CENTRO UNIVERSITÁRIO DE BRASÍLIA - UNICEUB
FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

GUILHERME PIMENTA BARRETO

SISTEMA DE MONITORAMENTO E CONTROLE DE CONSUMO DE ENERGIA ELÉTRICA PARA UMA RESIDÊNCIA

Orientador: Prof. MSc. Luciano Henrique Duque

Brasília
2015

GUILHERME PIMENTA BARRETO

SISTEMA DE MONITORAMENTO E CONTROLE DE CONSUMO DE ENERGIA ELÉTRICA PARA UMA RESIDÊNCIA

Trabalho de Conclusão de Curso apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito para obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. MSc. Luciano Henrique Duque

Brasília

2015

GUILHERME PIMENTA BARRETO

SISTEMA DE MONITORAMENTO E CONTROLE DE CONSUMO DE ENERGIA ELÉTRICA PARA UMA RESIDÊNCIA

Trabalho de Conclusão de Curso apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito para obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. MSc. Luciano Henrique Duque

BANCA EXAMINADORA

_____

Prof. MSc. Luciano Henrique Duque
Orientador

_____

Prof. MSc. Nilo Sérgio Soares Ribeiro

_____

Prof. MSc. Henrique Marra Taira Menegaz

Brasília

2015

AGRADECIMENTOS

Ao longo do desenvolvimento deste trabalho, assim como no decorrer do curso, muitos generosamente cederam seu tempo e esforços para tornar esta realização possível. Diversos foram os desafios encontrados e afirmo, com certeza, que não poderia os ter ultrapassado sem o apoio de minha família, amigos e colegas.

Primeiramente, agradeço a Deus por abençoar-me com a vida, saúde e com a possibilidade de estar concluindo este curso, além de me proporcionar pais maravilhosos e amigos incríveis.

Aos meus pais, agradeço cada minuto dedicado a mim, minha criação, educação, por todo amor, cada gesto e pensamento. Agradeço a compreensão e as ajudas proporcionadas neste período e em todos que ainda estão por vir. Tenho muito orgulho e alegria de ser filho de vocês.

À minha família, também agradeço o apoio, amor e amizade em todos os períodos do curso, assim como em toda a minha vida. A família é o bem mais precioso que existe.

Agradeço aos meus amigos e colegas de turma que me acompanharam ao longo do curso, por todo o auxílio, brincadeiras e compreensão proporcionadas. Que possamos manter esta amizade por muito tempo. Muitos tomaram outros caminhos ou não puderam chegar até o fim. A eles, desejo toda a força possível, pois são pessoas competentes e terão sucesso em suas escolhas.

Para meus amigos da Monitoria, agradeço a amizade, convivência e as ajudas compartilhadas por todo o período em que participei das atividades. Sem a assistência de alguns, não poderia ter concluído este projeto. Que este ambiente de solidariedade continue iluminando a nossa vida.

Agradeço ao meu orientador pelo apoio dado ao longo do projeto, em tarefas desempenhadas na Monitoria e em sala de aula. Ao meu coordenador, agradeço a preocupação e apoio fornecidos a mim e aos meus colegas, além da organização e disciplina necessárias nesta posição. A todos os professores com quem tive a oportunidade de aprender, agradeço o esforço e atenção, assim como a persistência em cumprir seu papel social. Bons professores são inestimáveis.

Agradeço a todos que estiveram comigo e puderam compartilhar parte de suas vidas. Para todos, eu ofereço um grande Obrigado!

CITAÇÃO

“Os grandes feitos são conquistados não pela força, mas pela perseverança ”

Samuel Johnson

SISTEMA DE MONITORAMENTO E CONTROLE DE CONSUMO DE ENERGIA ELÉTRICA PARA UMA RESIDÊNCIA

RESUMO

Esse trabalho propõe um sistema de monitoramento e controle do consumo de energia de duas cargas monofásicas, comumente utilizadas em residências, referentes a iluminação e condicionamento ambiental. O monitoramento se dá a partir da leitura de sensores de tensão e corrente conectados aos equipamentos monitorados e o controle das cargas pelo acionamento de interruptores eletromecânicos. O projeto utiliza conceitos de controle automatizado, de instalações elétricas e conceitos da área crescente de eficiência energética para abordar os temas relacionados a controle e monitoramento de energia. A parte do projeto prático responsável por medir os valores de consumo utiliza a plataforma *Arduino* para realizar a leitura, através de biblioteca especifica. Os valores são apresentados em uma interface gráfica, implementada utilizando a plataforma *Raspberry,* onde gráficos de consumo também são gerados. Os sinais de controle responsáveis pelo acionamento manual e automático das cargas, assim como a leitura dos valores, são enviados por uma interface serial de um cabo USB, utilizada para conectar os leitores ao sistema principal.

Palavras-chave: Consumo de energia, Monitoramento, Controle, Eficiência energética.

POWER CONSUMPTION CONTROL AND MONITORING SYSTEM FOR A RESIDENCE

ABSTRACT

This paper proposes a system to monitor and control the power consumption of two single-phase loads, commonly used in homes, regarding lighting and environmental conditioning. The monitoring take effect in the reading of the equipment's voltage and current through related sensors connected to them, while turning on electromechanical switches results in the control of the loads. The project handles concepts of automated control, electrical installations and concepts of the growing area of energy efficiency to address issues related to power control and monitoring. The practical segment responsible for measuring consumption values employs the *Arduino* platform to perform the reading, through specific library. The values are presented in a graphical interface implemented using the *Raspberry* platform, where consumption graphics are also plotted. Control signals responsible for the manual and automatic activation of the loads, as well as reading the values, are sent over a serial interface with a USB cable, used to connect monitoring devices to the main system.

Key words: Power consumption, Monitoring, Control, Energy Efficient.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

A energia elétrica é um bem social indispensável no dia-a-dia para a maior parte da população do planeta. Está presente nas tarefas rotineiras, nas horas de lazer, no trabalho e na própria manutenção do bem-estar social. Dados do *EIA (U.S. Energy Information Administration)* revelam que o mundo consome aproximadamente 19.710 bilhões de *quilowatts*/ hora (EIA, 2012). Cerca de 83,2% da população global já tem acesso à energia elétrica (THE WORLD BANK, 2010).

Além de um bem básico para a população, a energia elétrica é um fator determinante no desenvolvimento da sociedade (KONDO, 2005), podendo ser aplicada na produção de alimentos, moradia, comércio, indústria, agricultura, saúde, transporte, entre outros. Para manter a economia próspera, um país precisa de uma matriz energética que suporte essa intensidade de energia.

O Brasil possui no total 4.210 empreendimentos em operação, totalizando 138.155.616 kW de potência instalada (ANEEL, 2015). Destes empreendimentos 18,11% são fontes não-renováveis de combustíveis fósseis, 8,80% são fontes de biomassa, 61,71% são de fontes hídricas e os outros 11,36% restantes compreendem as usinas eólicas, nucleares, solares e energia advinda de importação (ANEEL, 2015). Com 1.181 usinas e centrais geradoras hidrelétricas espalhadas pelo país, gerando 90.300.259 kW da capacidade instalada, com mais 47 em construção e 176 em planejamento (ANEEL, 2015), o Brasil é um investidor pesado na geração de energia hidráulica.

Um país deve sempre investir no recurso que possui em abundância. O Brasil possui a maior produção hídrica do mundo com uma geração média de 5.745 km^3/ano (LIMA, 2001). Com uma situação hídrica favorável, o investimento em uma matriz hidráulica robusta é o passo mais lógico a se fazer.

O Brasil se beneficia com a construção de usinas hidrelétricas, entretanto, muitos são os impactos ambientais envolvidos na sua construção, como mudanças no clima, erosão marginal, alteração nas características do leito, perda de biodiversidade, elevação de matéria orgânica na água, dentre outros *(AYA e EDGAR, 2005 apud LEITE, 2005)*. Com a chegada do período das secas, a capacidade de geração de energia das usinas é reduzida. O déficit na geração de hidroeletricidade de 2015 em comparação com o do ano de 2014 chegou a 8,31% em janeiro, 9,15% em fevereiro e 2,76% em março (ONS, 2015). Além de deixar cidades sem água para o uso doméstico, também houve um aumento médio de 33% (ELETROPAULO, 2015) na tarifa

de energia residencial, trazendo o sistema de bandeiras tarifarias para custear o maior acionamento das usinas termelétricas.

O impacto na geração de energia nuclear pelo evento em Fukushima limitou o crescimento desta na matriz energética (US GAO, 2015). As fontes renováveis estão sendo foco de intenso estudo, em especial a energia eólica. Em todo o mundo, o investimento em energia renovável aumentou aproximadamente 17% em 2014 (UNEP, 2015).

A previsão é de que cresça no mundo o consumo de combustíveis fósseis. Até 2040, a demanda de energia pode aumentar em 35%, com China e Índia liderando o cenário de consumo energético. Com 2 bilhões de pessoas a mais no planeta a demanda mundial de eletricidade pode aumentar em 90% (EXXONMOBIL, 2014).

Suprir a demanda mundial no futuro será um desafio. Com essa perspectiva em mente, além dos esforços no desenvolvimento das tecnologias de geração de energia renovável, também se tem um incentivo crescente no aperfeiçoamento de conceitos, metodologias e tecnologias na área de eficiência energética. Esta atividade tem como definição usar o conhecimento de forma aplicada, empregando os conceitos da engenharia, da economia e da administração aos sistemas energéticos, melhorando o seu desempenho e reduzindo suas perdas (HENRIQUES, DA COSTA, *et al.*, 2007).

No Brasil, o Programa Nacional de Conservação de Energia (PROCEL) lidera o esforço para promover o uso eficiente de energia, assim como a Empresa de Pesquisa Energética (EPE) e a Agência Nacional de Energia Elétrica (ANEEL) também contribuem para o estudo e pesquisa do setor elétrico, todos eles são iniciativas do governo. A lei 10.295/2001 de eficiência energética estabelece níveis máximos de consumo e mínimos de eficiência energética, regulamentando a mesma no país. O Instituto Nacional de Eficiência Energética (INEE) também tem um papel importante na promoção da eficiência energética no Brasil, assim como várias outras ONGs, associações, planos, congressos, universidades e iniciativas dos setores público e privado.

Referente a energia elétrica, são muitas as aplicações práticas de gestão e otimização do uso de energia nas áreas residencial, industrial, transportes, agricultura, saúde e etc. As contribuições dessas implementações são importantes para a segurança energética, modicidade tarifária, competitividade da economia e redução de impactos ambientais, entre eles as emissões de gases de efeito estufa (EPE, 2014).

Na construção civil, o correto dimensionamento da distribuição de energia elétrica pelos pontos de tomada e iluminação da estrutura é atingido pelo desenvolvimento do projeto de instalação elétrica. De acordo com Ademaro Cotrim, uma instalação elétrica é “o sistema

elétrico físico, ou seja, é o conjunto de componentes elétricos associados e coordenados entre si, composto para um fim específico. ” (2008). Neste caso, o fim específico é o funcionamento dos equipamentos conectados à rede elétrica.

O controle automático nasceu nas indústrias, com o primeiro trabalho significativo na área criado no século XVIII (OGATA, 2003). O controlador centrífugo para controle de velocidade de uma máquina a vapor de *James Watt* trazia à tona questões de controle automático que são base para os sistemas de controle atuais. Os resultados positivos na indústria trouxeram os conceitos de controle automatizado às residências, com promessas de integração entre sistemas distintos, maior praticidade e redução de perdas e desperdício de energia. O controle automático das residências de forma eficiente é objetivo da automação residencial.

Com a aplicação da automação residencial nas instalações elétricas, tarefas e comandos que antes eram realizados pelo ser humano agora são executados por sistemas inteligentes. As perdas e erros intrínsecos à imprecisão humana são corrigidos pela precisão da máquina, limitada apenas pelo atual desenvolvimento da tecnologia. Um sistema de automação residencial é um ambiente ideal para a aplicação dos conceitos de eficiência energética. A área residencial, assim como a industrial, detém juntas mais da metade do potencial de eficiência energética do país (EPE, 2010).

Com a integração disponibilizada por tecnologias também usadas na automação residencial, já existem projetos de redes inteligentes responsáveis por distribuir a energia de forma segura e com tolerância às possíveis falhas que podem ocorrer, promovendo uma medição justa do consumo para os consumidores. Estas redes são chamadas de Smart Grids. Seu principal benefício é dar ao usuário a capacidade de gerenciar e tomar decisões relacionadas ao seu consumo de energia, que hoje é feito pelas empresas de geração de energia. A redução dos gases de efeito estufa também são consequências das redes inteligentes. Este conceito de *Smart Grid* tenta mudar um sistema que permanece praticamente inalterado desde o século 19, com estações geradoras e um sistema de entrega de energia eletromecânico operado por centrais de controle (O. SIDDIQUI, 2008).

Para se obter um melhor controle do consumo de energia elétrica de uma residência, é necessário ter o conhecimento dos níveis atingidos de consumo em um período específico. A medição do consumo de energia nas *Smart Grids* é feita através de medidores eletrônicos chamados *Smart Meters*. Estes aparelhos possuem uma conexão com a concessionária e trocam dados de cobrança e monitoramento, em tempo real, periodicamente (FERREIRA, 2012). São a evolução dos medidores eletromecânicos atualmente instalados na maior parte do Brasil.

Infelizmente, a implantação deste sistema no país representa um grande desafio, pois os custos envolvidos para substituição de todos os equipamentos são elevados.

Partindo do cenário descrito, a proposta do projeto se baseia no desenvolvimento de um sistema de monitoramento e controle dos níveis de consumo de energia de duas cargas monofásicas de uma residência. Como a rede elétrica das residências possui sinais de tensão e corrente sinusoidais, ou seja, funcionam em corrente alternada, para a obtenção dos valores da potência ativa é necessário converter os sinais de tensão e corrente em níveis manipuláveis pelo circuito. A redução e adequação do sinal de corrente é realizada por um sensor de corrente específico e o sinal de tensão é tratado por um sensor de tensão descrito ao longo do projeto. A plataforma *Arduino* será responsável por receber os valores dos sensores de tensão e corrente, realizar os cálculos necessários para cada leitor de consumo e enviar os mesmos pela comunicação serial até o sistema principal. A apresentação dos valores lidos pelo leitor de consumo, em intervalos de tempo fixos, será realizada em uma interface gráfica e gráficos de consumo de fácil compreensão ao consumidor, localizados na plataforma *Raspberry*.

Além de adquirir os níveis de consumo da residência, sensores de presença e temperatura, localizados em dois cômodos da residência, enviarão seus dados para o sistema principal. Para cada condição previamente definida, sinais de controle serão enviados aos interruptores eletromecânicos, acionando ou desligando as cargas ligadas aos leitores de consumo. Os equipamentos e cômodos monitorados, ao atingirem níveis superiores de consumo, vão gerar notificações ao usuário. Pesquisas no consumo residencial mostram que os equipamentos: ar condicionado, refrigerador, congelador, lâmpadas, chuveiro elétrico, máquina de lavar roupas e televisão representam 85% do consumo de uma residência brasileira típica ou média (EPE, 2010). A automação e controle destes equipamentos, com a diminuição do desperdício de energia e conscientização do consumidor, contribuem para a manutenção do meio ambiente e para uma redução na cobrança de energia elétrica da residência.

1.1. Objetivos do Trabalho

O objetivo geral deste trabalho é desenvolver um sistema de monitoramento e controle do consumo de energia de duas cargas monofásicas de uma residência. Os níveis de consumo serão traduzidos em gráficos de fácil compreensão e o controle de cômodos e equipamentos será feito por atuadores individuais e por conscientização do usuário.

Objetivos específicos:

- Criar o protótipo do *hardware* do leitor de consumo de energia usando a plataforma *Arduino*, o sensor de corrente invasivo e os componentes eletrônicos diversos;
- Desenvolver o código, em linguagem C, do Arduíno para o leitor de consumo;
- Adaptar o leitor de consumo para equipamentos e cômodos diferentes;
- Criar o protótipo do *hardware* do sistema principal usando *Raspberry*;
- Integrar sensores de presença dos cômodos, sensor de temperatura e atuadores ao sistema;
- Desenvolver o código, em linguagem *Python*, do *Raspberry* para o sistema;
- Desenvolver a integração entre os leitores de consumo e o sistema principal;
- Traduzir os níveis de consumo recebidos pelo *Raspberry* em gráficos;
- Desenvolver uma interface amigável e de fácil utilização;
- Realizar testes de condicionamento após o desenvolvimento dos leitores de consumo e do sistema principal;
- Projetar placa de circuito impresso para cada leitor de consumo;
- Projetar maquete didática para apresentação na banca;

1.2. Metodologia

O projeto é dividido em leitores de consumo de energia e sistema principal. Os leitores de consumo e suas variações possuirão sensores de corrente e tensão responsáveis por recolher os dados utilizando a plataforma *Arduino*. Os dados serão enviados pelo *Arduino* até o sistema principal através de um cabo USB, interface serial própria do *Arduino*. Os dados recebidos pelo *Raspberry* serão traduzidos em gráficos e os sensores de presença e temperatura, interpretados em respostas para os atuadores de cada equipamento e cômodo. A *Figura 1* ilustra o modelo do projeto de forma simples, com o intuito de facilitar seu entendimento.

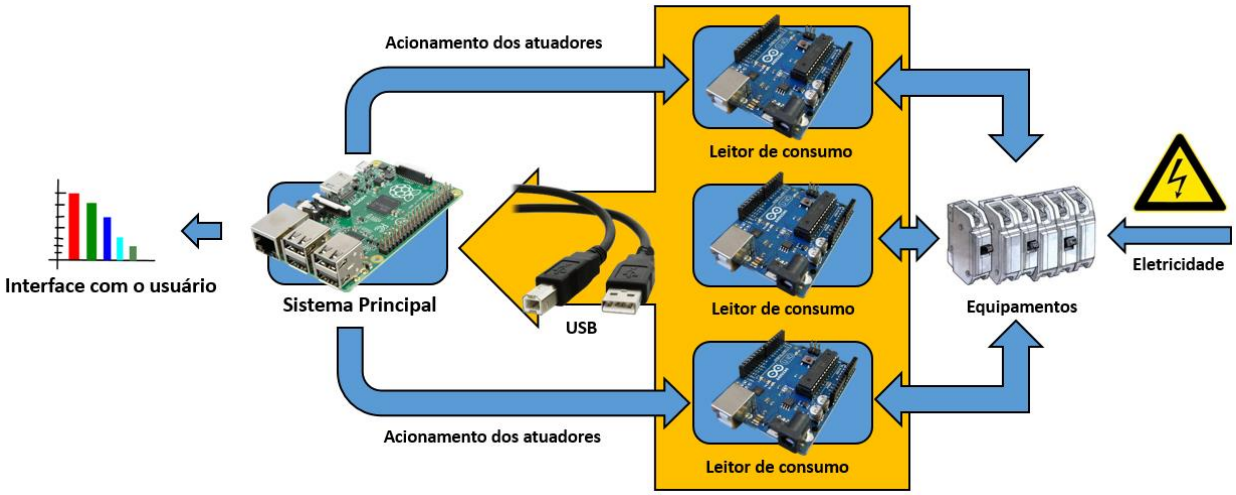

Figura 1 – Visão geral do projeto

Fonte: Elaborada pelo autor

Inicialmente será feita uma pesquisa sobre os componentes eletrônicos necessários e suas funções, tais como: sensores de tensão e corrente, *Arduino*, *Raspberry*, componentes eletrônicos diversos, conector USB e ainda as formas de medição de tensão e corrente elétrica. Após um estudo sobre os elementos envolvidos no consumo de energia, iniciar-se-á o desenvolvimento do *hardware* e código do leitor de consumo de energia e em seguida do *hardware* e código do sistema principal, divididos nas seguintes etapas:

- Etapa 1 – Leitor de consumo de energia: responsável por converter os níveis de tensão e corrente da rede elétrica em valores para a plataforma *Arduino*. Um sensor de corrente ACS712 estará localizado na fase da rede elétrica, assim como um circuito sensor de tensão localizado na fase e no neutro;
- Etapa 2 – Testes de veracidade: Realizar testes de condicionamento no leitor de consumo de energia e verificar se os valores de tensão e corrente condizem com os da rede. Os testes utilizarão osciloscópio digital, multímetro digital e analisador de consumo;
- Etapa 3 – Sistema principal: responsável por receber os valores de potência ativa, reativa aparente e fator de potência do leitor de consumo, além de receber os dados do sensor de presença HC-SR501, sensor de temperatura LM35 e controlar o acionamento dos atuadores (como LEDs, *buzzers* e relés);
- Etapa 4 – Comunicação: Desenvolver a integração entre os leitores e o sistema principal utilizando a interface USB do *Raspberry* e do *Arduino*;
- Etapa 5 – Testes de veracidade: Realizar testes de condicionamento no sistema

principal, verificando se os dados recebidos dos leitores estão corretos;

- Etapa 6 - Acionamento de motores: Sincronizar os dados dos sensores de presença, sensor de temperatura com os comandos do sistema principal para acionamento dos relés de cada atuador;
- Etapa 7 – Testes de acionamento: Realizar testes de condicionamento no sistema principal, verificando se cada comando para acionamento dos atuadores executa de forma correta sua função;
- Etapa 8 – Representação de gráficos: traduzir os níveis de consumo em relatórios e gráficos no monitor do usuário.
- Etapa 9 – Maquete didática: Desenvolver uma representação, em escala adequada, de uma residência, com os leitores e sistema intergrados e em funcionamento.

1.3. Motivação

A atual situação energética do Brasil mostra que o sistema não é perfeito. Em épocas de seca, o impacto na geração de energia é amplo. Com o agravamento dos efeitos do aquecimento global e com o aumento das tarifas de eletricidade, as oportunidades para um maior comprometimento da população com a gestão de energia estão cada vez mais frequentes.

O avanço nos estudos de eficiência energética e no desenvolvimento da automação residencial fornece ao consumidor o poder de controlar o seu próprio consumo. Os benefícios de medidores inteligentes e sistemas de gestão de consumo compreendem desde a evidente economia nos gastos, mas também o aumento da eficiência do processo pela diminuição das perdas e desperdícios, além de despertar a consciência racional do usuário.

1.4. Resultados esperados

Uma representação de dois cômodos será confeccionada para a apresentação, além dos leitores de consumo em placas de circuito impresso. Uma lâmpada incandescente será utilizada representando a iluminação da residência assim como um mini ventilador será utilizado representando o condicionamento ambiental.

Anseia-se, também, que os níveis de consumo monitorados se aproximem dos valores reais, ou seja, que o erro seja mínimo. Com a precisão disponibilizada pelos componentes

envolvidos, estima-se um erro de aproximadamente 6%.

O projeto procura incentivar o uso consciente de energia, instigando o usuário a contribuir cada vez mais com a preservação do meio ambiente.

