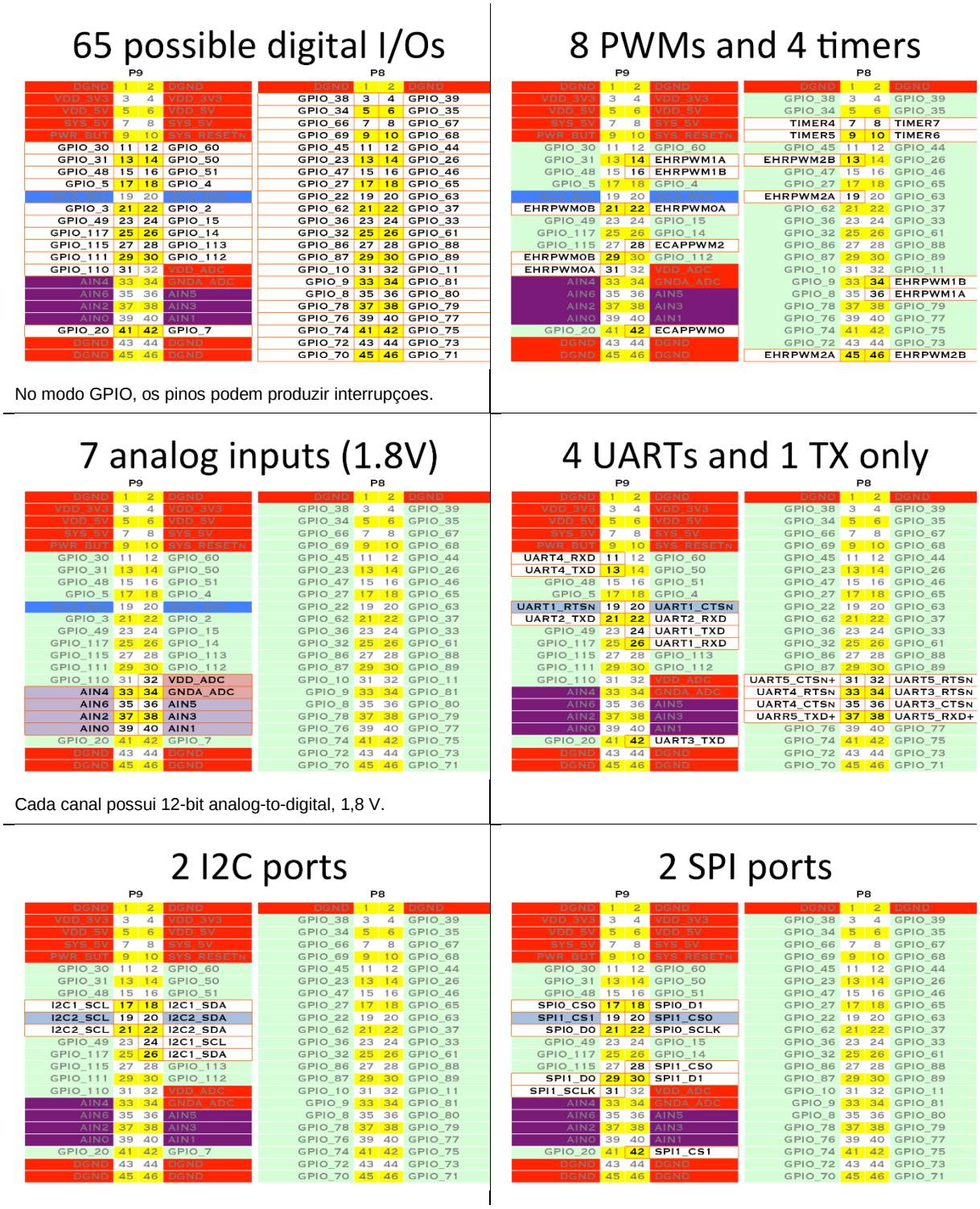
Figura 37 - Device tree original da BeagleBone Black.

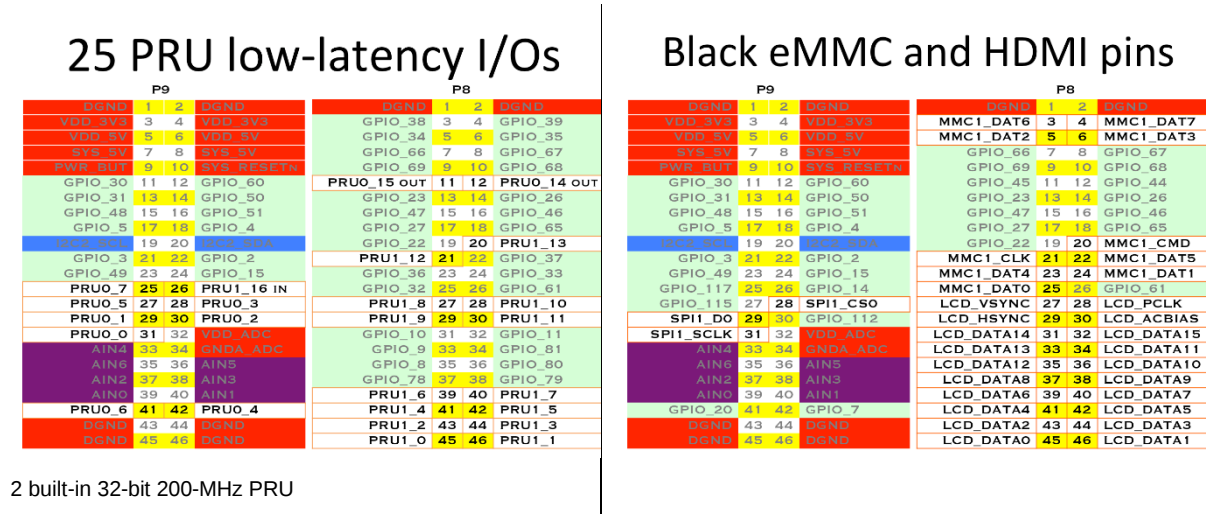


Fonte: (BEAGLEBOARD, 2018).