1.5. Estrutura do trabalho

O desenvolvimento do trabalho é dividido em capítulos, estes elementos textuais serão descritos a seguir: O Capítulo 1 é composto pela introdução aos assuntos tratados, objetivos gerais e específicos envolvidos, metodologia utilizada para realizar o projeto, motivação do mesmo e resultados esperados. O capítulo 2 expõe os conceitos e fundamentação teórica utilizados para a execução do trabalho. O capítulo 3 exibe a saída para o problema apresentado, descrevendo o modelo utilizado. O capítulo 4 analisa os resultados obtidos pela aplicação da proposta. E por último o capítulo 6 realiza as conclusões finais e sugestões para possíveis trabalhos futuros.

CAPÍTULO 2 - REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos teóricos abordados para o desenvolvimento do projeto proposto, além de uma descrição dos materiais utilizados e suas justificativas de uso.

2.1. Análise de energia elétrica

Um circuito elétrico é uma interconexão de elementos elétricos (CHARLES e MATTHEW, 2013). Estes elementos podem ser passivos (não geram energia), como resistores, capacitores e indutores, ou ativos (amplificam energia), como baterias, geradores e amplificadores operacionais. Estes circuitos são desenvolvidos com um objetivo específico e variam muito em função e complexidade.

O fluxo de carga por unidade de tempo, corrente elétrica que percorre um circuito, é medido em *ampères* (A) e a energia necessária para deslocar uma carga unitária através de um elemento, diferença de potencial, é medida em *volts* (V). Quando uma corrente elétrica percorre um elemento através de uma diferença de potencial, este consome ou fornece energia por unidade de tempo. Essa energia fornecida ou absorvida por um elemento é chamada de potência (*p*) e sua medição é dada em *watts* (W). A equação que descreve a potência instantânea, variável com o tempo, em um elemento é dada por:

$$\boldsymbol{p = v\,x\,i} \qquad \textit{(1)}$$

Pela convenção de sinal passivo, a corrente que entra no elemento pela polaridade positiva da tensão implica na absorção de potência pelo mesmo. Da mesma forma, a corrente que entra no elemento pela polaridade negativa da tensão implica no fornecimento de potência pelo mesmo (CHARLES e MATTHEW, 2013).

A capacidade de um elemento de realizar trabalho é chamada de energia (*w*), medida em *Joules*. Em circuitos baseados em transformação de energia, a corrente que flui por um elemento que possui uma resistência o aquece, dissipando energia térmica. Este princípio se chama efeito *Joule* e é usado em vários eletrodomésticos de uma residência, como um ferro de passar e um forno elétrico. A capacidade de resistir à corrente elétrica, resistência de um elemento, pode ser representada pela lei de *Ohm*, apresentada a seguir, isolando-se a mesma (*r*):

$$\boldsymbol{v = i\ x\ r} \qquad \textbf{(2)}$$

Parte da energia em um circuito realiza trabalho e a outra parte é dissipada em energia térmica. As concessionárias de energia elétrica medem a energia em *watts*-hora, correspondente a 3.600 *Joules*.

A energia por unidade de tempo, equação que define a equação da potência instantânea, é a substituição dos termos de corrente e tensão pelos seus correspondentes diferenciais. A corrente representa a derivada da carga (*q*) em função do tempo (*t*). A tensão representa a derivada da energia (*w*) em função da carga (*q*). A potência, como dito acima, representa a derivada da energia (*w*) em função do tempo (*t*). A interação entre as três está representada a seguir (NILSSON e RIEDEL, 2009):

$$\boldsymbol{p = \frac{dw}{dt} = \left(\frac{dw}{dq}\right) x \left(\frac{dq}{dt}\right)} \qquad \textbf{(3)}$$

Para a corrente elétrica, se suas cargas permanecem constantes com o tempo ela é chamada de corrente continua (CC) e se suas cargas variam com o tempo segundo uma forma senoidal, ela é chamada de corrente alternada (CA) (CHARLES e MATTHEW, 2013).

2.1.1. Análise de energia elétrica em corrente alternada

Para o correto funcionamento dos eletrodomésticos de uma casa, é necessário que estes recebam energia da rede elétrica da residência. Estas unidades funcionais que desempenham alguma função elétrica nas casas são chamadas de equipamentos elétricos. Estes, em especial, responsáveis por converter energia elétrica em outra forma de energia diretamente utilizável são chamados de equipamentos de utilização, como os eletrodomésticos, equipamentos industriais e aparelhos de iluminação (ADEMARO A. M. B. COTRIM, 2008).

A energia elétrica que alimenta os equipamentos de utilização é gerada nas usinas em corrente alternada, desde a adoção deste padrão desenvolvido por Nikola Tesla com o apoio de George Westinghouse (CHARLES e MATTHEW, 2013). O benefício principal do método de geração, transmissão e utilização em corrente alternada, em comparação com o método em corrente continua, é a transformação da tensão. De acordo com a Lei de Joule, a corrente que percorre um condutor com uma resistência, em um determinado tempo, gera energia térmica. Similarmente, ao transmitir a energia elétrica pelos condutores desde a usina geradora até uma residência, parte da energia transmitida é dissipada em forma de calor. Usando a *Equação 1* como referencial: Para manter-se a potência constante, a tensão é elevada, ocasionando a

redução da corrente que percorre o condutor e, consequentemente, na diminuição da perda de potência em forma de energia térmica. Similarmente, ao transmitir a energia elétrica pelos condutores desde a usina geradora até uma residência, parte da energia transmitida é dissipada em forma de calor. As altas tensões são reduzidas em outras centrais transformadoras e distribuídas para as residências.

Os sinais senoidais são gerados por fontes de tensão e fontes de corrente senoidais. Estas, produzem uma tensão e corrente que variam senoidalmente com o tempo (NILSSON e RIEDEL, 2009). Usinas hidrelétricas, por exemplo, transformam a energia mecânica das turbinas em energia elétrica através de dínamos. Estes dínamos são geradores síncronos e giram na mesma frequência que a rede elétrica conectada à usina.

Os sinais de tensão e corrente podem ser representados usando a função seno ou cosseno, como apresentado a seguir:

$$\boldsymbol{v = V_m\, x\, cos(\omega t + \varphi_v)} \qquad \textit{(4)}$$

$$\boldsymbol{i = I_m\, x\, cos(\omega t + \varphi_i)} \qquad \textit{(5)}$$

O sinal senoidal é periódico, ou seja, se repete em intervalos regulares. O intervalo de tempo de um período é representado por T e medido em segundos. O número de ciclos por segundo (frequência) é apresentado abaixo e medido em Hertz (Hz):

$$\boldsymbol{f = \frac{1}{T}} \qquad \textit{(6)}$$

O coeficiente *t* nas equações *4* e *5* representam o valor numérico de *T* ou *f*. O coeficiente ω representa a velocidade de rotação, frequência angular, medida em radianos. Os coeficientes V_m *e* I_m são respectivamente a amplitude máxima da tensão senoidal e corrente senoidal. O ângulo φ, conhecido como ângulo de fase, determina o ponto onde o tempo é igual a zero.

Um sinal senoidal ainda possui um valor eficaz ou *rms* (*root mean square*) dependente apenas da amplitude máxima da onda. Esta medida representa a equivalência de dissipação de energia em um resistor da fonte CA e uma fonte CC. O valor de tensão *rms* da *Equação 4* é apresentado abaixo (NILSSON e RIEDEL, 2009):

$$\boldsymbol{V_{rms} = \frac{V_m}{\sqrt{2}}} \qquad \textit{(7)}$$

Uma função senoidal também pode ser representada como um fasor, introduzido por Charles Proteus Steinmetz (1865-1923). Este método representa um número complexo que contém as informações de amplitude e ângulo de fase destas funções (NILSSON e RIEDEL, 2009). A transformação fasorial utiliza as notações retangular, polar e exponencial de uma

representação complexa. Um número complexo z pode ser representado nessas três formas a seguir:

$$z = x + jy \quad \textit{Forma retangular} \quad \textbf{(8)}$$

$$z = r\angle\varphi \quad \textit{Forma polar} \quad \textbf{(9)}$$

$$z = re^{j\varphi} \quad \textit{Forma de Euler} \quad \textbf{(10)}$$

Utilizando a identidade de *Euler* apresentada abaixo, é possível que os coeficientes de um sinal senoidal periódico possam ser representados usando a notação complexa, simplificando a análise do regime permanente de circuitos AC.

$$e^{\pm j\varphi} = \cos\varphi \pm j\sin\varphi \quad \textbf{(11)}$$

Assim como na representação retangular, a função cosseno representa a parte real da função exponencial e a função seno representa a parte imaginária (NILSSON e RIEDEL, 2009). Utilizando o sinal de tensão da Equação 4, no domínio do tempo, e a representação de Euler, obtém-se a transformada fasorial na forma polar da função senoidal, no domínio da frequência, apresentada abaixo:

$$V = V_m e^{j\varphi} \quad \textbf{(12)}$$

Tomando como base os sinais senoidais de tensão e corrente das equações *4* e *5*, ambos substituídos na *Equação 1* da potência, aplicando-se algumas relações trigonométricas, obtém-se a potência instantânea numa carga monofásica:

$$p = \left[\frac{1}{2} V_m I_m \cos(\varphi_v - \varphi_i)\right] + \left[\frac{1}{2} V_m I_m \cos(2\omega t + \varphi_v + \varphi_i)\right] \quad \textbf{(13)}$$

O primeiro termo em colchetes não depende do tempo e sim da diferença de fase entre tensão e corrente. O segundo é uma função senoidal de frequência 2ω, dobro da tensão ou corrente (CHARLES e MATTHEW, 2013).

Como a potência instantânea varia com o tempo, wattímetros (instrumentos de medição de potência) utilizam a potência média para realizar a aferição, dada por:

$$P = \frac{1}{T}\int_0^T p(t)\,dt \quad \textbf{(14)}$$

Aplicando a *Equação 13* na equação de potência média (*Equação 14*), obtém-se:

$$P = \frac{1}{2} V_m I_m \cos(\varphi_v - \varphi_i) \quad \textbf{(15)}$$

Em circuitos puramente resistivos a tensão e a corrente estão em fase, $\varphi_v = \varphi_i = 0$. Estes circuitos sempre absorvem potência. Em circuitos puramente reativos, $\varphi_v - \varphi_i = \pm 90°$, não há absorção de potência (CHARLES e MATTHEW, 2013).

Considerando V_m *e* I_m na *Equação 15,* com seus valores eficazes RMS V_{rms} *e* I_{rms}, o produto de ambos é conhecido como potência aparente (*S*), medida em volt-ampères (VA). A razão entre a potência média e a aparente é chamado de fator de potência, apresentado abaixo:

$$\boldsymbol{FP = \frac{P}{S} = \cos(\varphi_v - \varphi_i)} \qquad \textit{(16)}$$

A impedância equivalente de uma carga elétrica pode ser representada abaixo, sendo *R* a resistência e *X* a reatância da carga:

$$\boldsymbol{\dot{Z} = R + jX} \qquad \textit{(17)}$$

A potência complexa, utilizada para expressar o efeito total das cargas em paralelo, pode ser expressa em termos de impedância local Z, apresentada abaixo:

$$\boldsymbol{S = I_{rms}^2 Z = I_{rms}^2 (R + jX) = P + jQ} \qquad \textit{(18)}$$

Como a tensão fasorial nos terminais de um resistor é simplesmente a resistência vezes a corrente fasorial que o percorre, o elemento ativo (resistência), numa carga elétrica, consome a potência média liberada (ativa) e o elemento reativo (reatância) troca energia com a fonte (ADEMARO A. M. B. COTRIM, 2008). A parte real *P* depende da resistência da carga e representa a potência ativa. O sinal nunca se torna negativo, sempre fornecendo potência instantânea à carga. A parte imaginária *Q* depende da reatância e é denominada potência reativa. Esta apresenta valor médio nulo, trocando potência instantânea com a fonte, o que não é desejável. Ambas são apresentadas abaixo:

$$\boldsymbol{P = V_m I_m \cos(\varphi_v - \varphi_i)} \qquad \textit{(19)}$$

$$\boldsymbol{Q = V_m I_m \sin(\varphi_v - \varphi_i)} \qquad \textit{(20)}$$

De modo geral, as cargas elétricas em um circuito em CA consomem potência aparente, medida em *VA*, composta pela parcela de potência ativa, medida em *W*, e reativa, medida em *Volt-Ampére* reativo *VAr* (ADEMARO A. M. B. COTRIM, 2008). Uma carga indutiva absorve Q positiva, ou seja, um indutor absorve potência reativa. Uma carga capacitiva absorve Q negativa, ou seja, gera potência reativa e cargas resistivas consomem potência ativa.

O uso de certos equipamentos e dispositivos junto a rede elétrica causam variações nos

sinais de corrente e tensão das mesmas. Estes dispositivos não-lineares causam distorções na rede chamadas de harmônicas (DECKMANN e POMILIO, 2010). As relações entre potências e fator de potência, apresentadas nas *Equações 19 e 20*, são válidas para cargas com sinais lineares, sinais senoidais puros. Porém, cargas não-lineares não possuem esta relação entre ângulos de fase, tendo o efeito das harmônicas grande influência nos sinais de corrente e tensão. Para o carregamento não-linear, a única relação que permanece para o fator de potência é a razão entre potência ativa e aparente, não dependendo do ângulo de fase entre sinais.

2.1.2. Medidores de energia elétrica e tarifação

Em sistemas elétricos, *demanda* se refere a potência (kW) e *consumo*, ao requerimento energético (kWh). O consumo energético de uma residência é cobrado através da fatura elétrica. Nela, a concessionária de energia elétrica apresenta a quantia total que deve ser paga pelo consumidor pela prestação do serviço público de energia elétrica referente a um período especificado. O conjunto de normas e regulamentos que estabelecem o valor monetário da eletricidade para todas as classes consumidoras é chamado de sistema tarifário de energia elétrica, regulamentado pela ANEEL (Agência Nacional de Energia Elétrica) (SANTOS, SIMÕES, *et al.*, 2006).

Nas Condições Gerais de Fornecimento de Energia Elétrica, Resolução ANEEL n° 456, as unidades consumidoras são divididas em grupos e subgrupos, por níveis de tensão de fornecimento. Cada um destes grupos possui um valor definido de tarifa, apresentada na *Tabela 1*. Sistemas elétricos de potência entre 69 a 500 kV são considerados de Alta Tensão, entre 13,8 a 138 kV de Média Tensão e abaixo de 13,8 kV de Baixa Tensão.

Tabela 1 – Grupos de Fornecimento

Fonte: Adaptado de (SANTOS, SIMÕES, et al., 2006)

Grupo A		**Grupo B**	
Subgrupo	**Fornecimento**	**Subgrupo**	**Fornecimento**
A1	≥ 230 kV	B1 - Residencial	> 2,3 kV
A2	88 kV a 138 kV	B1 – Residencial baixa renda	
A3	69 kV	B2 – Rural	
A3a	30 kV a 44 kV	B2 – Cooperativa de eletrificação rural	
A4	2,3 kV a 25 kV	B2 – Serviço público de irrigação	
AS	Subterrâneo	B3 – Demais classes	
		B4 – Iluminação pública	

Com relação a modalidade tarifária, a hora do consumo de energia elétrica pode ser classificada em Convencional, independe das horas de utilização com demanda inferior a 300 kW, ou Horo-Sazonal, tarifas diferenciadas de acordo com as horas de utilização do Grupo A da *Tabela 1*. Esta última ainda pode ser classificada em: Tarifa azul, Horário de ponta, Horário fora de ponta, Período úmido e Período seco (VIANA, BORTONI, *et al.*, 2012).

O faturamento, nota fiscal com a quantia total a ser paga pela prestação de serviço público de energia, se refere a um período específico de monitoramento. Após serem aplicadas as tarifas de fornecimento, o valor líquido é representado em moeda corrente. Para unidades pertencentes ao grupo B, o faturamento é baseado no consumo de energia elétrica ativa e, quando aplicável, no consumo de energia elétrica reativa excedente. A *Equação 21* apresenta o cálculo de consumo de energia. Uma taxa de disponibilidade e manutenção do sistema elétrico ainda é aplicada, representando o valor mínimo faturável. Para sistemas monofásicos e bifásicos a dois condutores, esta taxa equivale a 30 kWh em moeda corrente. Para sistemas bifásicos a três condutores, a taxa equivale a 50 kWh. Para sistemas trifásicos, o equivalente é 100 kWh (VIANA, BORTONI, *et al.*, 2012).

$$\boldsymbol{Consumo} = \left(\frac{\boldsymbol{Potência\ em\ Watt}}{\boldsymbol{1000}}\right) \boldsymbol{x}\ (\boldsymbol{Tempo\ em\ horas}) = \boldsymbol{Total\ em\ KWh} \qquad \textit{(21)}$$

A leitura do consumo de energia elétrica, segundo a resolução ANEEL n° 414, deve ser realizada por equipamentos de medição, instalados pela distribuidora, em intervalos de 27 a 33

dias. São dispositivos eletromecânicos/ eletrônicos capazes de medir o consumo de energia elétrica através de indução eletromagnética. As *Figura 2* e *Figura* 3 ilustram dois tipos destes dispositivos. O primeiro representa um medidor de ponteiros e o segundo, um medidor ciclométrico. Ambas as leituras são feitas através dos números indicados em cada relógio.

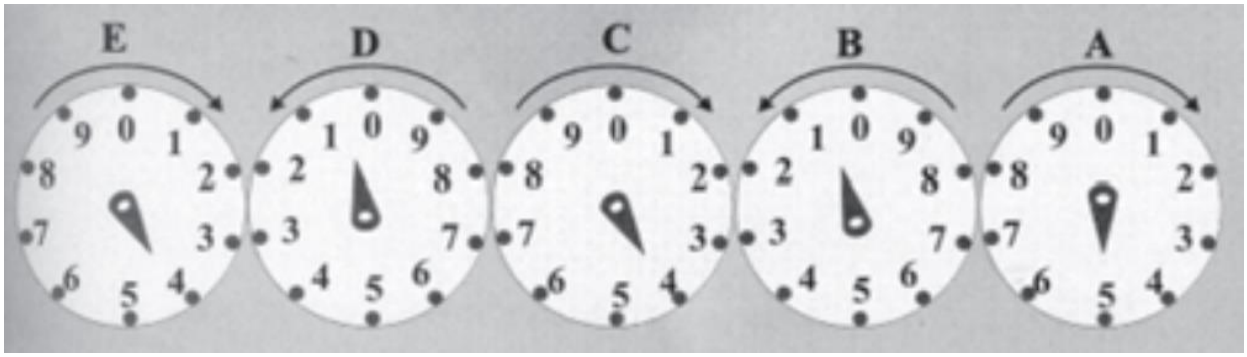

Figura 2 – Medidor de ponteiro

Fonte: PROCEL, 2009

Figura 3 – Medidor ciclométrico

Fonte: PROCEL, 2009

Os medidores eletrônicos possuem um circuito interno capaz de medir o consumo de energia e mostrar os dados em um display digital. No projeto em questão o leitor de consumo de energia simula um medidor eletrônico de energia, enviando seus dados de leitura para o sistema principal.

2.2. Automação e controle

A ideia de controlar tudo ao seu redor se desenvolveu durante muitos anos, por várias etapas de desenvolvimento da civilização. São muitos os exemplos de mecanismos capazes de mover objetos e executar ações a partir de determinadas referências, principalmente em grandes centros de desenvolvimento intelectual. Controles de nível por boias, na Grécia (300 a.C) (FACCIN, 2004), são os primeiros indícios da ideia de retroalimentação registrados.

Assim como disse *Katsuhiko Ogata*: "O controle automático representa um papel vital no avanço da engenharia e da ciência" (2003). O controle de processos industriais permite ao sistema uma maior confiabilidade de resultados, maior segurança, maior eficiência e, consequentemente, mais economia no resultado final. Estes métodos de controle, depois de comprovados sua eficiência, foram trazidos às residências pela automação residencial.

Sem controles automáticos, a regulagem de motores e válvulas em indústrias é realizada manualmente, dando margem a erros de manipulação do operador e falta de precisão. Um controlador automático industrial sempre trabalha com alguma grandeza no sistema de controle. Em um forno industrial a gás, a temperatura da chama é o fator principal que rege as ações que serão realizadas dentro do sistema. Neste exemplo, a variável que se deseja controlar é a temperatura, porém, a mesma não é puramente controlável e necessita que um dispositivo altere seu valor que, no caso do forno, é a liberação de combustível. Logo, as variáveis envolvidas no controle de um forno são: a temperatura e o gás, representando, respectivamente, a variável de processo (*PV*) e a variável manipulada (*MV*) no sistema. A temperatura ideal que o fogão deve manter é chamada de valor desejado, ou *Setpoint*. A diferença entre o valor desejado da variável de processo e o valor que é controlado no processo (*SP*) representa o erro. Em um controlador automático industrial de malha fechada, depois que a medição da saída do sistema foi feita, o sinal de erro precisa ser amplificado antes de ser enviado a um atuador para que a ação de controle seja executada. Todo este processo tem como objetivo a redução do desvio entre *Setpoint* e *PV*, buscando o equilíbrio do sistema (SMUTS, 2011).

2.2.1. Controle de malha fechada

Um sistema de controle pode ser de malha aberta ou malha fechada. No primeiro, a saída não tem efeito na ação de controle, ou seja, não existe medição do sinal de saída e nem sua comparação com o erro respectivo. Um exemplo seria uma máquina de lavar roupa, onde a

lavagem é apenas executada, sem a medição da limpeza das roupas.

Em um sistema de malha-fechada a saída possui um efeito direto na ação de controle. A saída, ou uma função dela, é realimentada no sistema e comparada com seu valor anterior, de modo a reduzir o erro e manter a saída em um nível desejado. Um exemplo seria um sistema térmico onde um ser humano controla uma válvula de saída de vapor que aquece a água, que é então analisada por um termômetro. Com a redução da temperatura da água, o usuário abre mais a válvula para que o vapor a esquente novamente.

Esta operação é um controle de malha-fechada manual, onde o valor de temperatura final é analisado pelo ser humano. Este, compara o valor de saída com o ideal e ajusta a válvula de acordo com o erro. O usuário poderia ser substituído por um controlador automático, de forma análoga: os medidores de erros substituiriam os olhos do operador, seu cérebro equivaleria ao controlador automático e seus músculos o atuador (OGATA, 2003).

2.2.2. Controlador automático

Os componentes básicos de um controlador automático industrial são: Controlador, atuador e sensor. Seu diagrama de blocos é mostrado na *Figura 4*. O controlador é responsável por determinar o erro atuante através de alguma referência previamente definida e amplificar esse erro a um nível de potência suficiente para ativar um atuador (motor, válvula). Este é responsável por executar o sinal de controle no objeto físico a ser controlado. Após a execução do sinal de controle, a saída é medida através de um sensor, realimentando o sistema com o valor real e servindo de base para o controlador, reiniciando o ciclo (OGATA, 2003).

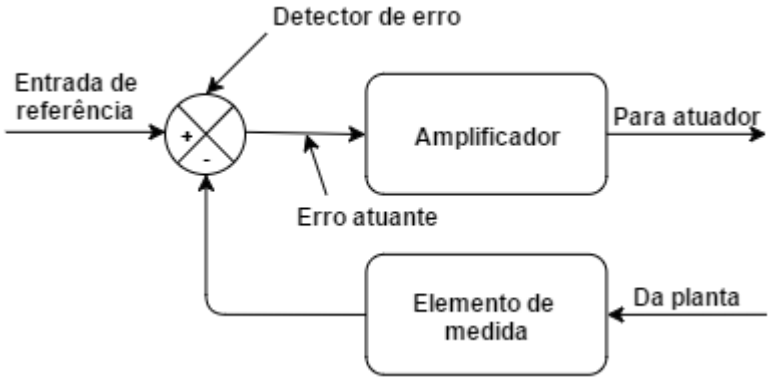

Figura 4 – Diagrama de blocos de um controlador automático industrial

Fonte: Adaptado de (OGATA, 2003)

O princípio fundamental de sistemas de controle é a estabilidade. Se um sistema não é

estável, não há confiabilidade no seu resultado. Logo, ele não desempenhará sua função corretamente. Outro princípio diz que um sistema de controle deve ser capaz de reduzir um erro a valores próximos de zero. A ação de controle está em contato com distúrbios internos e externos e deve ser capaz de tratar estes distúrbios de forma a não prejudicar a saída.

A forma como o controle será realizado define a ação de controle utilizada. As ações de controle mais comuns são: duas posições *on-off*, proporcional, integral proporcional-integral, proporcional-derivativo e proporcional-integral-derivativo. Cada uma com suas vantagens e desvantagens, sendo indicadas para cenários de controle específicos (OGATA, 2003).

2.2.3. Controlador de duas posições *on-off*

Em controladores *on-off*, o elemento atuante possui apenas duas posições fixas, geralmente ligada e desligada. Este tipo de ação de controle é relativamente simples e barato, sendo muito usado em sistemas de controle industriais e domésticos. O intervalo através do qual o sinal de erro atuante deve mover-se antes de ocorrer o chaveamento é denominado intervalo diferencial, ou histerese diferencial. Este pode ser ocasionado por uma lentidão mecânica ou introduzido no sistema de forma intencional para prevenir operações frequentes do mecanismo *on-off* (OGATA, 2003).

Considerando *m(t)* para o sinal de saída do controlador e *e(t)* para o sinal de erro atuante, um controle de duas posições pode ser representado pela *Equação 21* e pela *Figura 5*.

$$\boldsymbol{m(x) = \begin{cases} M_1, & e(t) > 0 \\ M_2, & e(t) < 0 \end{cases}} \qquad \textbf{(22)}$$

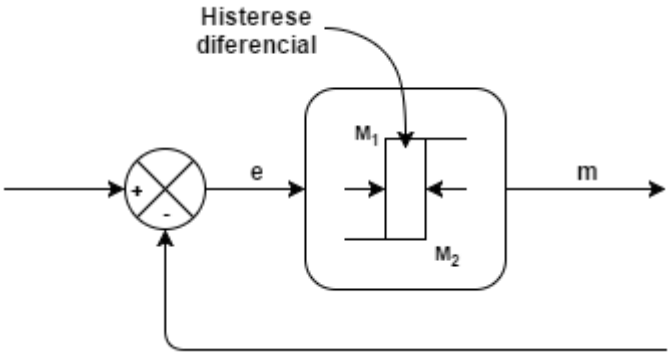

Figura 5 – Diagrama de blocos de um controlador liga-desliga

Fonte: Adaptado de OGATA, 2003

Assim, o projeto será baseado em um controle de malha-fechada *on-off*. O controlador

automático será representado pelo *Raspberry*. Os diversos sensores que compõem o sistema, bem como os leitores de consumo utilizando o *Arduino*, enviarão seus dados ao sistema principal. As informações serão analisadas e comparadas com os níveis ideais configurados. A correção do erro será realizada em forma de sinais de ativação dos atuadores. A variável manipulada (MV) será a tensão de alimentação do circuito, enquanto que as variáveis de processo mudarão com o cômodo e atuador, sendo elas a iluminação (lâmpada) e a temperatura (ventilador).

2.3. Eficiência energética

No setor comercial, assim como no industrial, a importância da energia vem aumentando cada vez mais. Aspectos relacionados à redução de custos advindos do mercado competitivo, incertezas da disponibilidade energética e restrições ambientais corroboram com a ideia de se ter uma melhor administração desta energia. A definição de eficiência energética baseia-se justamente no gerenciamento dos sistemas energéticos aplicando conceitos de engenharia, economia e da administração. Para este gerenciamento ser possível são necessárias as informações dos fluxos de energia do sistema, o acompanhamento dos índices de controle e a atuação dos mesmos tendo em vista a redução do consumo energético através do uso racional de energia (HENRIQUES, DA COSTA, *et al.*, 2007).

O conceito de energia possui várias definições e está presente em diversas áreas do conhecimento humano. Uma destas definições parece abranger a maior parte destas áreas, feita por *James Clerk Maxwell em 1872*: “Energia é aquilo que permite uma mudança na configuração de um sistema, em oposição a uma força que resiste a esta mudança”. Esta definição apresenta duas ideias principais: a transposição de uma resistência para alteração de um estado e o papel da energia nesta superação (VIANA, BORTONI, *et al.*, 2012). Assim, pode-se relacionar esta definição à declaração feita pelo químico *Antoine Lavoisier*: “Na Natureza, nada se perde, nada se cria, tudo se transforma”. Em ambos os conceitos o fluxo energético de um sistema para outro implica na mudança de um estado. Outro conceito importante é o de potência, sendo esta o fluxo de energia em um intervalo de tempo. As diversas formas de energia presentes no mundo podem ser convertidas entre si, reafirmando as leis das conversões energéticas.

A primeira lei da conservação de energia, também conhecida pela primeira lei da termodinâmica, denota que o somatório dos fluxos e estoques de energia em um processo ou sistema é constante em um dado espaço de tempo. A partir desta definição é possível representar

o conceito de eficiência energética, η_{energ} na *Equação 23*, relacionando o efeito energético útil com o consumo energético do sistema e as perdas envolvidas (VIANA, BORTONI, *et al.*, 2012). Parte destas perdas são emitidas em forma de energia térmica e representadas através da lei da dissipação da energia:

$$\boldsymbol{\eta_{energ} = \frac{E_{útil}}{E_{consumida}} = \frac{E_{consumida} - Perdas}{E_{consumida}} = 1 - \frac{Perdas}{E_{consumida}}} \qquad \textbf{(23)}$$

Os recursos energéticos disponíveis na Natureza são classificados como fósseis e renováveis. Ambos são utilizados pelo homem em suas transformações energéticas. Os recursos fósseis, como o petróleo e o gás natural, estão presentes em depósitos naturais, são extraídos e refinados em processos industriais e não são renováveis. As fontes renováveis, como a energia solar e a eólica, são fluxos naturais reabastecidos periodicamente e não devem se extinguir. Estes recursos são classificados como energia primária e são usados para atender a sociedade de diversas formas. Quando a energia primária é transformada, no setor energético, para fins de armazenamento, distribuição e adequação ao uso, como a energia elétrica e derivados do petróleo, ela se torna energia secundária. A energia tratada e disponibilizada ao usuário é chamada de energia útil e é utilizada em equipamentos de uso final, como lâmpadas e motores (SANTOS, SIMÕES, *et al.*, 2006).

No Brasil, muitas são as instituições que lidam com o tema de eficiência energética, assim como existem algumas leis e decretos responsáveis por regulamentar esta prática no país. A lei de eficiência energética (Lei 10.295) foi sancionada em 2001 e estabelece níveis máximos de consumo e mínimos de eficiência energética para equipamentos fabricados e comercializados no país. Assim, equipamentos menos eficientes são retirados do mercado a médio e longo prazo, o desenvolvimento tecnológico de práticas eficientes é incentivado e o gasto final dos consumidores é reduzido (VIANA, BORTONI, *et al.*, 2012). Logo, a garantia de eficiência energética de cada equipamento é garantida por lei através de cada fabricante e distribuidor. Porém, a garantia de eficiência energética, na forma como estes equipamentos serão utilizados, só será atingida através de uma auditoria energética. Este conceito se refere a aplicação de métodos e técnicas para definir objetivos e ações com o intuito de melhorar o desempenho energético e reduzir as perdas de um sistema. A mínima estrutura gerencial de um sistema para promoção da eficiência energética passa pelas etapas apresentadas na *Figura 6*.

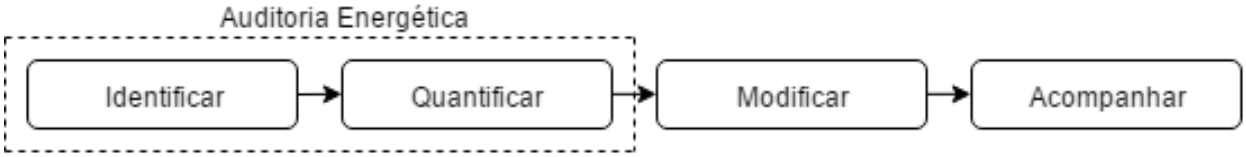

Figura 6 – Etapas de um programa de uso racional de energia

Fonte: Adaptado de (SANTOS, SIMÕES, et al., 2006)

As duas primeiras etapas, correspondentes à auditoria energética, se referem ao diagnóstico da realidade energética de um sistema, identificando o fluxo de energia e os traduzindo em valores mensuráveis. Após definido o comportamento da energia e identificados os pontos de desperdício, a terceira etapa modifica o cenário, usando técnicas variadas, com o objetivo de reduzir estes desperdícios. A última etapa representa a continuidade do processo, o efetivo acompanhamento visa corrigir falhas e imperfeições dos métodos e a análise dos resultados. A execução de forma planejada e organizada desta estrutura constitui um instrumento essencial para o progresso da eficiência energética em um sistema.

Dentre os procedimentos de auditoria de energia, o Diagnostico Energético é o método escolhido para ser utilizado no projeto, sendo adaptado para o uso residencial. Este método visa estudar a unidade consumidora essencialmente levantando o perfil de consumo por uso final. A análise de dados dos pontos críticos indica a necessidade de atuação em equipamentos específicos, porém não trata com detalhes os aspectos econômicos.

Na etapa de modificação são aplicadas as correções dos pontos de desperdício identificadas na auditoria energética. Vários são os métodos de eficiência energética responsáveis por reduzir este desperdício, um deles (relacionado a iluminação) é apresentado a seguir.

2.3.1. Sistemas de iluminação

Em 2012 a iluminação representou 16,2% do total de energia elétrica demandada do setor residencial no Brasil (EPE, 2014). Sendo a luz um recurso fundamental na vida da sociedade, o gerenciamento correto do consumo de energia elétrica de dispositivos emissores de luz se mostra um aspecto essencial no estudo de eficiência energética.

Vários são os dispositivos emissores de luz, cada um possuindo um rendimento e aplicação diferentes. A eficiência luminosa de uma fonte (η) é calculada pelo quociente do fluxo luminoso em lumens (Φ) e a potência consumida em Watts, apresentada na *Equação 24*.

$$\eta = \frac{\Phi}{P} \qquad (24)$$

As lâmpadas incandescentes, muito usadas em residências, emitem luz através de um filamento percorrido por corrente elétrica. Estas possuem uma baixa eficiência luminosa, 11,9 lm/W aproximadamente. As lâmpadas fluorescentes são lâmpadas de descarga de baixa pressão onde a luz é produzida por pós fluorescentes ativados por radiação ultravioleta. As versões compactas das mesmas possuem uma eficiência luminosa superior à das incandescentes, 66 lm/W aproximadamente, também sendo muito usadas em residências (SANTOS, SIMÕES, *et al.*, 2006).

Ao estabelecer as características de iluminação de uma residência em sua concepção, alguns passos são sugeridos para aumentar a eficiência dos focos de luz. Escolha do nível de iluminamento, escolha das lâmpadas e luminárias, cálculo do número de luminárias e sua distribuição são alguns métodos utilizados para o correto dimensionamento do ambiente iluminado (HENRIQUES, DA COSTA, *et al.*, 2007).

Um mecanismo capaz de reduzir a perda de energia com a iluminação de cômodos desocupados é chamado de sensor de presença. Estes usam a radiação infravermelha dos corpos para detectar pessoas e assim acionar ou desligar a iluminação de um ambiente (LAMBERTS, DUTRA e PEREIRA, 2014). O projeto em questão utiliza um sensor de presença, acionando a iluminação somente quando necessário.

2.4. Linguagens de programação

As instruções executadas pela máquina são de baixo nível, compreensíveis por computadores. Esta são definidas pelo design de hardware, geralmente reduzidas a 0s e 1s. No processo de desenvolvimento das tecnologias dos computadores, os programadores começaram a utilizar comandos que executavam sequencias maiores em linguagens de máquina. Estes comandos formaram a base da linguagem *assembly*, utilizada por programas tradutores para converter comandos em linguagem de máquina.

Com a popularização dos computadores, foram desenvolvidas linguagens de alto nível para acelerar o processo de tradução. Programas escritos em linguagem de alto nível por programadores são compreensíveis pelo ser humano, onde instruções únicas poderiam ser escritas para realizar tarefas substanciais. Programas tradutores chamados de compiladores convertem os programas de linguagem de alto nível em linguagem de máquina. Linguagens como o *C*, *C++*, *C# e Java* são exemplos destas linguagens. A compilação deste tipo de

linguagem pode consumir uma quantidade considerável de tempo do computador e para isso programas interpretadores foram desenvolvidos para executá-las diretamente, sem o tempo de compilação (DEITEL e HARVEY, 2010).

2.4.1. Linguagem *C*

A linguagem C é uma linguagem de alto nível estruturada, implementada em 1972 por Denis Ritchie. Se popularizou quando foi usada no desenvolvimento do sistema operacional UNIX. O C é uma das linguagens mais populares de hoje em dia pela sua facilidade de entendimento e vantagem de compilação em várias plataformas, influenciando muitas outras linguagens de programação. A extensão C++ foi desenvolvida por Bjarne Stroustrup no início da década de 1980 com o objetivo de "sofisticar" a linguagem C e a fornecer a capacidade para a programação orientada a objetos (DEITEL e HARVEY, 2010).

A IDE da plataforma *Arduino* utiliza essencialmente a linguagem C/ C++ para o desenvolvimento de seus códigos-fonte.

2.4.2. Linguagem *Python*

Com o advento da *internet*, *softwares* de código aberto, metodologias ágeis de desenvolvimento e as linguagens dinâmicas conquistam cada vez mais espaço na sociedade. O *Python* é uma linguagem dinâmica e poderosa que está se tornando cada vez mais popular (BORGES, 2010). É definida como de altíssimo nível, orientada a objeto, de tipagem dinâmica e forte, interpretada e interativa. Foi criada por Guido Van Rossum em 1991 e atualmente é gerenciada pela organização sem fins lucrativos *Python* Software Foundation (PYTHON, 2015).

A linguagem *Python*, versão 2.7, é utilizada no desenvolvimento dos códigos-fonte na plataforma *Raspberry*.

2.5. Materiais utilizados

Esta seção é reservada para a descrição dos materiais utilizados no projeto, sua função,

integração com outros componentes e justificativa de uso.

2.5.1. Transformador 110/220V 6+6V/ 200mA

A unificação matemática dos campos elétrico e magnético feita por James Clerk Maxwell em 1865 permitiu o desenvolvimento da área de eletromagnetismo na física. O conceito da indutância mútua derivada desses estudos diz que quando dois indutores (bobinas) estiverem bem próximos um do outro, o fluxo magnético provocado pela corrente em uma bobina se associa com a outra bobina induzindo, consequentemente, tensão nessa última (CHARLES e MATTHEW, 2013).

O transformador é um dispositivo magnético que utiliza o princípio do fenômeno da indutância mútua para elevar ou reduzir uma tensão. São utilizados na geração e transmissão de energia em corrente alternada. Este possui uma bobina diretamente ligada a fonte de tensão, geralmente 110V ou 220V, chamada enrolamento primário e uma bobina ligada a carga, chamada de enrolamento secundário. A tensão de saída na carga já é reduzida pelo fator específico de cada transformador. Ainda existem transformadores com uma derivação central no secundário, utilizados com circuitos retificadores para se obter duas tensões de mesma amplitude, defasadas em 180°.

Em transformadores ideais, o número de espiras em cada bobina pode ser relacionado com a tensão em cada enrolamento, como apresentado abaixo, sendo a energia em cada terminal, equivalentes. A *Figura 7* apresenta um transformador ideal, o número de espiras e a tensão entre os terminais sendo representadas, respectivamente, por *N* e *V*.

$$\frac{V_2}{V_1} = \frac{N_2}{N_1} \qquad (25)$$

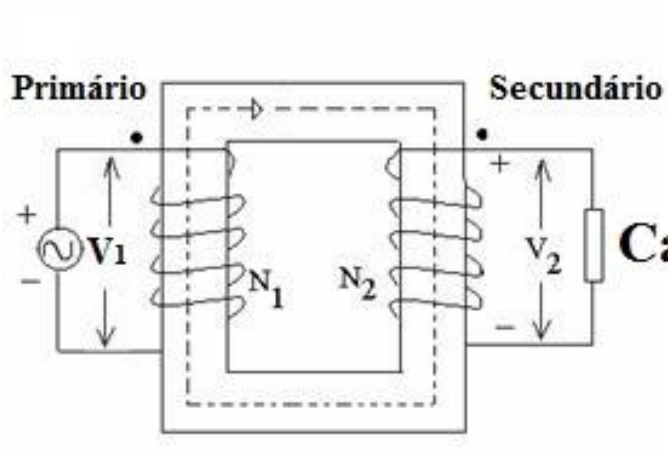

Figura 7 – Transformador ideal

Fonte: Adaptada de iete-elan.ac.in/SolQP/AE10.htm

O projeto utiliza um transformador abaixador de tensão de 110/220V para 12V (terminal +6V utilizado) com quatro terminais no primário e derivação central no secundário. Utilizado no leitor de consumo para reduzir a tensão da rede elétrica o suficiente para, em conjunto com outros componentes eletrônicos, analisar o sinal de tensão pela porta analógica do *Arduino*, que suporta até 5 volts.

2.5.2. Sensor de corrente invasivo

O sensor de corrente utilizado no leitor de consumo de energia utiliza um circuito linear de efeito Hall ACS712. Esse fenômeno descoberto por Edwin H. Hall é caracterizado pela deflexão dos portadores de carga em um condutor com passagem de corrente elétrica. Este condutor está sobre influência de um campo magnético transversal à velocidade de deriva dos portadores. Este acúmulo de carga com sinais opostos em lados opostos da amostra cria um campo elétrico transversal à corrente, que por sua vez produz uma força elétrica que se opõe a magnética, diminuindo o fluxo transversal de elétrons (PINTO, 2010).

Um segmento de cobre nos terminais do CI (Circuito Integrado) ACS712 conduz a corrente que passa por ele, gerando um campo magnético convertido pelo circuito em uma tensão proporcional. O isolamento entre condução da rede elétrica e condução de baixa tensão é mantido, com a precisão do sensor sendo garantida pela proximidade do campo magnético e o circuito. Uma ilustração da passagem de corrente elétrica pelo sensor ACS712 é apresentada na *Figura 8*:

Figura 8 – Fluxo de corrente no CI ACS712

Fonte: Adaptada de www.sunrom.com/p/current-sensor-20a-acs712

Sua tensão de operação é de 5 volts e é capaz de mensurar até 5A, com erro de aproximadamente 1,5% a 25°C e sensibilidade de 66 a 185 mV/A. A *Figura 9* apresenta sua curva característica para cada valor de corrente analisado e seus respectivos valores de tensão de saída (tensão de alimentação V_{CC} = 5V).

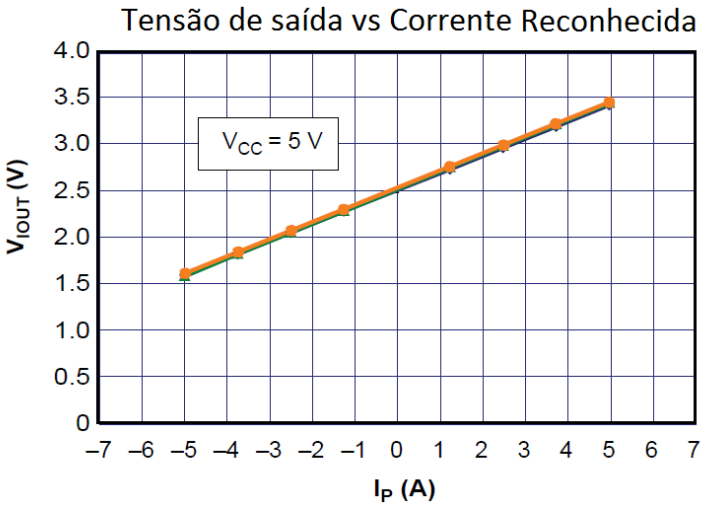

Figura 9 – Curva característica do ACS712

Fonte: Adaptada de www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf

Este sensor foi selecionado para o projeto por ser de baixo custo e invasivo, condutor é segmentado e duas partes, possibilitando uma maior precisão do que o seu semelhante SCT-013. A *Figura 10* apresenta a imagem do sensor.

Figura 10 – Sensor de corrente ACS712

Fonte: electrosome.com/shop/acs712-hall-current-sensor-module/

2.5.3. Sensor de temperatura e conversor A/D

O sensor LM35, da National Semiconductor, é um circuito integrado de precisão analógico para medição de temperatura, cuja tensão de saída de dados é linearmente proporcional a temperatura em Celsius, variando de 0 a 1V. Sua tensão de operação está no intervalo de 4 a 30 volts. Possui uma precisão de temperatura de aproximadamente 0,5°C (a +25°C) com fator de escala de 10mV/°C. Opera a temperaturas de -55 a 150°C e consome uma corrente inferior a 60μA. Possui três terminais: *Vcc*, *Data e GND* e um encapsulamento TO-92. A *Figura 11* apresenta a curva característica do erro relativo na medição de temperatura. Ao se distanciar do valor de temperatura de 25°C em ambas as direções, o coeficiente aumenta proporcionalmente para as duas versões do sensor (LM35 e LM35A).