Além da configuração inicial é possível alterar a função dos pinos. Na Figura 38 é apresentado um resumo das possíveis configurações da BeagleBone Black.

Figura 38 – Possíveis configurações dos pinos da BeagleBone Black.





Fonte: Autor, adaptado de (BEAGLEBOARD, 2018).

APÊNDICE B – CONFIGURAÇÃO BEAGLEBONE BLACK

Os procedimentos empregados para configuração da BeagleBone Black para uso no sistema proposto são aqui detalhados.

B.1. Formatar BeagleBone

O primeiro passo antes de configurar a BeagleBone foi atualizar a versão do Linux. Para isso é necessário gravar a imagem em um cartão *uSD*, com espaço sufuciente, que é utilizado pela BeagleBone para inicialização do sistema operacional

Há então dois tipos de imagens disponíveis, as que manterão o sistema no cartão de memória de onde ele rodará, e as que possuem um *script* que assim que alimentarmos a BeagleBone Black, irá copiar todo o sistema do cartão de memória para a memória *Flash* (eMMC) que está na placa, permitindo assim que o cartão de memória seja removido, ou até mesmo usado para espaço extra para o sistema.

Para se obter uma imagem atualizada é possível acessar https://rcn-ee.com/rootfs/, localizar a seção Flasher e fazer *download* da imagem BBB-eMMC-flasher selecionada (no Linux pode-se utilizar o comando wget).

Copiar a imagem para o cartão SD, pode-se utilizar um *software* “writer image” ou utilizar o comando:

---

*xzcat BBB-eMMC-flasher-**“...”**.img.xz | sudo dd of=/dev/sdX*

Inserir o cartão micro SD na BeagleBone, manter pressionado o botão próximo ao cartão micro SD e após energizar a BeagleBone (utilizar fonte de alimentação externa). Quando todos os LED da BBB forem acionados pode-se soltar o botão. Os LED devem piscar de forma sequencial, um por vez, durante todo o processo de instalação. No momento que todos os LED permanecerem acessos a instalação está concluída, assim, pode-se desenergizar a BBB, remover o cartão SD e então utilizar normalmente.

Para executar o *boot* por uma imagem diretamente no *slot* SD, sem sobrescrever a imagem original, é necessário, que a imagem não esteja configurada para gravar na memória *flash*. É possível encontrar imagens no mesmo *link* citado anteriormente, no

diretório microSD. Também é possível alterar alguns arquivos, na imagem gravado no cartão SD e torna-la gravável na memória *flash*.

Para isso é necessário editar o arquivo **/boot/uEnv.txt** na partição do linux e descomentar a linha **cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh**.

B.2. Acesso SSH

Conectar a BBB em um roteador, ou conectar com o cabo USB (neste caso o IP padrão é 192.168.7.2) e no terminal inserir o comando

```
ssh user@host ; user: (ubuntu, debian...), host: (IP da BBB)
```

O acesso SSH permite que outras aplicações tenham acesso ao sistema instalado na BeagleBone, permitindo utilizar *cross-compile* com Eclipse, Netbeans ou acessar um terminal através do Putty no Windows. Caso não possua um *cape* LCD ou adaptador microHDMI é possível utilizar o acesso SSH juntamente a um *software* VNC para visualizar o *display* da BeagleBone.

B.3. Importando API Python

A distribuição Debian 9.4 "Stretch" IoT já vem pré-compilada com Python e algumas bibliotecas e é recomendada para a utilização em conjunto com a API Adafruit_BBIO desenvolvida em Python para habilitar as saídas como GPIO, ADC, PWM.

A instalação desta API é feita através dos comandos:

```
sudo ntpdate pool.ntp.org
sudo apt-get update
sudo apt-get install build-essential Python-dev Python-pip -y
sudo pip install Adafruit_BBIO
```

B.4. Configuração de IP estático

A configuração das interfaces de rede na distro Debian Stretch é realizada por meio do serviço connman. Assim selecionando a interface de rede correta, pode-se configurar um IP estático para a interface de rede Ethernet, com o seguinte comando.

```
sudo connmanctl config Ethernet_78a504cdf7e5_cable --ipv4 manual 192.168.137.180
255.255.255.0 192.168.137.1 --nameservers 8.8.8.8
```

B.5. Configuração para rodar scripts na inicialização

Para executar um script na inicialização é necessário criar um arquivo .service no diretório /etc/systemd/system/, com o seguinte formato:

```
[Unit]
After=network.service

[Service]
ExecStart=/home/debian/_rlc.sh

[Install]
WantedBy=default.target
```

No bash criado é inserido o comando para executar o script. No caso de um script Python: “sudo Python name.py”. Após isso, são dadas as permissões de execução aos arquivos.

```
chmod +x /home/debian/_rlc.sh
chmod +x /home/debian/name.py
chmod 664 /etc/systemd/system/arquivo.service
```

Por fim é necessário instalar o servisse unit criado e reinializar o sistema.

```
systemctl daemon-reload
systemctl enable arquivo.service
sudo reboot
```