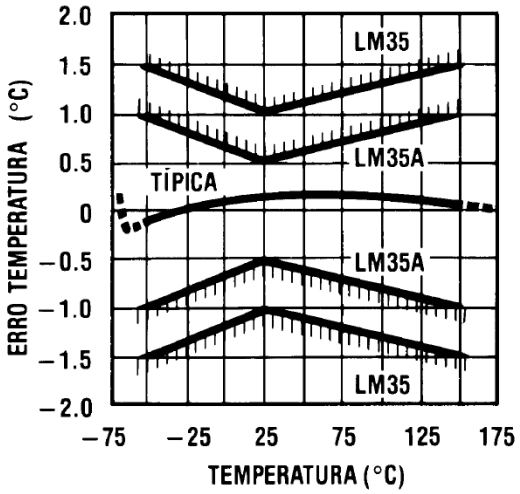

Figura 11 – Curva característica do erro no sensor LM35

Fonte: Adaptado de pdf1.alldatasheet.com/datasheet-pdf/view/8866/NSC/LM35.html

Este sensor foi selecionado para fazer parte do projeto por ser de baixo custo, pequeno e ter um baixo consumo de energia, possuindo uma boa precisão em suas medidas. A *Figura 12* apresenta a imagem do sensor. Porém, por ser um sensor analógico é necessária a utilização de um conversor analógico-digital (A/D) junto ao *Raspberry*, pois o mesmo não é habilitado a realizar leituras analógicas.

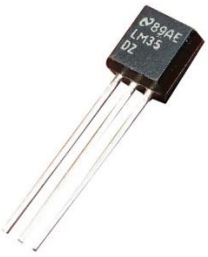

Figura 12 – LM35

Fonte: blog.vidadesilicio.com.br/Arduino/basico/lm35-medindo-temperatura-com-Arduino/

O conversor A/D PCF8591 é um CMOS (Complementary Metal-Oxide-Semiconductor) de baixo consumo de energia e resolução de 8 *bits*. Possui quatro canais de entrada analógica e três pinos de endereço usados para a programação dos mesmos. A comunicação é realizada através da trilha I^2C bidirecional, multiplexando entradas analógicas em até 8 dispositivos e convertendo valores analógicos em digitais (e vice-versa). Opera na faixa de tensão entre 2,5 e 6V, possuindo um pino reservado para armazenar tensões de referência. A *Figura 13* apresenta a imagem do conversor com a descrição dos seus pinos.

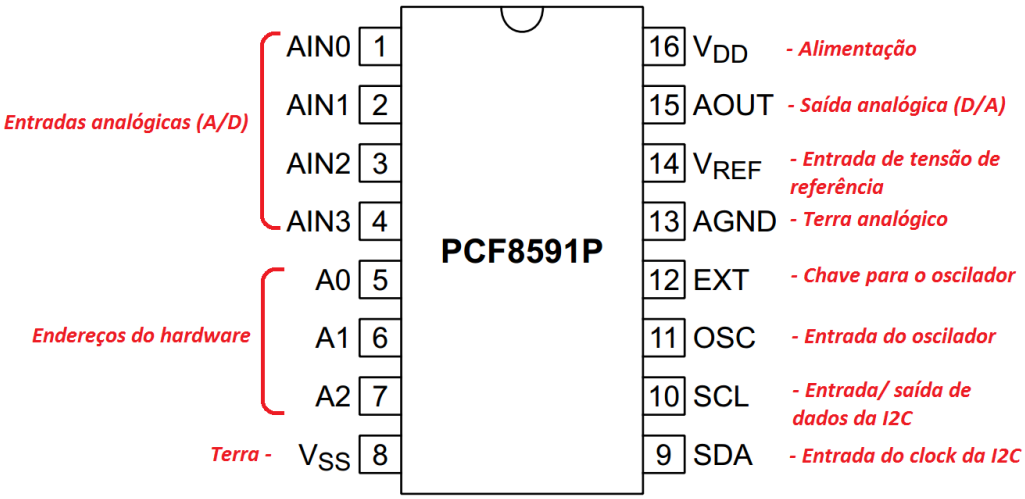

Figura 13 – PCF8591 e descrição de pinos

Fonte: Adaptada de www.nxp.com/documents/data_sheet/PCF8591.pdf

O PCF8591 foi selecionado para fazer parte do projeto por complementar o funcionamento

do sensor analógico, possuir uma resolução razoável de 8 bits e utilizar a comunicação I2C (comunicação direta entre CIs), assim como o Raspberry.

2.5.4. Sensor de presença

Os sensores de presença usados nas residências são dispositivos eletrônicos capazes de detectar a movimentação de corpos em um raio de ação. Estes são do tipo PIR (Passive Infrared) e analisam a luz infravermelha emitida por objetos que emitem calor.

Dispositivos PIR possuem dois sensores sensíveis a luz infravermelha direcionados ao ambiente. Estes sensores no mercado variam em sensibilidade. A alteração nas características de alcance, padrão de detecção e raio de detecção variam com o efeito ótico das lentes utilizadas. As lentes são de plástico do tipo Fresnel, inventada pelo físico Augustin Fresnel. Estas lentes são compactas e condensam a luz, oferecendo uma gama maior de raios infravermelhos para o sensor. Uma ilustração da lente Fresnel em comparação com uma convencional (esquerda) e o efeito dos raios incidentes na lente Fresnel (direita) é apresentada na *Figura 14*.

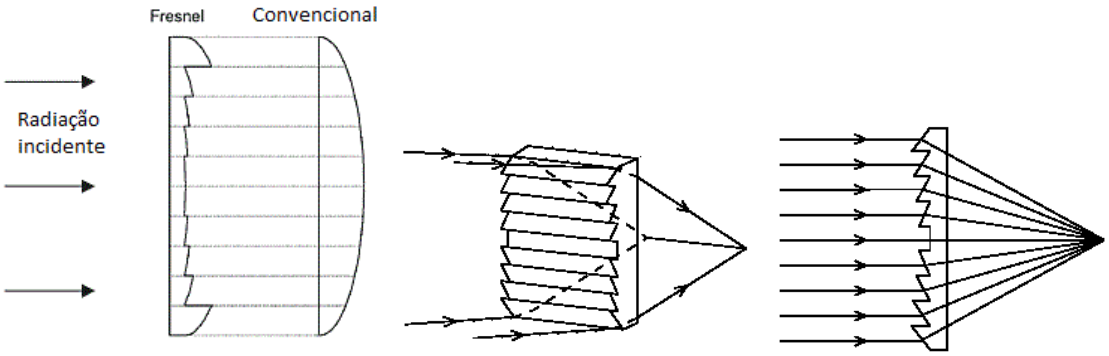

Figura 14 – Lente Fresnel

Fonte: Adaptada de learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work

Quando o dispositivo está inativo, ambos sensores recebem a mesma quantidade de IR (Infravermelho), a quantidade de radiação do ambiente. Quando um corpo que emite radiação IR entra no cômodo, inicialmente o primeiro sensor recebe a diferença de radiação, causando uma mudança diferencial positiva entre as metades (3,3V). Quando este mesmo corpo se distancia do alcance do sensor, o contrário acontece e outro sinal é gerado, porém causando uma mudança diferencial negativa (-3,3). A *Figura 15* apresenta a curva característica do sensor (esquerda) e uma ilustração do processo (direita).

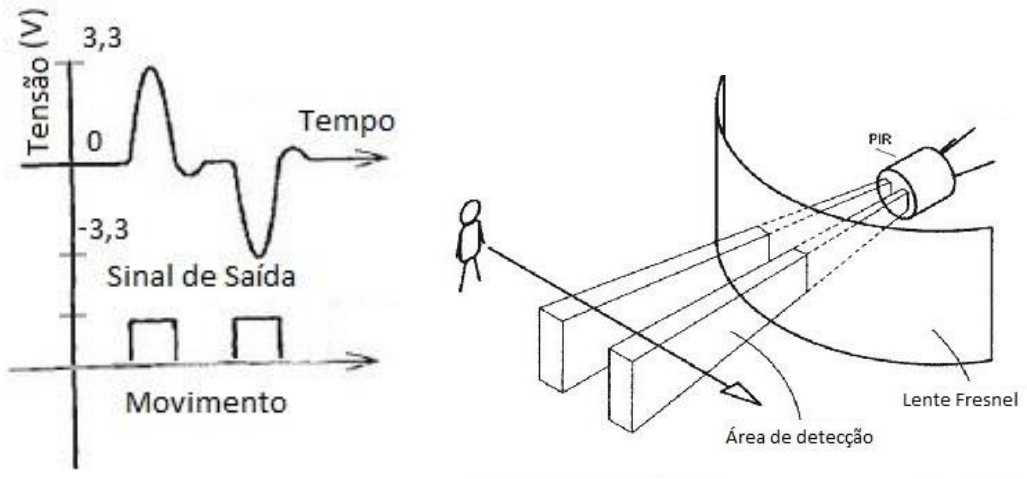

Figura 15 – Funcionamento do sensor PIR

Fonte: Adaptada de learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work

O PIR HC-SR501 é utilizado no projeto como sensor de presença. Este possui tensão de operação de 5 a 20 volts, sinal de saída binário (0/ 3,3 volts) e um *delay* ajustável de 0,3 a 5 minutos. Possui um raio de alcance de 120° até 7 metros (ajustável) e funciona de -15 até 70°C. Foi selecionado para o projeto por ser de baixo custo e possuir uma distância e *delay* ajustáveis. A *Figura 16* apresenta a imagem do sensor.

Figura 16 – Sensor PIR

Fonte: www.buildbot.com.br/produto/sensor-infravermelho-de-presenca-pir-hc-sr501/

2.5.5. Relé eletromecânico

Outro componente que utiliza o conceito de indução eletromagnética é o relé eletromecânico. Este caracteriza-se por ser uma chave móvel atraída por um eletroímã quando corrente elétrica o percorre. O acionamento do eletroímã é feito através de um circuito de baixa tensão isolado fisicamente de um segundo circuito. Este último é onde a chave está localizada e, geralmente, é conectado a tensões maiores, como a da rede elétrica (CHARLES e MATTHEW, 2013). A chave possui três modos de acionamento possíveis: Normalmente aberta (NA),

normalmente fechada (NF) e comutada. A *Figura 17* apresenta um circuito de um relé eletromecânico em notação elétrica (esquerda) e ilustrativa (direita). Os terminais da bobina são conectados ao circuito de baixa tensão. A corrente de controle que percorre a bobina produz um campo magnético forte o suficiente para atrair a armadura de sua posição inicial e fechar a chave. No caso da imagem, o contato é comutado, pois alterna entre o circuito NF e NA.

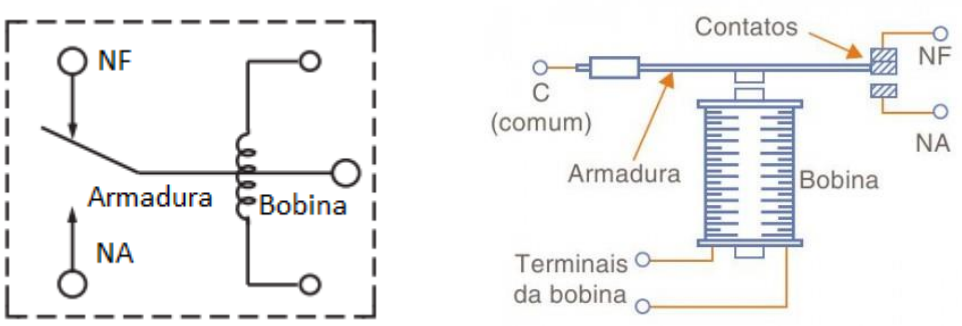

Figura 17 – Relé eletromecânico

Fonte: www.sabereletronica.com.br/artigos/1702-testando-rels

Um dos benefícios do relé é sua corrente baixa utilizada para atrair a chave, permitindo seu acionamento por circuitos eletrônicos fracos. Outra vantagem é a segurança dada pelo isolamento físico entre circuito de baixa e alta tensão. O projeto utiliza os reles eletromecânicos de 5V e 7A para acionar as cargas dos equipamentos controlados através do sistema principal. A *Figura 18* apresenta a curva característica do relé eletromecânico. A potência máxima de operação do componente é limitada pela corrente máxima no contato e a tensão máxima no contato, respectivamente 7A e 250VAC.

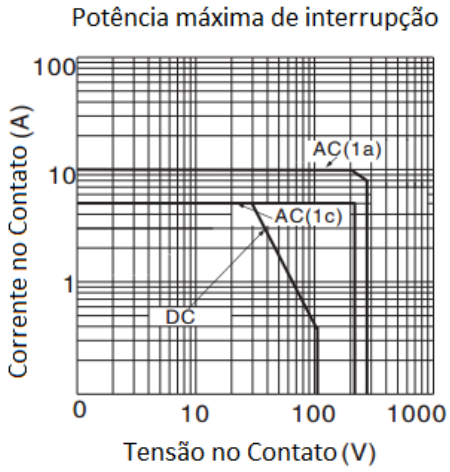

Figura 18 – Curva característica do relé

Fonte: Adaptado de www.sanyourelay.ca/public/products/pdf/SRD.pdf

2.5.6. Plataforma *Arduino*

O *Arduino* é uma plataforma *Open Source* de desenvolvimento de protótipos eletrônicos amplamente utilizada atualmente pelo seu baixo custo, grande apoio da comunidade livre e pela sua facilidade de uso. Utilizando um microcontrolador Atmel AVR, as várias modificações e tipos de placas *Arduino* existentes são capazes de exercer diversas funções desde a leitura de sensores básicos até a automação e controle de motores e atuadores. Alguns modelos ainda permitem o acoplamento de outras placas chamadas s*hields*, adicionando mais variedades de funções a gama de aplicações existentes.

O projeto utiliza o *Arduino* UNO R3, apresentado na *Figura 19.* Seu esquemático elétrico é apresentado na *Figura 20*. Este modelo possui um micro controlador ATmega328 e um ATmega16U2. O primeiro é ligado à todas as portas de E/S e alimentação disponíveis na plataforma, além das portas ICSP (*In-System Programming*), usadas para a instalação do *bootloader* no micro controlador. O segundo possui todas as ligações necessárias para utilização da interface USB, além da ICSP de mesma função da primeira. Outros resistores, capacitores, diodos, amplificadores operacionais, reguladores de tensão e conectores de uso diverso também compõem a plataforma. O *bootloader* é um programa gravado na memória do micro controlador que simplifica o carregamento de programas para o chip de memória flash *(ARDUINO, 2015)*.

Sua tensão de operação é de 5 volts, um cristal de 16MHz é utilizado para gerar a frequência (*clock*) em que as instruções são executadas. Este ainda possui uma memória Flash de 32 kB (*0,5 kB reservados ao bootloader*) para armazenamento de programas, 2 kB de SRAM e 1 kB para a EEPROM. 20 pinos de E/S, sendo 14 deles digitais (6 com PWM) e 6 analógicos estão disponíveis para utilização *(ARDUINO, 2015)*. Possui um buffer de dados na serial de 64 bytes, com uma fila utilizando o algoritmo *round robin*.

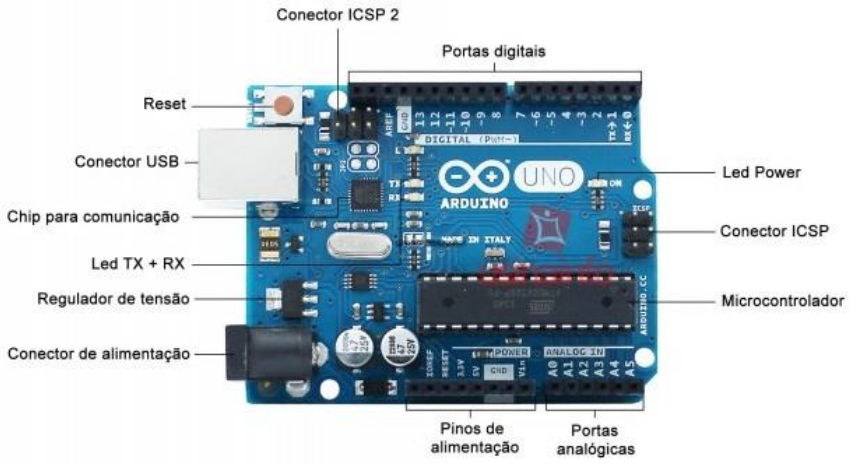

Figura 19 – Arduino UNO

Fonte: www.equibancada.com.br/produto/Arduino-UNO.html

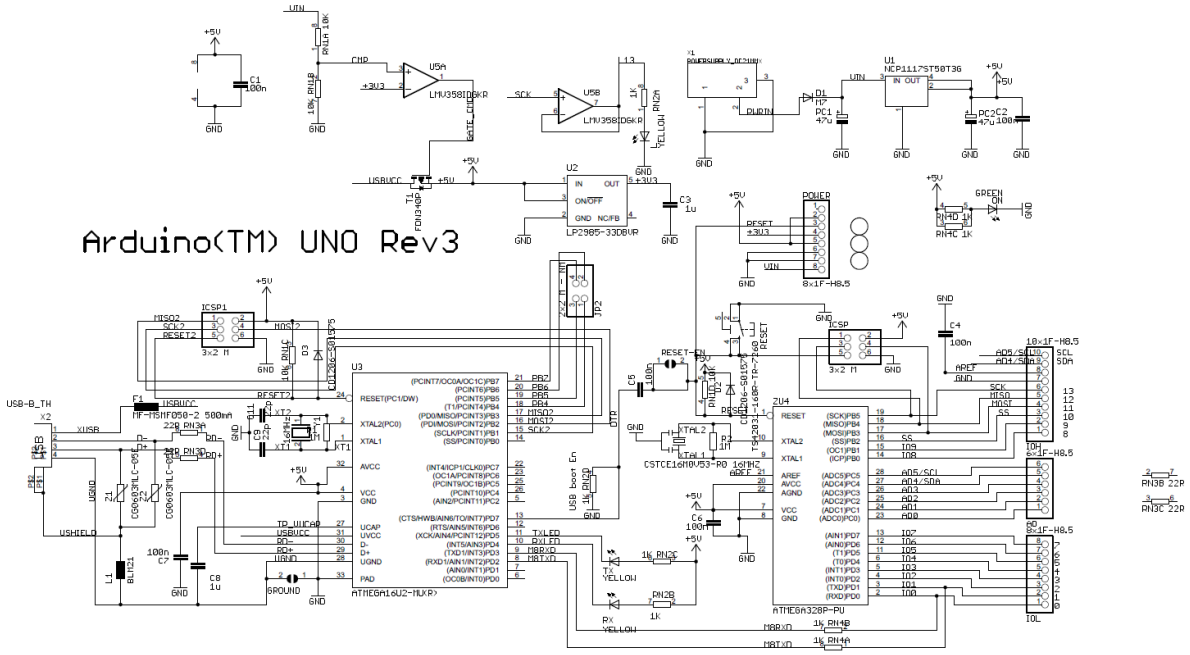

Figura 20 – Esquemático elétrico Arduino UNO

Fonte: www.Arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

A IDE (*Integrated Development Environment*) do *Arduino*, utilizada para desenvolver os *sketches* (códigos-fonte), foi desenvolvida a partir da IDE da ferramenta *Open Source Processing*, implementada em Java. A linguagem de programação, essencialmente C e C++, é derivada do *Wiring*, plataforma de prototipagem e framework também *Open Source* para microcontroladores. Sua interface é simples e concisa, disponibilizando o necessário para o desenvolvimento de um projeto. O código gerado possui formato próprio (.*ino*) identificado pela IDE (*Itegrated Development Environment*). Uma janela com o programa exemplo *Blink* é apresentada na *Figura 21*

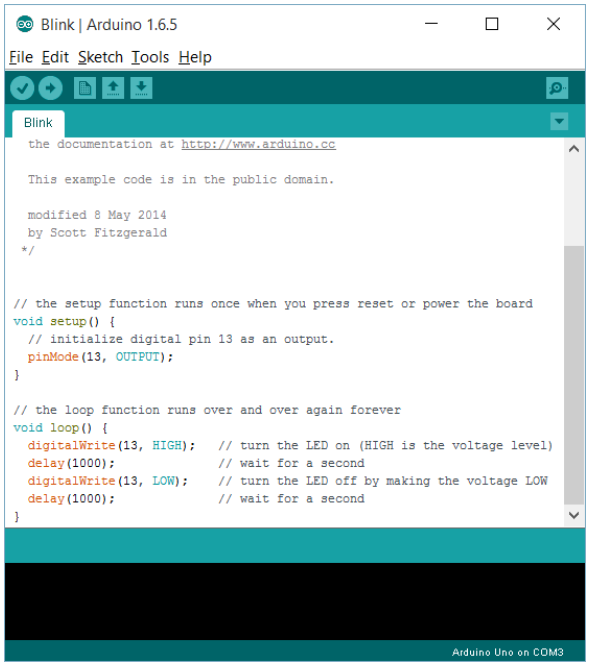

Figura 21 – IDE com o exemplo Blink

Fonte: www.Arduino.cc/en/Guide/Windows

O *Arduino* é uma plataforma *Open Source*, de baixo custo, seu software é livre e sua IDE possui uma biblioteca própria para leitura de corrente e tensão, chamada *EmonLib*, descrita no *Capítulo 3*. Estes fatores contribuíram para a integração da plataforma no projeto.

2.5.7. Plataforma *Raspberry*

A plataforma *Raspberry* é um computador de baixo custo, pequeno, O*pen Source* e com interfaces para vários periféricos. Foi desenvolvido pela fundação *Raspberry* PI, no Reino Unido, com o objetivo de aprimorar o ensino, na área computacional, de crianças e adultos. O modelo mais recente lançado é o *Raspberry* Pi 2 Modelo B, projetado para substituir a versão anterior, o *Raspberry* Modelo B+. A *Figura 22* apresenta o modelo B. Este possui um processador ARM Cortex-A7 com quatro núcleos de 900MHz, memória RAM de 1GB compartilhada com a GPU (*Graphics Processing Unit*), GPU *Broadcom VideoCore* IV 3D, alimentação de 5V, 4 portas com interface USB 2.0, 40 pinos E/S de uso geral (GPIO), porta HDMI (*High-Definition Multimedia Interface*), porta Ethernet 10/ 1000MB com interface RJ-45, interface de cartão SD, interface para câmera e interface para um display (RASPBERRY FOUNDATION, 2015). A *Figura 22* apresenta uma imagem da plataforma e a pinagem padrão do *Raspberry* Pi 2 Modelo B é apresentada na *Figura 23*.

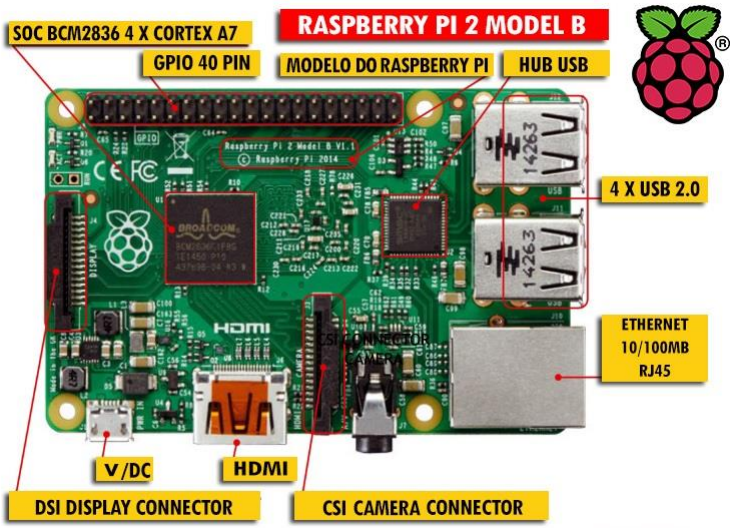

Figura 22 – Raspberry PI 2 Modelo B

Fonte: Adaptado de www.raspberryshop.es/

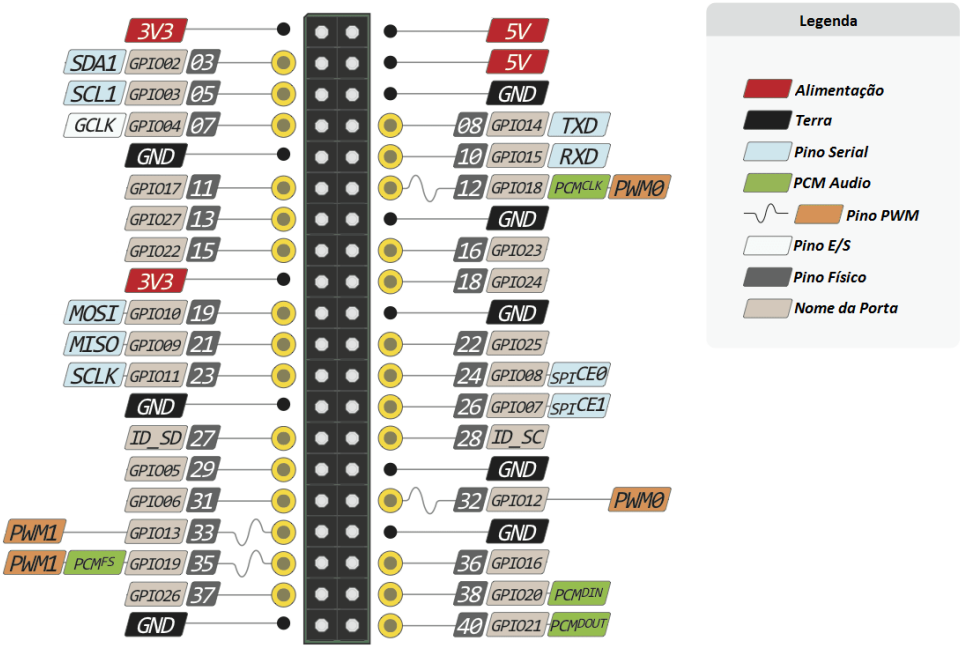

Figura 23 – Pinos do Raspberry PI 2 Modelo B

Fonte: Adaptado de www.megaleecher.net/Raspberry_Pi_2_Schematic_And_Pinout_Diagram

O sistema operacional (SO) do *Raspberry* é instalado no cartão SD. Existem várias versões diferentes de SOs compatíveis com a plataforma, cada um deles com suas próprias características. Para instalar um SO é necessário formatar o cartão no sistema de arquivos FAT32 (File Allocation Table). O SO utilizado no projeto é o padrão disponibilizado no site do *Raspberry*, chamado *Raspibian Jessie* (Versão do *Kernel 4.1*), baseado em *Linux (Debian*). Após a instalação e configuração do mesmo, a plataforma estará pronta para a utilização. Os periféricos necessários

são: um monitor, um teclado e um mouse. Com um cabo *ethernet* ou um adaptador WI-FI é possível ter acesso a uma rede LAN *(Local Area Network*) local. Assim, as preparações são realizadas, a plataforma ligada e a tela inicial da área de trabalho é exibida, como apresentado na *Figura 24*.

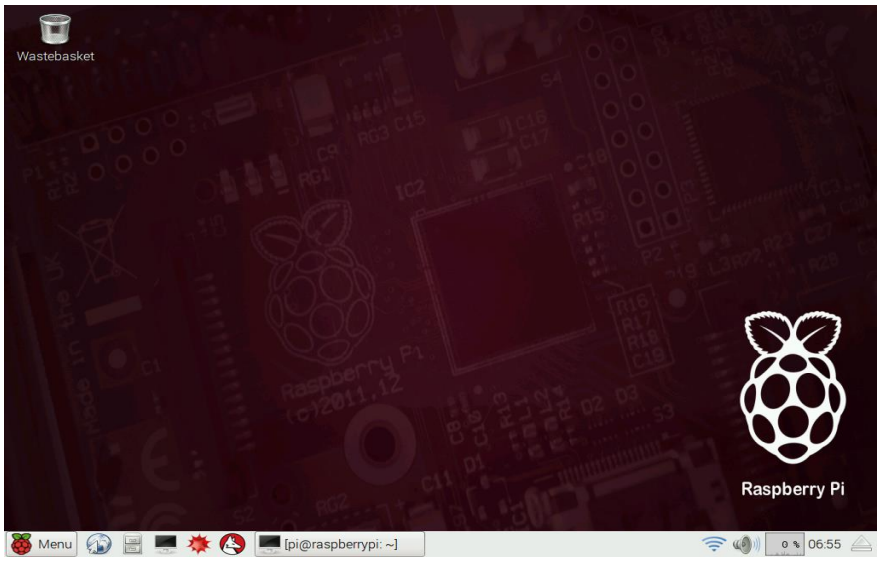

Figura 24 – Raspbian Desktop

Fonte: Elaborada pelo autor

A plataforma *Raspberry* foi selecionada para integrar o projeto pela sua alta capacidade de processamento, necessária para gerenciar os leitores de consumo, seu baixo custo e disponibilidade de código aberto. Códigos fonte podem ser desenvolvidos utilizando várias IDEs diferentes, dentre elas, o *Scratch* e a nativa em *Python*. A selecionada para o projeto é denominada *Geany* (versão 1.24.1) e possui uma interface de projetos com abas adicionais com contagem de símbolos em utilização. O código fonte do sistema principal é desenvolvido em *Python*, através da *IDE Geany no Raspberry*.

Para o desenvolvimento da interface gráfica (*GUI*) do projeto é utilizada a biblioteca GTK+ versão 2.24.23 através do módulo *PyGTK* específico para a linguagem *Python*. Esta aplicação livre orientada a objeto provê uma grande diversidade de elementos visuais e *widgets* para criação de aplicações de forma fácil e rápida (PYGTK TEAM, 2009). O PyGTK é constituído de vários módulos, dentre eles a classe base GObject e a própria GTK. Os objetos *widgets* criados na GTK são divididos em *widgets* filhos (*childs*) e *widgets* contêineres (*containers*), com os primeiros sendo alocados dentro dos segundos. Desta forma, os objetos são separados e organizados pelos contêineres contendo conjuntos de *widgets* filhos.

Com os valores de leitura recebidos pelo *Raspberry* os gráficos dos mesmos são gerados pelo utilitário *Gnuplot*. Esta ferramenta de gráficos 2D e 3D pode ser utilizada em várias

plataformas diferentes, sendo distribuída gratuitamente (GNUPLOT, 2015). O mesmo é executado em linha de comando pelo terminal do *Raspberry* e, no projeto, é chamado automaticamente na *GUI* por um botão específico.

2.5.8. Comunicação serial USB

A comunicação entre os leitores de consumo de energia e o sistema principal é realizada pela porta serial USB de ambas as plataformas (*Arduino* e *Raspberry*). O USB (Universal Serial Bus) é um padrão de conectividade especificado pelas indústrias líderes no mercado desde 1995, sendo atualizado constantemente até hoje e amplamente utilizado em diversos produtos (INTEL, 2015). A construção da versão 2.0 é baseada em quatro fios encapsulados em uma blindagem de malha metálica e poliéster metalizado. Dois dos fios presentes se destinam aos condutores 5V e terra, os outros dois fios são par trancados (D+ e D-) de dados. As blindagens têm o objetivo de isolar a transmissão de dados de influências eletromagnéticas externas (INTEL, , *et al.*, 2000). A *Figura 25* apresenta a construção de um cabo USB 2.0.

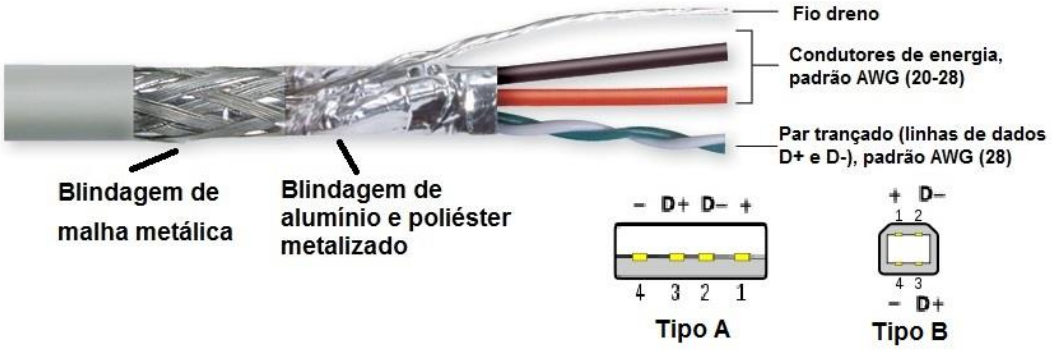

Figura 25 – Construção de um cabo USB 2.0

Fonte: Adaptado de www.l-com.com/content/USB-Tutorial.html

As taxas de transmissão nos cabos USB 2.0 podem chegar a até 480Mb/s por até 5 metros. O *clock* é transmitido e codificado juntamente com os dados. O esquema de codificação do *clock* é o *Not-Return-to-Zero Inverted* (NRZI) com inserção de *bits* (*bit stuffing*), para garantir transições adequadas. O padrão USB é orientado a *hosts*, o que significa que apenas um dispositivo pode ser definido como *master* em uma conexão. Este é responsável por realizar todas as operações e planejar a largura de banda. Possui uma topologia em estrela com camadas, impondo *hubs* para conexão ao *host*. Cada *hub* tem o controle da alimentação do dispositivo conectado a ele, bem como a filtragem das transações em baixa, média e alta velocidade.

Uma transação normal consiste num pacote *token*, um pacote de dados e um pacote de estado (*handshake*). Um pacote é constituído de elementos de controle da informação, dados do usuário e bits de correção/ detecção de erros, assim como em outras tecnologias de comunicação entre dispositivos. O host inicia a comunicação enviando um *token*, descrevendo características da transação que se seguirá. O pacote seguinte representa os dados em sí (*payload*), seguido de um pacote de *handshake* para confirmação do sucesso da transação (PEACOCK, 2010). A *Figura 26* apresenta as estruturas de um pacote *token*, dados e *handshake* e seus respectivos tamanhos em *bits*. O campo *Sync* é utilizado para sincronizar os relógios entre dispositivos. O campo *PID* (*Packet Identifier*) determina o tipo e formato do pacote. O campo *ADDR* representa o endereço destino do dispositivo (de 1 a 127). *ENDP* especifica o tipo de transação a ser realizada, assim como o campo *Data* representa os dados (INTEL, , *et al.*, 2000) enviados e recebidos. O campo EOP representa o encerramento de um pacote. *CRCX* (*Cyclic Redundancy Checks*) representa a sequência de *bits* onde será realizado o método CRC de identificação de erros, onde '*X*' representa a ordem do polinômio verificador, no método (KUROSE, 2009).

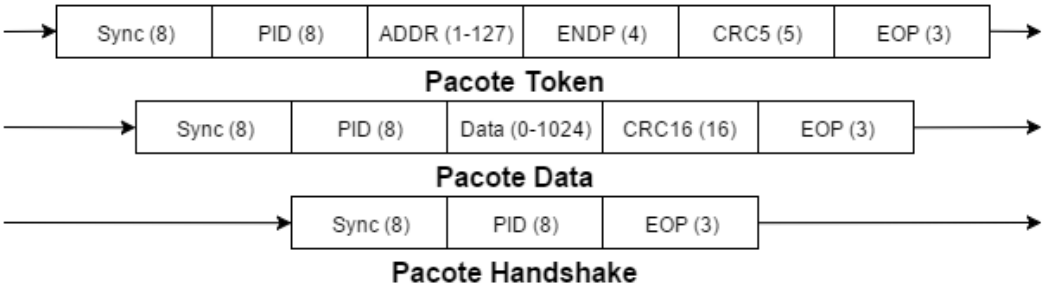

Figura 26 – Estrutura dos pacotes no protocolo USB

Fonte: Adaptada de www.beyondlogic.org/usbnutshell/usb3.shtml

A comunicação USB foi selecionada para o projeto por ser a forma padrão de comunicação entre *Arduino* e computador, não sendo necessário custos e ligações adicionais. A maior taxa de transferência em *bauds* especificada no site (ARDUINO, 2015) é de 115200 (outras taxas são possíveis) assim como a padrão definida no sistema operacional do *Raspberry* (outras taxas também são possíveis).

CAPÍTULO 3 – DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo apresenta o desenvolvimento do leitor de consumo de energia, do sistema de controle e monitoramento e a transformação dos dados de consumo em sinais de controle para os atuadores. Estes passos são necessários para a obtenção do produto final, utilizando os conceitos e materiais apresentados no capítulo anterior e que serão descritos a seguir.

3.1. Descrição do sistema proposto

O sistema proposto visa monitorar os valores de consumo de uma residência com dois cômodos e apresentar um equipamento específico para ambos. Um ventilador e a iluminação da residência serão controlados através de atuadores, com o intuito de reduzir este consumo e notificar o usuário do mesmo. O leitor de consumo consiste na captação dos valores de tensão e da corrente da rede elétrica. A partir destes valores, serão realizados os cálculos de potência ativa, aparente e fator de potência. O sistema principal consiste na captação dos valores de consumo de energia provenientes dos leitores mencionados e acionamento dos respectivos atuadores através dos leitores de consumo. A *Figura 27* apresenta o diagrama de blocos do projeto, assim dividido, para melhor visualização e desenvolvimento.

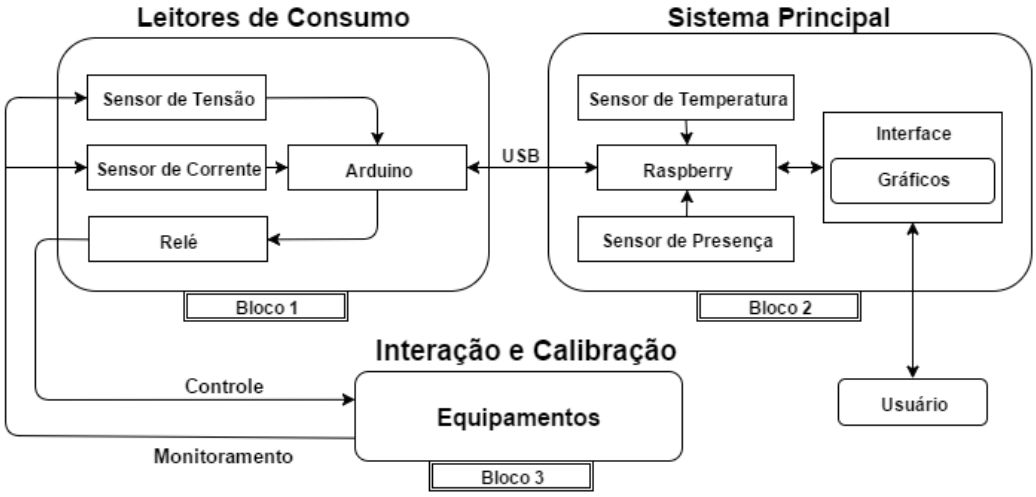

Figura 27 – Diagrama de blocos do projeto

Fonte: Elaborada pelo autor, utilizando ferramenta Flow Chart Maker: www.draw.io

O Bloco 1 representa o conjunto dos leitores de consumo de energia. Cada circuito consiste de um Arduino Uno, sensor de corrente, sensor de tensão e relé eletromecânico. O Bloco 2 se refere ao conjunto constituído de Raspberry e sensores relacionados. Este recebe os dados de consumo de energia e os traduz em sinais de controle de volta aos leitores, além de apresentar os gráficos de consumo na interface com o usuário. O Bloco 3 representa a interação e calibração entre sinais de controle e equipamentos de cada cômodo.

3.2.1. Leitor de Consumo de Energia (BLOCO 1)

O leitor de consumo de energia é controlado pela plataforma *Arduino* e conectado à rede elétrica através do sensor de corrente e tensão. Os sinais são recebidos pela plataforma e enviados para o sistema principal. Sinais de controle são recebidos da plataforma *Raspberry*, acionando ou desligando seus respectivos relés.

3.2.2. Montagem do protótipo

A interpretação dos sinais nos sensores é feita através da biblioteca *EmonLib.h* utilizada no *Arduino* e disponibilizada pelo site *OpenEnergyMonitor*. O objetivo deste site é desenvolver ferramentas de código aberto relacionadas a monitoramento de energia. O sensor de tensão utilizado no leitor de consumo segue o esquemático elétrico sugerido no site e apresentado na *Figura 28*.

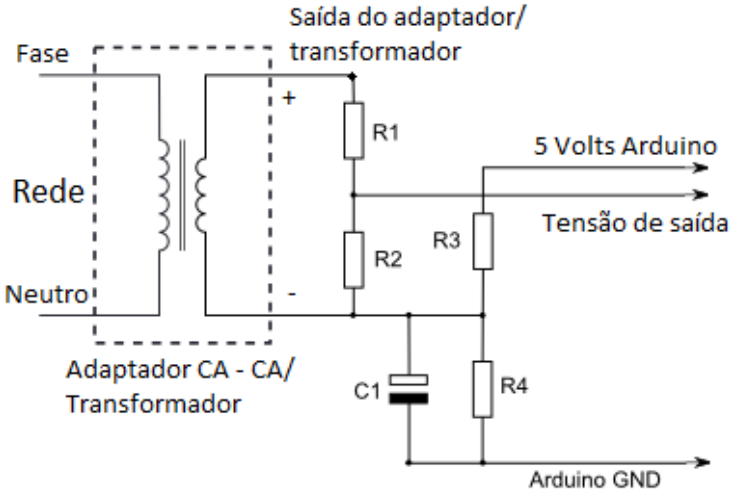

Figura 28 – Diagrama elétrico do sensor de tensão

Fonte: Adaptada de openenergymonitor.org/emon/buildingblocks/measuring-voltage-with-an-acac-power-adapter

Na Figura, os condutores fase e neutro da rede são conectados à um adaptador de corrente alternada redutor de tensão (CA - CA). No leitor de consumo de energia, do projeto, é utilizado um transformador 220/6+6V responsável por reduzir a tensão da rede. Os condutores 220V (vermelho) e terra (preto) do primário são conectados à rede e um dos condutores 6V (vermelho) e a derivação central (preto) do secundário são conectados ao restante do circuito. Dessa forma, a tensão é reduzida de 220V para 6V RMS na saída do transformador, ainda em corrente alternada.

A tensão da saída de 6V RMS (±8,48V valor máximo e mínimo) alternada do transformador deve ser reduzida a um valor adequado à entrada analógica do *Arduino*, capaz de operar com tensões positivas entre 0 e a tensão de referência da plataforma (5V no caso do Uno) (ARDUINO, 2015). Esta redução, então, deve manter o valor máximo de pico positivo menor que 5V e o valor máximo de pico negativo maior que 0V, faixa de operação das portas do *Arduino*. O diagrama apresenta um divisor de tensão, representado pelos resistores R1 e R2, responsável por reduzir o sinal de tensão proporcionalmente aos valores destes resistores, além de permitir uma margem de variação para este sinal. A *Equação 26* apresenta este divisor de tensão. No projeto, o sinal de tensão é reduzido variando entre 1,057V a -1,057V na porta analógica A2 do *Arduino*, utilizando um resistor de 330KΩ como R1 e 47KΩ como R2. Quanto maior o valor da resistência do resistor, menor será a corrente que o percorre e, consequentemente, menor será o consumo de energia do circuito.

$$\boldsymbol{V_{saída} = \frac{R_2}{R_1+R_2}\, x\, V_{entrada} = \frac{47K}{330K+47K}\, x\, 8,48V = 1,057V} \quad \textit{(26)}$$

O sinal de tensão de saída reduzido ainda permanece variando entre valores positivos e negativos. O segundo divisor de tensão, representado pelos resistores R3 e R4, é responsável por causar um *offset* (deslocamento) no sinal para que o mesmo somente adquira valores positivos. Em um divisor de tensão com resistores de valores iguais, a tensão de saída é dividida exatamente pela metade. Conectando os terminais deste divisor de tensão na porta de alimentação 5V e na porta GND do *Arduino*, a tensão de saída é dividida por dois (2,5V). No diagrama, esta tensão é conectada na derivação central do transformador (terminal negativo) alterando a referência do primeiro divisor de tensão e deslocando o sinal deste em 2,5V. A *Equação 27* apresenta o segundo divisor de tensão do diagrama.

$$\boldsymbol{V_{saída} = \frac{R_3}{R_3+R_4}\, x\, V_{entrada} = \frac{470K}{470K+470K}\, x\, 5V = 2,5V} \quad \textit{(27)}$$

Com o deslocamento de 2,5V o sinal de saída para a porta A2 do *Arduino* passa a variar entre um pico máximo de 3,557V (2,5+1,057) e o pico mínimo de 1,443V (2,5-1,057), dentro da faixa de operação designada. O capacitor C1 utilizado tem o objetivo de filtrar a parte CC do sinal, permitindo somente a passagem da parte AC de volta ao transformador. A *Figura 29* apresenta a relação entre os terminais do sensor e a forma do sinal de tensão.

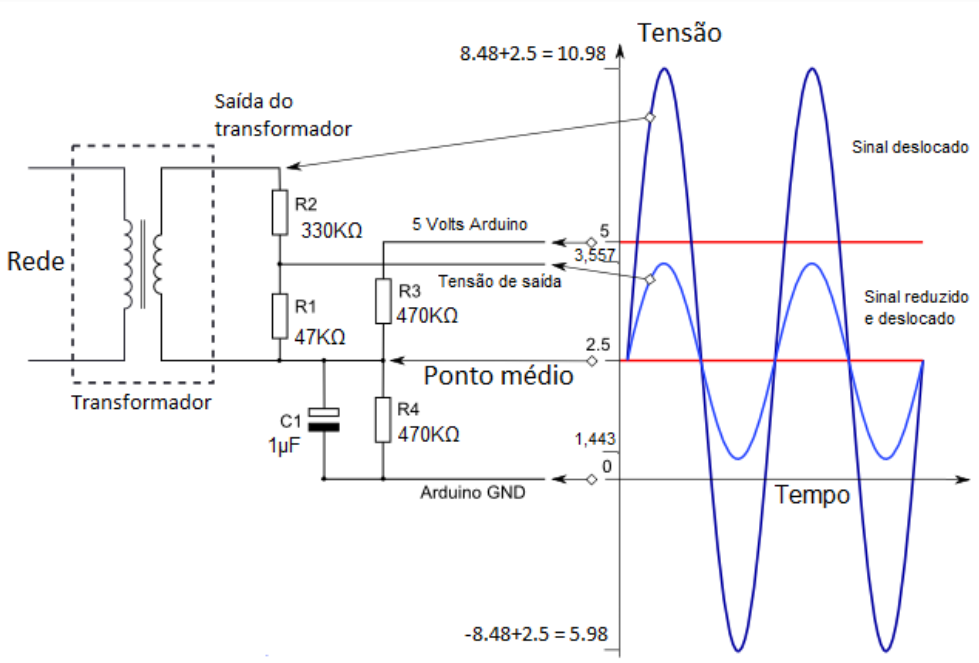

Figura 29 – Relação entre os terminais do sensor e a forma do sinal de tensão

Fonte: Adaptada de openenergymonitor.org/emon/buildingblocks/measuring-voltage-with-an-acac-power-adapter

O sensor de corrente ACS712, descrito no *Capítulo 2*, é utilizado no leitor de consumo em conjunto com o sensor de tensão. Um dos condutores da rede elétrica (fase ou neutro) é segmentado e o sensor de corrente, conectado nos seus dois terminais. O pino *VCC*, o pino *GND* e o pino *OUT* são conectados às portas de alimentação 5V, porta GND e porta analógica A1 do *Arduino* respectivamente. A *Figura 30* apresenta as conexões do leitor de consumo. O sensor de corrente, como dito anteriormente, é conectado à um condutor da rede elétrica e ao *Arduino*. Sua representação na figura corresponde ao seu esquemático elétrico, constituído de CI ACS712 e capacitores utilizados como filtro. O sensor de tensão é conectado aos dois condutores da rede elétrica e, também, ao *Arduino*. Um dos terminais do secundário (6V) e o terminal de 110V do primário não são utilizados, sendo consequentemente aterrados. O conector *CARGA* é ligado ao equipamento a ser monitorado e o conector *REDE*, ligado à própria rede elétrica. O capacitor C4 de 10uF, conectado à porta externa de RESET e ao terra, é responsável por impedir que o sinal de reset chegue ao *Arduino* sempre que a serial é acessada, mantendo o mesmo ligado continuamente

e impedindo alteração de valores advindos da reinicialização (ARDUINO, 2015).

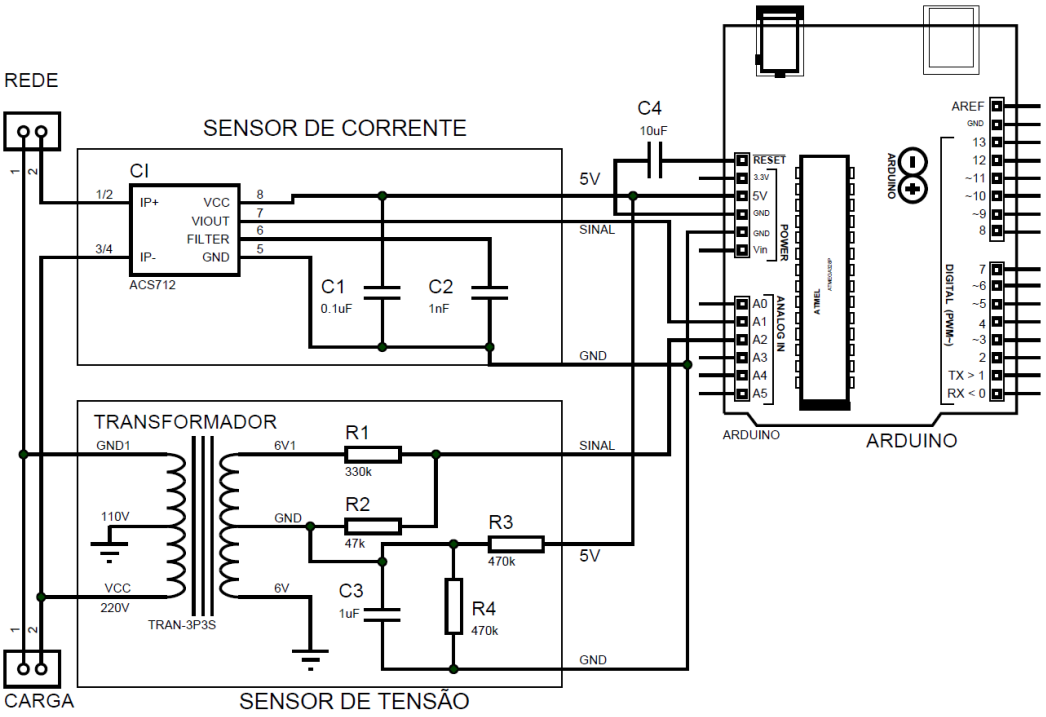

Figura 30 – Conexões do leitor de consumo

Fonte: Elaborada pelo autor

O acionamento das cargas é realizado através do relé eletromecânico de 5V e 7A. O sinal de acionamento é gerado pelo *Arduino* na porta A4 e amplificado no transistor. Como a corrente nominal necessária para acionar o relé é de aproximadamente 71,42mA e a corrente máxima gerada pelas portas da plataforma é de 40mA, esta amplificação é necessária e realizada através do BC337 (Transistor de junção bipolar NPN). Para este transistor o valor de ganho (β) varia entre 100 e 600, dependendo da frequência de acionamento, temperatura ambiente e nível de dopagem na fabricação. A corrente no terminal coletor (I_C) deve ser de 72mA, como já mencionado, resultando na equação para obtenção da corrente na base (I_B), abaixo (BOYLESTAD e NASHELSKY, 1999):

$$\boldsymbol{\beta = \frac{I_C}{I_B} \rightarrow I_{Base} = \frac{0{,}00072}{100} = 720\mu A} \qquad \textit{(28)}$$

Para uma corrente de 72µA na saída da porta do *Arduino* é preciso determinar a resistência mínima entre terminal base e porta A4 com um resistor, sendo a queda de tensão necessária de 5V subtraído da tensão de polarização direta (0,7V para o silício, material do transistor). A equação abaixo apresenta esta relação:

$$R_{Base} = \frac{V}{I_B} \rightarrow R_{Base} = \frac{(5-0{,}7)}{720\mu A} \sim 6K\Omega \qquad (29)$$

O LED presente é usado para indicar visualmente o acionamento da carga pelo relé. Um diodo emissor de luz (LED) vermelho opera na faixa de 1,8V a 2V com uma corrente nominal de 20mA, sendo necessário um resistor ligado entre sua perna positiva e o 5V do *Arduino*. A equação abaixo apresenta o cálculo desta resistência mínima:

$$R_{LED} = \frac{Vcc-VF}{I_{LED}} \rightarrow R_{LED} = \frac{(5-2)}{20mA} = 150\Omega \qquad (30)$$

Um diodo não conduz em polarização reversa, sendo utilizado o 1N4148 como diodo flyback para bloquear o pico de tensão característico do desligamento de cargas indutivas. Seguindo o mesmo esquemático elétrico da *Figura 30*, porém, com a substituição do esquemático do sensor de corrente por um conector borne de três pinos para o encaixe do mesmo e a inclusão do esquemático elétrico do atuador de controle, a *Figura 31* apresenta o esquemático final do leitor de consumo de energia e a *Figura 32* apresenta o protótipo final do mesmo.

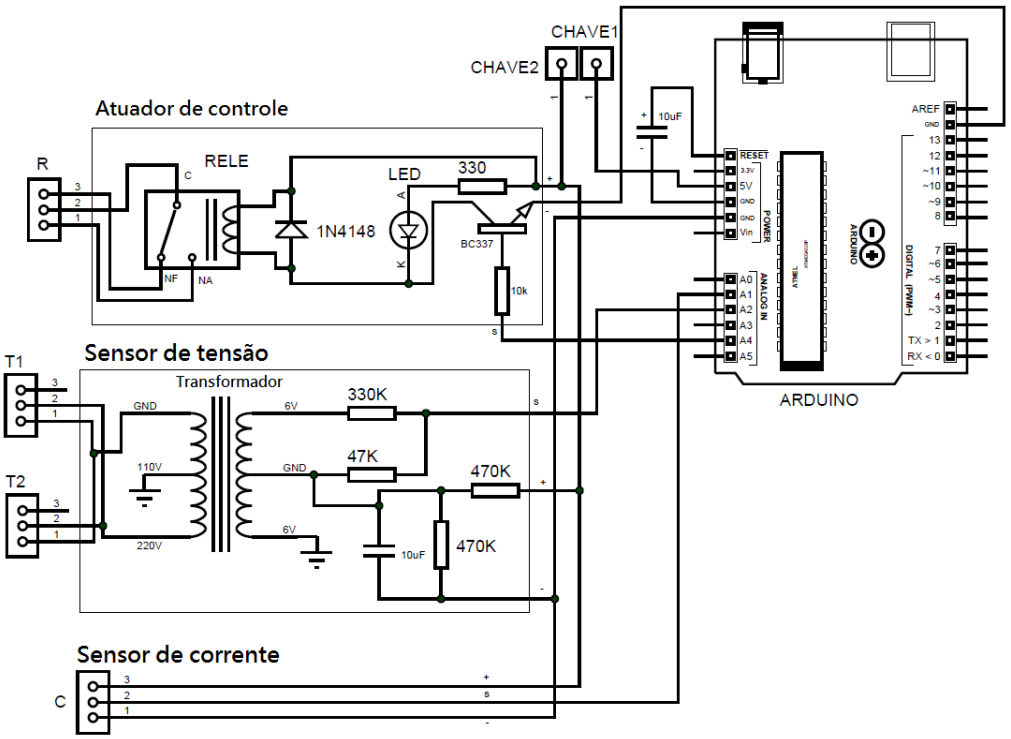

Figura 31 – Esquemático elétrico final do leitor de consumo

Fonte: Elaborada pelo autor

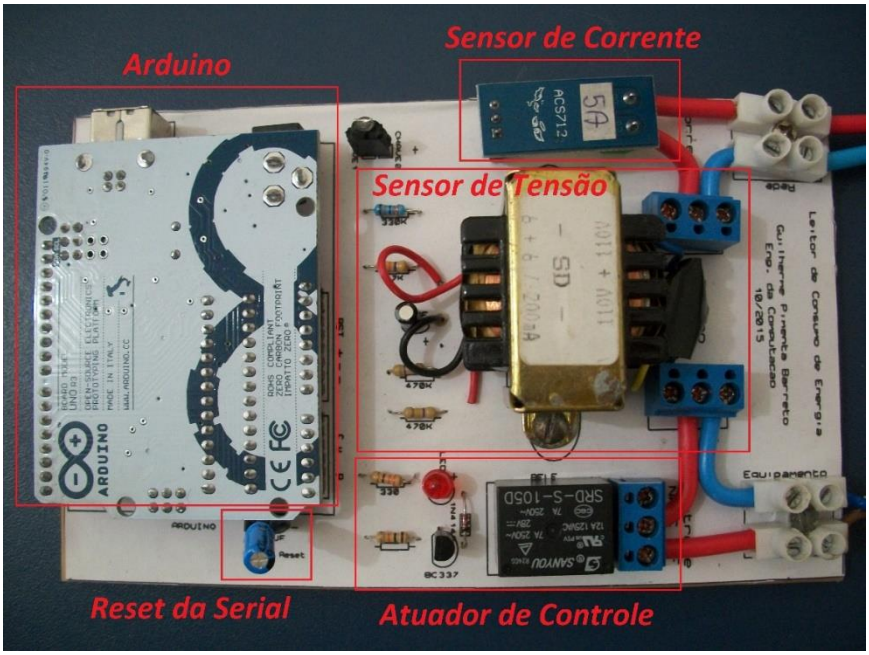

Figura 32 – Protótipo final do leitor de consumo

Fonte: Elaborada pelo autor

3.2.3. Código fonte

O código em C, apresentado no APÊNDICE A e utilizado no *Arduino* do protótipo, utiliza os comandos de leitura do exemplo *voltage_and_current* da biblioteca *EmonLib.h.* A descrição dos comandos é exibida acima da linha de código referente, nos comentários em C (Símbolo '//'). Inicialmente a biblioteca *EmonLib.h* é incluída no código fonte. Em seguida é criada uma instância do monitor de energia chamada *emon1*, responsável pelos atributos do objeto. Também são declaradas variáveis relacionadas ao funcionamento do código: A variável booleana *estado* contém o estado atual da carga, podendo variar entre 1 (ligada) e 0 (desligada). A variável *rele* recebe o valor da porta analógica 4 onde o mesmo é conectado. A variável *controleLeitura* do tipo *char* recebe o valor lido pela leitura da serial. A variável *valores* recebe a interligação de todos os valores de leitura e separadores, assim como a variável *separador* recebe o símbolo '*@*'.

A segunda parte do código é destinada a função *setup*, estrutura básica na programação do *Arduino*. A serial é iniciada com uma taxa de 115200 *bauds*. O valor de símbolos por segundo enviados e recebidos por leitores de consumo e sistema principal deve ser o mesmo para que a comunicação seja realizada. Ainda na mesma estrutura, a função *emon1.voltage ()* é responsável por definir os parâmetros da leitura de tensão da instância *emon1*, (estes e outros parâmetros de calibração serão explanados no *BLOCO 3)*. Similarmente, a função *emon1.current ()* recebe os parâmetros de medição de corrente da mesma instância.

A terceira parte do código é dedicada à função *loop*, também fundamental e responsável

por criar uma recursão permanente. Dentro da estrutura de repetição, a função *controle()* é chamada, sendo esta responsável por todo o processo de recepção dos comandos de controle advindos da plataforma *Raspberry*. Na própria função *controle()* o primeiro comando realizado é a verificação de conexão serial, seguida da leitura do byte mais recente apontado no buffer de entrada da serial pelo comando *Serial.read()*. Com isto, o *Arduino* permanece em espera, lendo a serial enquanto nenhum comando é enviado. Caso o comando alocado na variável de controle seja um sinal alto, o relé conectado à porta analógica 4 é acionado, permitindo que a corrente CA percorra o contato do mesmo e acione a carga nele conectada. Com um sinal baixo de controle recebido pela serial o oposto ocorre, o contato é interrompido pelo relé e a corrente elétrica cessa seu percurso. Deste modo, o controle *on-off* descrito no Capítulo 2 é implementado no projeto através de sinais digitais de acionamento do relé. Caso a leitura da serial represente o caractere 'l', a função leitura é chamada, dando continuidade ao processo.

A última parte do código representa a sequência de comandos responsáveis pelo monitoramento dos valores. Tal atribuição fica a cargo da função *emon1.calcVI ()*, responsável por realizar os cálculos de tensão e corrente RMS da instancia *emon1* e, em seguida, potência ativa, potência aparente e fator de potência. Os valores de cada cálculo são alocados nas variáveis *tensao, corrente*, ativa, *aparente* e *fator*, respectivamente. Para facilitar o envio dos valores pela serial, os mesmos são alocados em uma *string* e concatenados a separadores definidos como símbolos *'@'*. Esta *string* é enviada pela serial em um único comando Serial.println(), responsável por escrever o texto em formato ASCII (8 bits por caractere) além de enviar uma quebra de linha ao final da mensagem.

Os cálculos dos valores realizados pela biblioteca são apresentados a seguir, baseados em carregamentos não-lineares (HUDSON e LEA, 2015).

Cálculo da potência ativa: Os valores instantâneos de corrente e tensão medidos são multiplicados entre si, obtendo-se o valor de potência instantânea. Este é somado um número de vezes igual ao número de amostras interno da biblioteca e esta soma total, dividida por esta quantidade de amostras, resultando na média dos valores de amostras. Esta lógica é associada a equação de potência media (*Equação 14*) apresentada no capitulo 2.

```
for (n=0; n<number_of_samples; n++)
{
  inst_power = inst_voltage * inst_current;

  sum_inst_power += inst_power;
}

real_power = sum_inst_power / number_of_samples;
```

Cálculo da tensão RMS: Os valores instantâneos de tensão são multiplicados entre si um número de vezes igual ao número de amostras interno da biblioteca e o resultado destes são adicionados a um somatório. A média dos valores quadrados é obtida pela razão entre o somatório e o números de amostras, com a raiz deles sendo aplicada nesta média. Esta lógica é associada a equação de tensão RMS (*Equação 7*) apresentada no capitulo 2.

```
for (n=0; n<number_of_samples; n++)
{
  squared_voltage = inst_voltage * inst_voltage;

  sum_squared_voltage += squared_voltage;
}

mean_square_voltage = sum_squared_voltage / number_of_samples;
root_mean_square_voltage = sqrt(mean_square_voltage);
```

Cálculo da corrente RMS: Similarmente como é feito na tensão RMS, a raiz da média dos valores quadrados da corrente é obtida, através dos valores instantâneos de corrente. Esta lógica é associada a equação de tensão RMS (*Equação 7*) apresentada no capitulo 2, com a substituição da tensão pela corrente.

```
for (n=0; n<number_of_samples; n++)
{
  squared_current = inst_current * inst_current;

  sum_squared_current += squared_current;
}

mean_square_current = sum_squared_current / number_of_samples;
root_mean_square_current = sqrt(mean_square_current);
```

Cálculo da potência aparente: O produto entre tensão RMS e corrente RMS resulta no valor de potência aparente. Esta lógica é associada a equação de fator de potência (*Equação 16*) apresentada no capitulo 2, isolando-se *S*.

```
apparent_power = root_mean_square_voltage * root_mean_square_current;
```

Cálculo do fator de potência: A razão entre potência ativa e aparente resulta no fator de potência. Esta lógica é associada a equação de fator de potência (*Equação 16*) apresentada no capitulo 2.

```
power_factor = real_power / apparent_power;
```

O monitor serial é uma ferramenta da IDE do *Arduino* responsável por realizar a comunicação entre plataforma e computador através de uma porta serial UART *(Universal Asynchronous Receiver/Transmitter*) da plataforma. Esta ferramenta possibilita ao usuário a visualização de dados desta comunicação, utilizando comandos de leitura (*Serial.read*) e escrita (*Serial.print*) na serial, em uma taxa de *bauds* pré-definida (ARDUINO, 2015). O monitor serial é utilizado no projeto para apresentar visualmente os dados sendo enviados e recebidos na comunicação serial.

Os valores decorrentes dos cálculos descritos são apresentados no monitor serial do *Arduino*, na forma como são recebidos pelo sistema principal. A *Figura 33* apresenta o funcionamento do módulo de consumo de energia 1 (Porta COM18) com uma carga de um ventilador portátil de 15W (Potência nominal), 60Hz e 220V da marca Fame. A *Figura 34* apresenta o funcionamento do leitor de consumo de energia 2 (Porta COM34) com uma carga de uma lâmpada incandescente de 60W (Potência nominal), 220-240V da marca Osram. Em ambos são enviados três sinais de leitura (caractere ‘*l*’) com a carga desligada, um sinal de acionamento de controle (número *1*), três sinais de leitura com a carga ligada, um sinal de desligamento de controle (número *0*), três sinais de leitura com a carga desligada, e um acionamento e

desligamento sucessivos alternados por leituras, feitas pela entrada de dados do monitor serial.

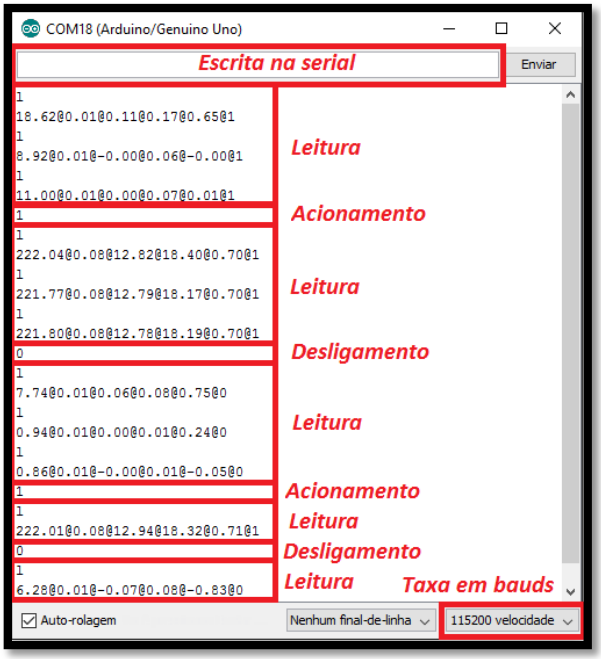

Figura 33 – Monitor serial do Leitor 1

Fonte: Elaborada pelo autor

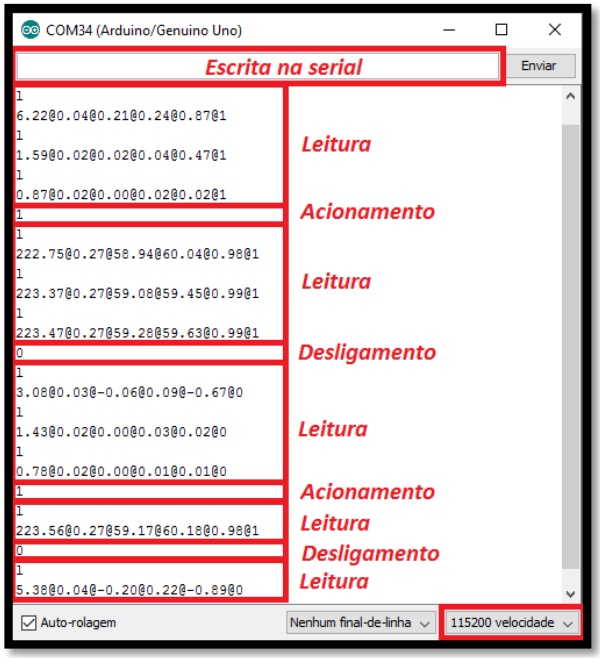

Figura 34 – Monitor serial do Leitor 2

Fonte: Elaborada pelo autor

3.3. Sistema Principal (BLOCO2)

O sistema principal é composto de plataforma *Raspberry* e periféricos conectados a ele, além de sensores de captação dos dados (incluindo os leitores de consumo). Os sinais dos sensores são recebidos pela plataforma, onde são realizados os cálculos e operações necessários para realizar o controle dos equipamentos. Os sinais de controle são enviados aos leitores de consumo, responsáveis por ativar ou desativar os equipamentos. A estrutura da interface gráfica e seus gráficos de consumo também são desenvolvidos nesta seção.

3.3.1. Montagem do protótipo

A plataforma *Raspberry* utilizada no projeto é conectada a um monitor LCD (*Liquid Crystal Display*) através de um adaptador VGA/HDMI e a um mouse e teclado sem fio através de um receptor USB. Um receptor WI-FI é utilizado para acessar a rede local e realizar atualizações e instalações de aplicativos pelo terminal. Uma fonte de alimentação de 110-240V/ 5V e 2,5A conecta o *Raspberry* a rede elétrica. A comunicação entre leitores de consumo de energia (*Arduino*) e sistema principal é realizada através de uma conexão serial USB. A *Figura 35*

apresenta as ligações dos periféricos na plataforma *Raspberry*.

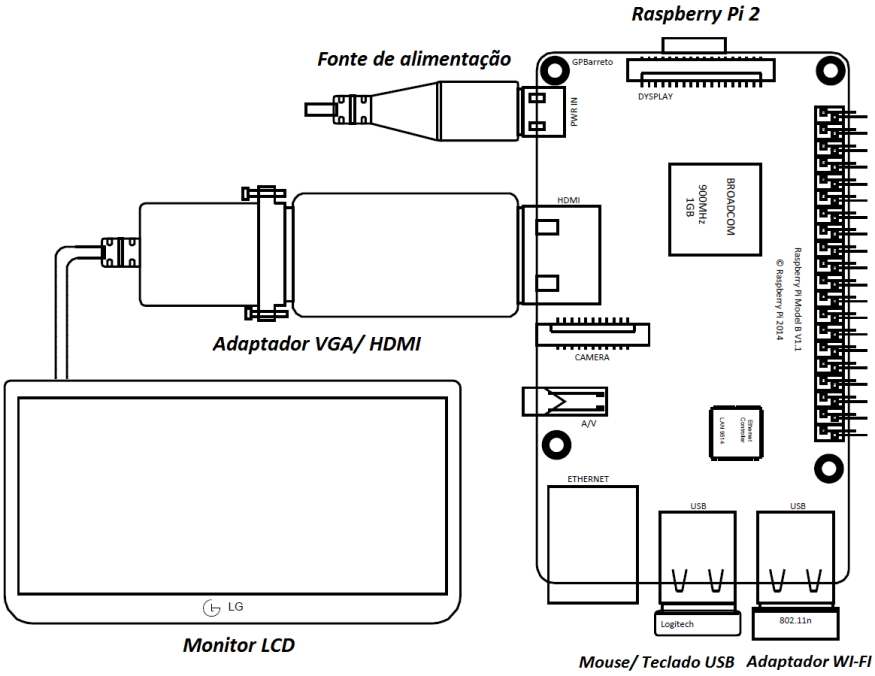

Figura 35 – Conexões do Raspberry e periféricos

Fonte: Elaborada pelo autor

O sensor HC-SR501 é conectado aos pinos GPIO07, 5V e GND, assim como um Buzzer (dispositivo sinalizador de áudio) de 5V é conectado ao pino GPIO12 e ao GND, utilizado para reproduzir sons de notificação ao usuário. Para a ligação do sensor LM35 ao *Raspberry*, o conversor PCF8591 é conectado ao mesmo. Os pinos terra (V_{SS} e A_{GND}) do conversor e GND do LM35 são aterrados em uma porta GND da plataforma, assim como os pinos não utilizados (AIN1,AIN2,AIN3,A0,A1,A2 e EXT). Os pinos de alimentação (V_{DD}) do conversor e do sensor são conectados ao 5V da plataforma. O pino de saída de dados do LM35 é conectado à entrada analógica AIN0, única utilizada do conversor. As portas I^2C do PCF8591 são conectadas diretamente as suas respetivas portas no *Raspberry*. Como o sensor LM35 varia a tensão de saída numa faixa entre 0 e 1V, é necessário inserir 1V na tensão de referência do conversor para que a proporção da resolução de 8 *bits* seja mantida. Para isso o pino V_{REF} é conectado ao divisor de tensão, entre 5V e GND com um resistor de 2KΩ e um de 500Ω, apresentado abaixo:

$$\boldsymbol{V_{REF} = \frac{R_2}{R_1+R_2}\, x\, V_{entrada} = \frac{500K}{500K+2K}\, x\, 5V = 1V} \qquad \textit{(31)}$$

Desta forma, a variação de tensão de 0 a 1V na entrada do AIN0 será proporcional a

conversão realizada pelo PCF8591. A mesma tem uma resolução de 8 *bits* (256 valores possíveis), possuindo a equivalência em volts de sua variação no pino de referência abaixo:

$$\boldsymbol{V_{REF} = \frac{1V}{255\ bits} = 0,0039215V = 3,9215mV\ por\ bit} \quad \textit{(32)}$$

Para a leitura de temperatura, cada 10mV equivalem a 1°C. Logo, a equivalência em bits da mudança de um grau no sensor de temperatura é apresentada abaixo:

$$\boldsymbol{Bits\ por\ 1°C = \frac{10mV}{3,3215mV} = 2,5500\ bits} \quad \textit{(33)}$$

A *Figura 36* apresenta as ligações dos sensores e conversor A/D ao *Raspberry*. A figura x apresenta o protótipo final do sistema principal.

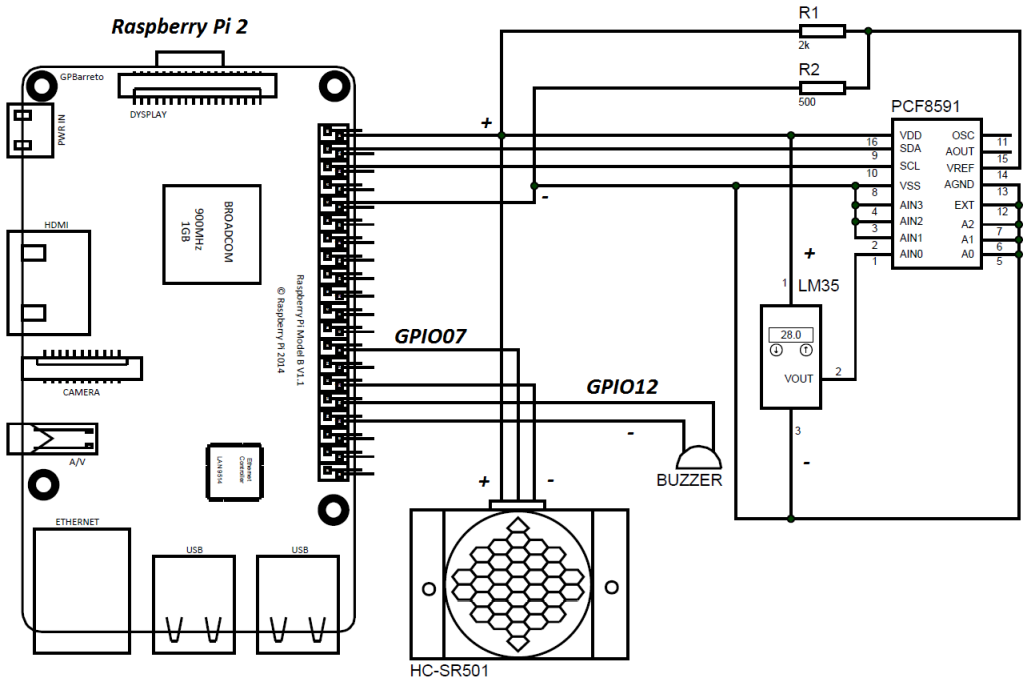

Figura 36 – Conexões do Raspberry, sensores e conversor A/D

Fonte: Elaborada pelo autor

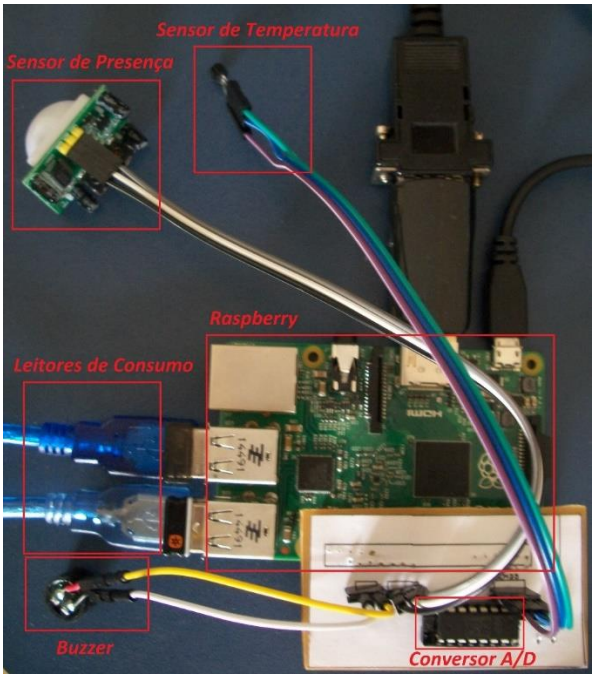

Figura 37 – Protótipo final do sistema principal

Fonte: Elaborada pelo autor

3.3.2. Código fonte

O código em *Python*, apresentado no APÊNDICE B, é desenvolvido na IDE *Geany* pelo sistema operacional do *Raspberry*. O programa é responsável por receber os valores de monitoramento pela interface serial, gerar os sinais de controle de volta para o *Arduino* e apresentar estes valores através de gráficos pela GUI. O resultado é uma interface gráfica e um Terminal auxiliar. O segundo é responsável por apresentar notificações importantes na execução do processo. A primeira parte do código é dedicada a adição dos módulos a serem utilizados. São eles: Módulo GTK (*gtk*), módulo serial, módulos de manipulação de tempo (*datetime* e *time*), módulo de criação de subprocessos, módulo adicional do GTK (*gobject*), módulo de manipulação dos pinos GPIO (*RPI.GPIO*) e módulo da comunicação I^2C. Ainda tratando de definições, são especificados: O modo de numeração dos pinos da plataforma pela numeração da *Broadcom* (BCM), desativação de mensagens de aviso sobre GPIOs, pinos dos sensores e *Buzzer*, comportamento destes pinos (E/S) e utilização da porta I^2C (endereço e registrador). São criadas instâncias do módulo serial para cada leitor de consumo conectado ao *Raspberry* (endereço e taxa da comunicação como argumentos).

A seção seguinte do código contém as duas classes envolvidas no funcionamento do sistema principal. A classe GUI representa a interface gráfica do mesmo e realiza todas as funções relacionadas a leitura, controle e atualização dos leitores de consumo. Esta classe herda as

características do objeto *window*, estrutura fundamental GTK para criação de janelas. O primeiro método da classe (*__init__*) é responsável por inicializar parâmetros globais a serem usados nos outros métodos. Dentre estes parâmetros, é criada a janela principal da interface e todas as suas características individuais, como tamanho e posicionamento. Um objeto contendo uma imagem com o logo do UniCEUB também é criado, assim como o evento responsável por fechar a janela ao se pressionar fechar (X). Variáveis diversas utilizadas ao longo do código são criadas e atribuídas com os devidos valores iniciais. Os campos de entrada de dados, utilizados para receber valores inseridos pelo usuário, são criados e seus parâmetros definidos. As etiquetas (*labels*), responsáveis por apresentar um texto na interface gráfica, também são criadas e definidas, com suas características e valores iniciais. Os contêineres, responsáveis por agrupar conjuntos de *widgets* , são criados e definidos similarmente. Botões para interação com o usuário (*Liga/Desliga*, *Apaga dados* e *Sair*) são criados e conectados a eventos de pressionamento (*clicked*). Textos de ajuda (*tooltips*) são criados com devidos textos associados aos objetos na interface. Com todos os objetos previamente criados, os mesmos são alocados no contêiner tabela em posições especificas. As colunas um, dois e três são dedicadas ao leitor 1, assim como as colunas 4, 5 e 6 são dedicadas ao leitor 2 e as colunas 7, 8 e 9 são dedicadas ao leitor 3 (não utilizado no projeto). A *Tabela 2* apresenta a disposição dos *widgets* na mesma, com a cor azul para os mesmos, vermelha para texto de etiquetas e branco para células vazias.

Tabela 2 – Posicionamento dos widgets na tabela da interface

Fonte: Elaborada pelo autor

	titulo						data	
Leitor 1			Leitor 2			Leitor 3		
Tensao:	tensao1	V	Tensao:	tensao2	V	Tensao:	tensao3	V
Corrente:	corrente1	A	Corrente:	corrente2	A	Corrente:	corrente3	A
Pot. Ativa:	ativa1	W	Pot. Ativa:	ativa2	W	Pot. Ativa:	ativa3	W
Pot. Aparente:	aparente1	VA	Pot. Aparente:	aparente2	VA	Pot. Aparente:	aparente3	VA
Fator de pot.:	fator1		Fator de pot.:	fator2		Fator de pot.:	fator3	
Consumo Atual:	consumo1	Wh	Consumo Atual:	consumo2	Wh	Consumo Atual:	consumo3	Wh
Status:	estado1		Status:	estado2		Status:	estado3	
Temperatura:	tempLabel	°C						
Temp. Acionamento:	entraTemp	°C						
Notificacao:	entraConsumo1	Wh	Notificacao:	entraConsumo1	Wh	Notificacao:	entraConsumo1	Wh
grafico1			grafico2			grafico3		apaga
liga1			liga2			liga3		
logo	nome							sair

A tabela acima é inserida no widget contêiner *vbox*, pois apenas um objeto pode ser adicionado a janela principal. A chamada do método responsável por apresentar uma imagem de fundo na janela é conectada à *vbox* com um evento *expose* (expansão e redução da janela). Em seguida, a janela é apresentada e a função *timeout* utilizada. A mesma é responsável por chamar um método em um período específico recursivamente. Posteriormente ao método de inicialização, os outros métodos envolvidos são definidos.

O método *timeout* é chamado ao final da recursão da interface gráfica, de um em um segundo. Este método é responsável por manipular os sinais de controle enviados aos leitores de consumo e realizar a leitura dos valores dos mesmos. Inicialmente é chamado o método *verificaPIR*, incumbido de alterar o estado da variável *controlePIR* com a presença ou ausência de iluminação. Similarmente, o método seguinte (*verificaLM35*) altera o estado da variável *controleLM35* caso a temperatura mensurada seja maior ou igual a definida na variável *tempDefinida*, inserida pelo usuário. Assim como os estados dos sensores podem ser alterados, o pressionamento dos botões liga/desliga alteram o estado atual da carga. Após a verificação dos sensores, o método de atualização chama o método *controleCarga*, encarregado de enviar os sinais de controle caso algum dos estados (carga e sensores) tenham sido alterados. Dentro do mesmo, o número '*1*' (sinal de acionamento) é enviado pela serial aos leitores caso qualquer um dos seus respectivos estados tenham sido alterados para um, ou o número '*0*' (sinal de desligamento) para um estado de zero. Desse modo, apenas dois estados são possíveis no controle (ligado ou desligado), definidos pelos estados dos sensores. Cada acionamento emite um som de notificação específico. Esta manipulação de estados atuais e anteriores permite o acionamento manual e automático das cargas sem interferência entre eles.

De volta ao método *timeout*, uma solicitação de leitura é enviada (caractere '*l*') após a verificação dos métodos anteriores. O valor lido é tratado removendo-se os *line breakers* inseridos pela função *println* (no *Arduino*) e separando-se as medições individuais pelos separadores '@' com a função *split*. O resultado é uma lista de seis posições com cada valor individual alocado em uma posição especifica. Uma estrutura de repetição é responsável por realizar a leitura e o tratamento de valores dentro da lista, convertendo-os no formato *float* (e *int* para o estado). Com a função *try* e uma variável de controle é possível direcionar a execução dos comandos para a função *exception* caso algum erro ocorra dentro da recursão, reiniciando a leitura. Limites inferiores de variação dos valores, com as cargas desligadas, são definidos. Para uma variação na tensão de até 3V, o valor 0 é atribuído. Assim como uma variação de até 0,04 da corrente, 0,01 da potência ativa, 0,10 da potência aparente e 0,10 do fator de potência. Estes limites são definidos como uma correção à imprecisão do conversor A/D de 10 bits do *Arduino* na presença de valores

muito pequenos (HUDSON e LEA, 2015).

Ainda dentro do método *timeout*, a parte responsável por atualizar a interface gráfica, com os valores lidos anteriormente, é executada. O valor atual de data é armazenado, definido no padrão brasileiro (dia/mês/ano hora:minuto:segundo) e atualizado na sua etiqueta correspondente. Para cada leitor de consumo definido pela variável *controleLeitor* é realizado o cálculo de consumo atual, temperatura (apenas em um leitor) e atualização dos valores em suas etiquetas correspondentes, assim como o armazenamento no arquivo leitor1.dat e leitor.dat das leituras realizadas. O cálculo de consumo de energia é apresentado a seguir, baseado na *Equação 21* do *capitulo 2*. Um somatório das leituras de potência ativa é dividido pelo período entre leituras (dois segundos para cada leitor, com dois leitores).

$$\boldsymbol{Consumo\ (Wh) = \frac{Somatório\ de\ potencia\ ativa\ (W)}{30x60\ (segundos)}} \qquad \textit{(34)}$$

Caso este consumo seja igual ou superior ao limite de consumo definido pelo usuário, uma notificação sonora é emitida até este limite ser alterado ou o sistema desligado. A leitura da temperatura é realizada com um comando de leitura da porta I^2C, dividindo o valor pela escala do conversor A/D (2,55). Semelhante ao consumo de energia, uma notificação sonora é emitida caso a mesma seja maior ou igual à definida pelo usuário. O armazenamento dos valores nos arquivos leitor1.dat e leitor.dat é realizada em linhas (em modo *append*), com uma coluna para cada sequência de valores individuais. Esta disposição de colunas é necessária para a correta criação dos gráficos pela ferramenta *gnuplot*. Um exemplo do arquivo leitor1.dat é apresentado na *Figura 38*. A primeira coluna é constituída da data das leituras, segunda coluna com a hora, minuto e segundo das mesmas. Terceira, quarta, quinta, sexta, sétima, oitava e nona são respectivamente: tensão, corrente, potência ativa, potência aparente, fator de potência, estado e consumo de energia. Uma coluna adicional é construída no arquivo *leitura2.dat*, contendo as leituras de temperatura.

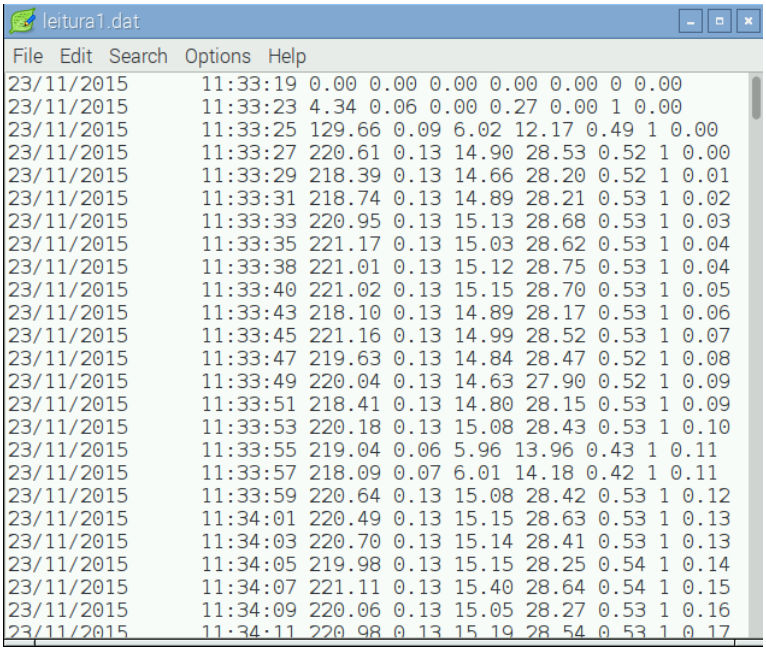

Figura 38 – Exemplo do arquivo leitura1.dat

Fonte: Elaborada pelo autor

Após o método de atualização, outros métodos auxiliares são definidos. O *imagemFundo*, como descrito anteriormente, define uma imagem como plano de fundo. O método *sairConfirma* cria uma mensagem de diálogo ao pressionar-se o botão '*Sair*', com os botões YES e NO. Caso seja pressionado o YES, uma mensagem é apresentada no Terminal, as portas GPIO são limpas, sinais de desligamento são enviados aos leitores de consumo, as conexões com as seriais são interrompidas e a interface gráfica fechada. Caso seja pressionado NO, a mensagem de diálogo é fechada. A mensagem de diálogo é apresentada na *Figura 39.*

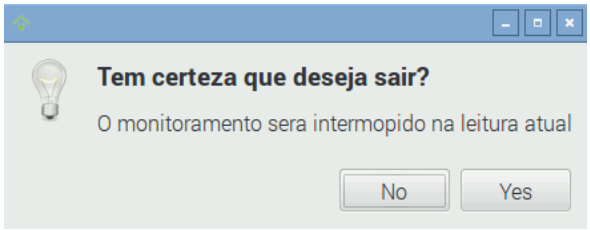

Figura 39 – Exemplo da mensagem de dialogo

Fonte: Elaborada pelo autor

O método *defineTemp* atribui o valor lido da entrada de dados ao limite de temperatura *tempDefinida*. Ambos defineConsumo1 e defineConsumo2 atribuem seus valores lidos pela entrada de dados aos seus respectivos valores de consumo definidos. Os métodos *gnuplot1* e *gnuplot2* executam um subprocesso responsável por enviar um comando ao Terminal (*shell*) do SO. Este comando executa o utilitário *gnuplot* por um *script* previamente definido (APÊNDICE C). Neste *script*, as especificações de criação da janela com o gráfico correspondente são

definidas, como a cor das linhas e posicionamento. Um exemplo do gráfico gerado ao pressionar-se o botão '*Gerar grafico*' é apresentado na *Figura 40*. O gráfico das potencias ativa e aparente, assim como o consumo atual é apresentado pelas linhas vermelha, verde e azul respectivamente. A divisão do eixo *x* é dada no formato brasileiro de datas.

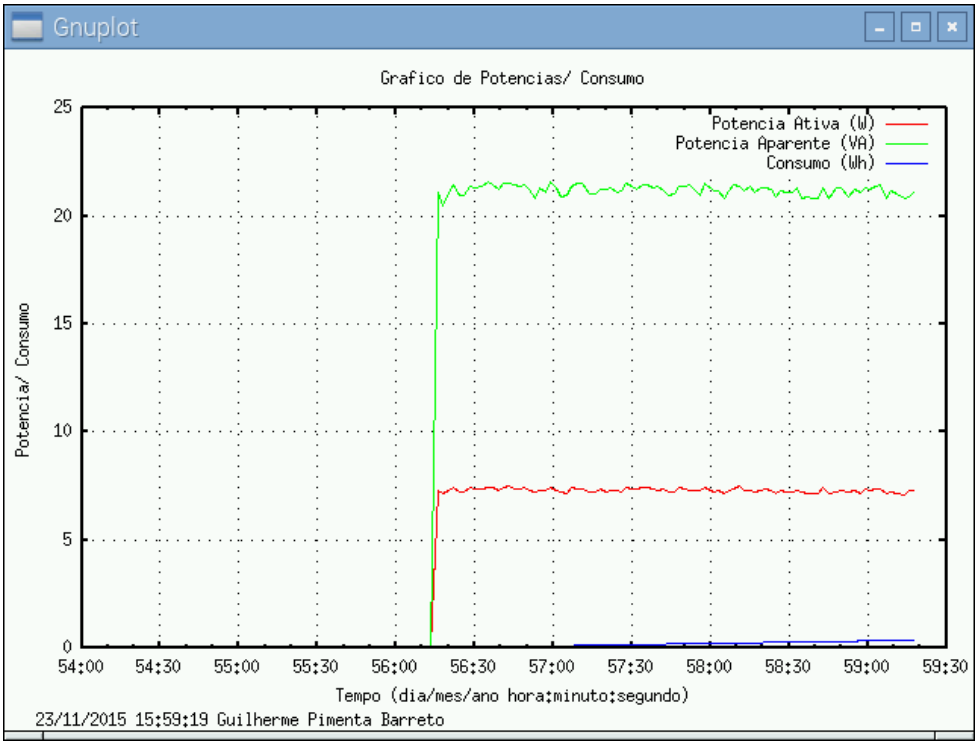

Figura 40 – Exemplo do gráfico gerado pelo programa

Fonte: Elaborada pelo autor

A segunda classe envolvida é responsável por reproduzir os sons de notificação presentes no projeto (PRATYAKSA, 2015). No método inicial são definidas as notas musicais básicas (dó, ré, mi, fá, sol, lá, si, Dó) com suas frequências correspondentes. O método *buzz* define os parâmetros de período, intervalo e ciclos, além de executar os tons com as durações relativas. O método *toca* define uma melodia para cada valor de tom, com suas notas e durações equivalentes. Para acionamentos o tom corresponde ao número 3, para apagar os dados o tom corresponde ao número 2 e para notificação de consumo superior ao limite o tom corresponde ao 4. A inicialização do sistema corresponde ao tom de número 1, melodia baseada em um pequeno trecho da conhecida 9ª sinfonia de Bethoven (*Ode to Joy*).

Na estrutura *main* do código fonte, um texto é apresentado no Terminal, o tom 1 é executado (após uma instância da classe Buzzer ser criada) e uma instância da classe GUI é criada. Ao final, o comando *gtk.main* é responsável por apresentar a interface gráfica como uma recursão, tratando cada interação com uma resposta definida. Com o sistema funcionando e conectado a dois leitores de consumo, um exemplo da janela é apresentado na *Figura 41*. Um exemplo do

Terminal funcionando junto à janela é apresentado na *Figura 42*. Os dados apresentados no Terminal também são armazenados no arquivo *.dat.*

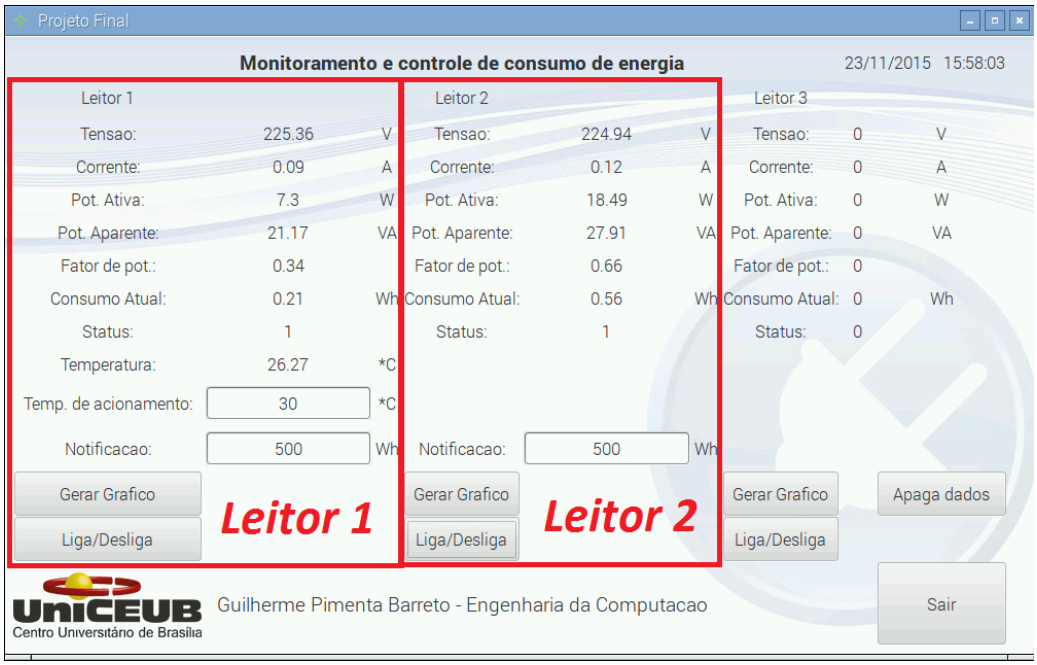

Figura 41 – Exemplo da janela do programa

Fonte: Elaborada pelo autor

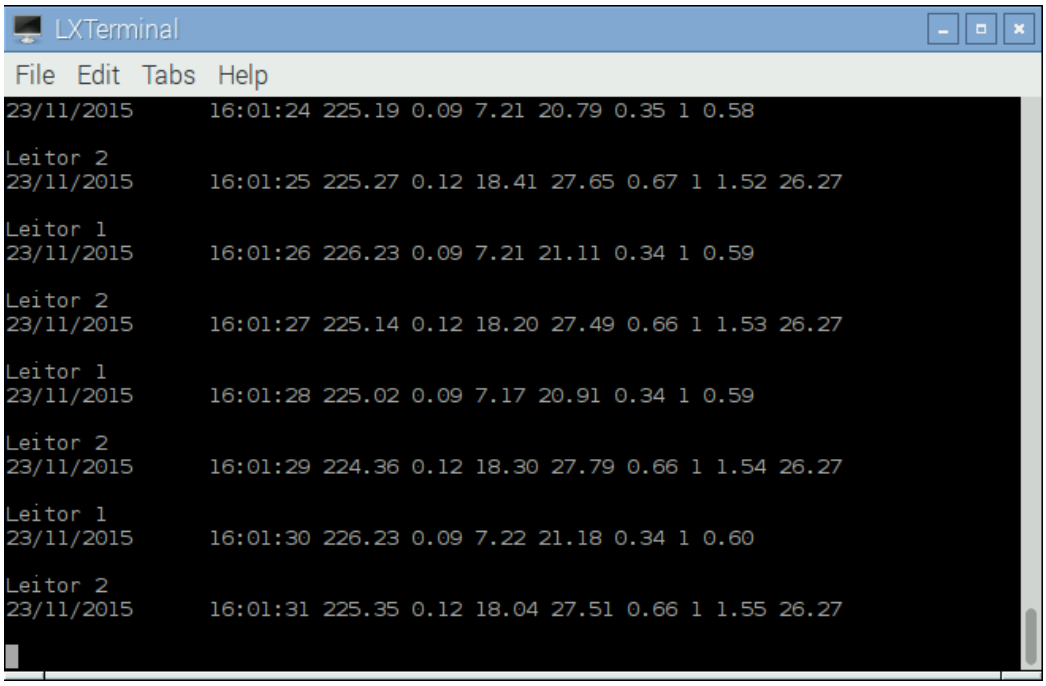

Figura 42 – Exemplo do Terminal do programa

Fonte: Elaborada pelo autor

3.4. Interação e calibração (BLOCO 3)

Para a comunicação serial entre *Arduino* e *Raspberry* é utilizado o próprio cabo USB A-B que acompanha o primeiro. Ambos possuem uma interface USB 2.0 (quatro do tipo A no *Raspberry* e uma tipo B no *Arduino*). Para o *Arduino*, esta interface é conectada ao ATmega16U2, programado como um conversor USB-Serial e utilizada para carregar os códigos no ATmega328, além de possuir um circuito de proteção de até 500mA (ARDUINO, 2015). No *Raspberry*, esta interface é principalmente utilizada para conexão de periféricos USB, sendo conectadas ao chip LAN9514, que por sua vez é conectado a *Broadcom* BCM2836. Em sua concepção, a *Broadcom* da plataforma *Raspberry* entraria no mercado mobile. Por consequência, sua porta USB é uma *On-The-Go* (OTG) *host*, *hardware* mais simples que sua equivalente em um computador convencional, com uma limitação de 100mA para cada porta. Consequentemente, o sistema operacional realiza o trabalho de adequação dos dispositivos conectados na USB (RASPBERRY, 2015).

A *Figura 43* apresenta a conexão entre *Arduino* e *Raspberry*, uma interface 2.0 de tipo B no primeiro e uma de tipo A no segundo.

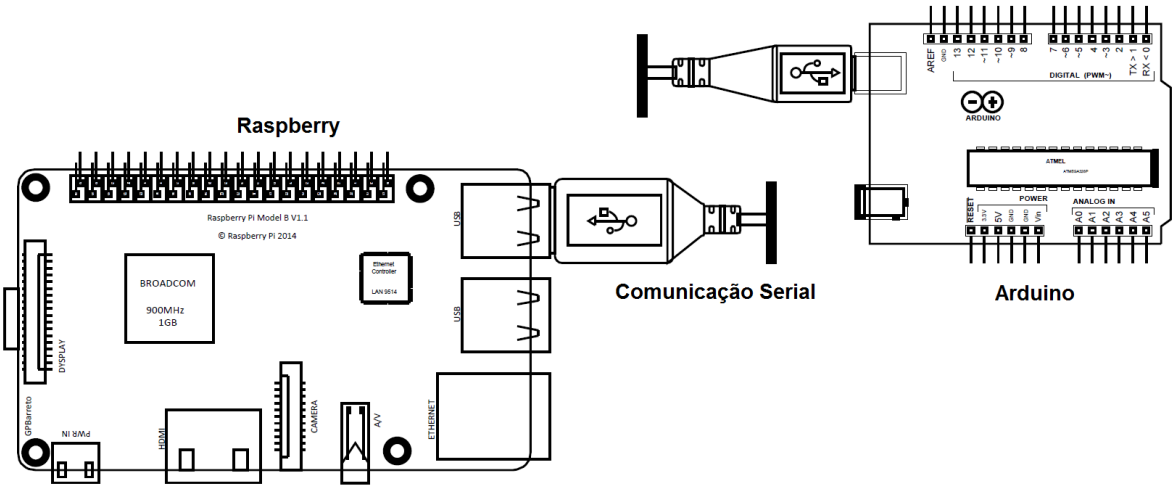

Figura 43 – Comunicação serial entre Raspberry e Arduino

Fonte: Elaborada pelo autor

3.4.1. Calibração dos leitores de consumo

O módulo leitor de consumo, constituído de *Arduino* e sensores de tensão e corrente, é montado como descrito no *capítulo 3*. A calibração de um equipamento representa a correção de suas diferenças de fábrica. Se realizada regularmente, também pode corrigir as diferenças físicas

que ocorrem ao longo do tempo. Os comandos responsáveis por definir os parâmetros das medições de tensão e corrente devem ser ajustados aos componentes utilizados e as características da rede elétrica (HUDSON e LEA, 2015). Para esta etapa o analisador de consumo de energia elétrica DDS238-2 (65A/220V), da marca HIKING é utilizado como referência na calibração. Este medidor de energia monofásico é classificado com uma precisão de classe 1 pelo padrão IEC61036 e pode medir potência ativa, frequência da rede, tensão, corrente, dentre outros parâmetros.

A *Tabela 3* apresenta os comandos, parâmetros (ordem que aparecem no código), suas funções e a porcentagem de erro atingida em cada comando (após a calibração).

Tabela 3 – Valores de calibração para uma lâmpada incandescente de 60W/ mini ventilador de 15W

Fonte: Elaborada pelo autor

Comando	**Parâmetros**	**Função**	**Erro (%)**
emon1.voltage	*input pin* (2) *calibration* (269/280) *phase_shift* (-0.1/ -1.1)	Define o pino do sensor de tensão Coeficiente de calibração de tensão Calibração de ângulo de fase	0,3%
emon1.current	*input pin* (1) *calibration* (4.5/1.8)	Define o pino do sensor de tensão Coeficiente de calibração de corrente	0%
emon1.calcVI	*crossings* (60) *time-out* (2000)	Amostragem de *zero crossing* Tempo de amostragem (milisegundos)	1,2%

Inicialmente o parâmetro *calibration* é ajustado com o auxílio do analisador de consumo, utilizando a carga puramente resistiva de uma lâmpada incandescente (60W/ 220V). O coeficiente de calibração de tensão é ajustado até os valores de tensão mensurados se aproximarem ao máximo dos valores correspondentes no analisador, ligado em paralelo à carga. A realização desta etapa obteve um valor de *269* para este coeficiente. Similarmente, a corrente é calibrada, obtendo-se um valor de 4.5. Para o ângulo de fase, mesmo usando uma carga resistiva, as medições de tensão e corrente não são simultâneas no microcontrolador, ocasionando em uma defasagem que deve ser calibrada com o parâmetro phase_shift, obtendo-se *-0.1*. Este não é ajustado com um equipamento auxiliar e se traduz na comparação entre os valores de potência ativa e aparente. Quanto mais próximos estes valores se encontrarem, resultando em um fator de potência mais próximo possível de *1,00* (para a carga puramente resistiva), mais precisas serão as medições. Para a medição dos valores das potências é definido o parâmetro crossings e o time-out. O

primeiro representa o número de vezes que o sinal de tensão atinge o valor de zero (zero cross) e o segundo, o tempo necessário para realizar este processo. Estes valores correspondem a uma taxa de amostragem e variam com a frequência da rede elétrica. Assim, quanto mais crossings (e consequentemente um maior time-out), mais consistente serão os resultados e maior será o processamento realizado pelo microcontrolador. Com a calibração dos mesmos, obteve-se um valor de *60 crossings*, com *2000* de *time-out*, sem perda significativa de processamento.

Para o mini ventilador os mesmos processos foram realizados, porém com valores de calibração diferentes. Para a tensão obtém-se o valor 280, para a corrente o valor de 1,8. A defasagem é de -0.99, com um time-out de 60 e crossing de 2000. A tabela com os valores de calibração com o ventilador é apresentada na *Tabela 4*.

Tabela 4 – Valores de calibração para um mini ventilador de 60W

Fonte: Elaborada pelo autor

Comando	**Parâmetros**	**Função**	**Erro (%)**
emon1.voltage	*input pin* (2) *calibration* (280) *phase_shift* (-0.99)	Define o pino do sensor de tensão Coeficiente de calibração de tensão Calibração de ângulo de fase	0,11%
emon1.current	*input pin* (1) *calibration* (1.8)	Define o pino do sensor de tensão Coeficiente de calibração de corrente	0,11%
emon1.calcVI	*crossings* (60) *time-out* (2000)	Amostragem de *zero crossing* Tempo de amostragem (milisegundos)	0,58%

3.4.2. Interação entre leitores de consumo e sistema principal

Após a criação do código fonte e da montagem da parte física, o funcionamento normal do sistema de monitoramento e controle pode ser resumido a seguir. A sequência de chamadas entre leitores de consumo, sistema principal e interface gráfica é apresentada no diagrama UML de sequência da *Figura 44*. Os leitores de consumo são ligados e suas comunicações seriais iniciadas na taxa 115200 *bauds.* Da mesma forma, o sistema principal é ligado e inicia as conexões seriais correspondentes em 115200 *bauds*. Os leitores aguardam solicitações de leitura, lendo constantemente a serial. A interface GUI é criada (com a estrutura *super*) e apresentada ao usuário (*show_all*). Uma solicitação de leitura é enviada ao leitor de consumo com o comado *write("l").*

A resposta é enviada pelo *Arduino* na serial com o comando *println(valores).* A leitura da resposta é realizada pelo comando *readline()* no *Raspberry*. Recebidos os valores da medição, estes são atualizados nas etiquetas da interface gráfica pelo comando *set_text(valores).* Caso o botão 'Liga/Desliga' na GUI seja pressionado, o estado atual da carga é invertido com o comando *not controleCarga,* ocasionando no envio de um sinal de controle (1 ou 0) ao leitor de consumo e no acionamento ou desligamento da carga conectada a ele. Com a alteração de estado dos sensores (PIR e LM35) sinais de controle também são enviados ao *Arduino*. Com o pressionamento do botão '*Sair*', o método *sairConfirma* é chamado, resultando no envio de sinais de desligamento aos leitores de consumo, e fechamento da interface gráfica.

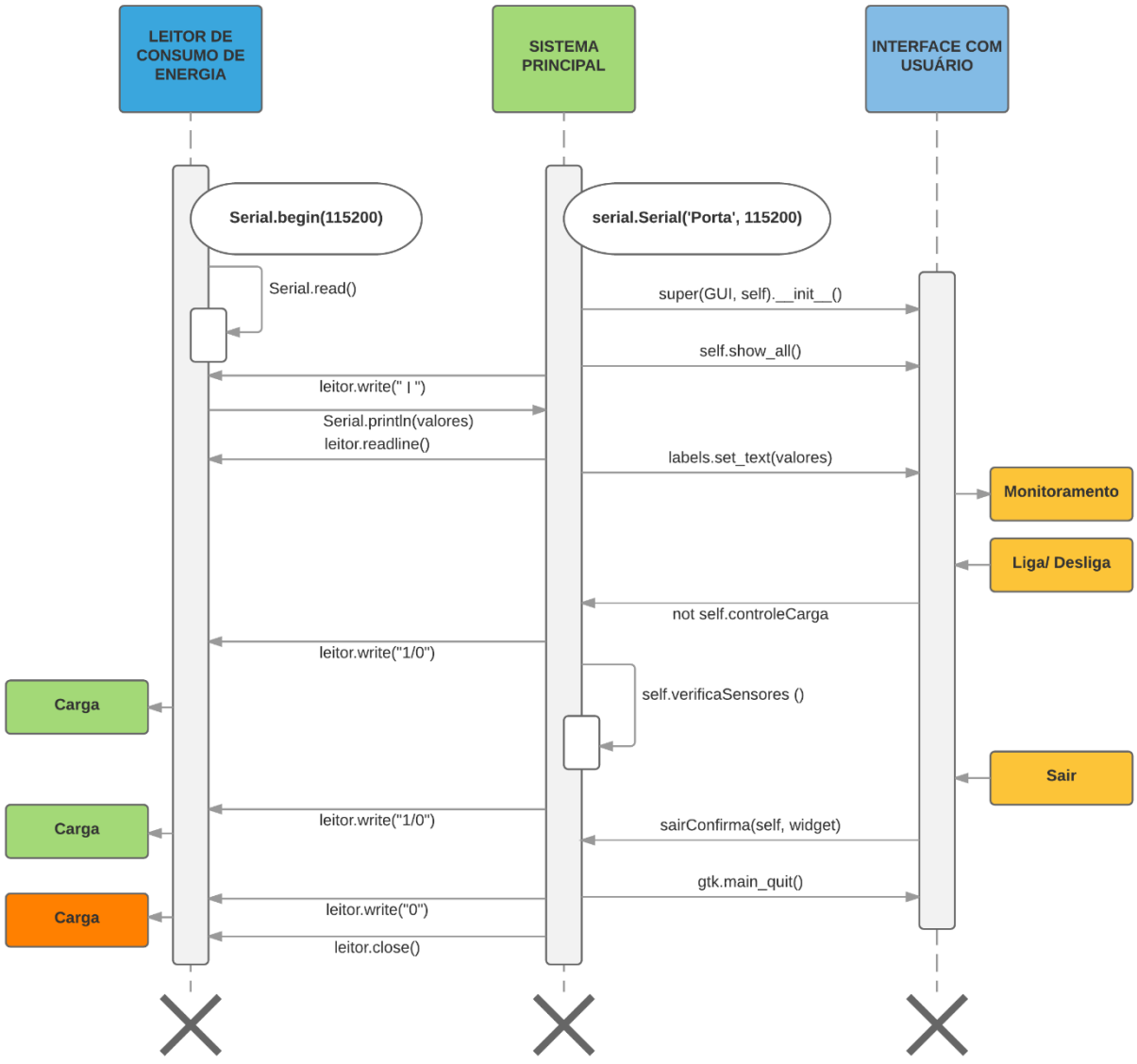

Figura 44 – Diagrama UML de sequência

Fonte: Elaborada pelo autor, utilizando ferramenta Lucidchart: www.lucidchart.com

CAPÍTULO 4 - TESTES E RESULTADOS

Neste capítulo serão apresentados os resultados dos testes realizados no projeto. Os três blocos definidos no capítulo 3 são avaliados de formas diferentes, respeitando suas funcionalidades e características. O primeiro bloco, referente ao leitor de consumo de energia, é testado com diferentes cargas. Seus sinais de tensão são analisados, comparando-se com os sinais de tensão esperados. Estes testes têm o objetivo de estimar a performance do leitor de consumo, verificando se os requisitos propostos foram atingidos. Para o sistema principal, segundo bloco, é testado seu comportamento em pleno funcionamento, assim como a realização de suas funções individualmente. A parte referente à calibração no bloco 3 é analisada na calibração do leitor de consumo, seguindo as especificações da biblioteca.

Uma característica importante no projeto é o baixo consumo do sistema geral, implementada através de componentes econômicos e técnicas de eficiência energética. Para os BLOCOS 1 e 2 é medido o consumo referente aos mesmos. Com a plataforma *Raspberry* conectada ao analisador de energia DDS238-2, o sistema foi ligado e seus valores analisados. A *Figura 45* apresenta os níveis atingidos dos mesmos. Uma corrente de 0,03 alimenta o *Raspberry* com o sistema em funcionamento, correspondendo a um consumo de 4W. Tal consumo se aproxima ao de uma lâmpada de LED econômica e não representa grande impacto no consumo da residência. O fator de potência baixo representa as distorções e correntes reativas inerentes as cargas eletrônicas.

Figura 45 – Consumo de energia do sistema principal

Fonte: Elaborada pelo autor

Além do sistema principal, o consumo do monitor utilizado para visualização de informação junto ao Raspberry foi mensurado e seu valor de potência ativa atingiu 29W, porém o mesmo pode permanecer em modo de economia enquanto não é utilizado reduzindo o consumo a valores menores que a escala mínima que o analisador tem capacidade de medir. O mesmo acontece com os leitores de consumo, possuindo uma corrente inferior a 0,01A.

4.1. Testes do BLOCO 1

4.1.1. Medição dos sinais de tensão

Os sinais de saída do sensor de tensão foram analisados com o auxílio de um osciloscópio digital DSO-X 3024A, da marca Agilent. A *Figura 46* e a *Figura 47* apresentam, à direita, os sinais de saída do transformador e o sinal de entrada do *Arduino*. Do lado esquerdo, o sinal teórico esperado no início do projeto. No primeiro, a tensão de 220 é reduzida a 8,8V, enquanto no segundo o sinal é reduzido a 1,01V e deslocado em 2,5V para cima.

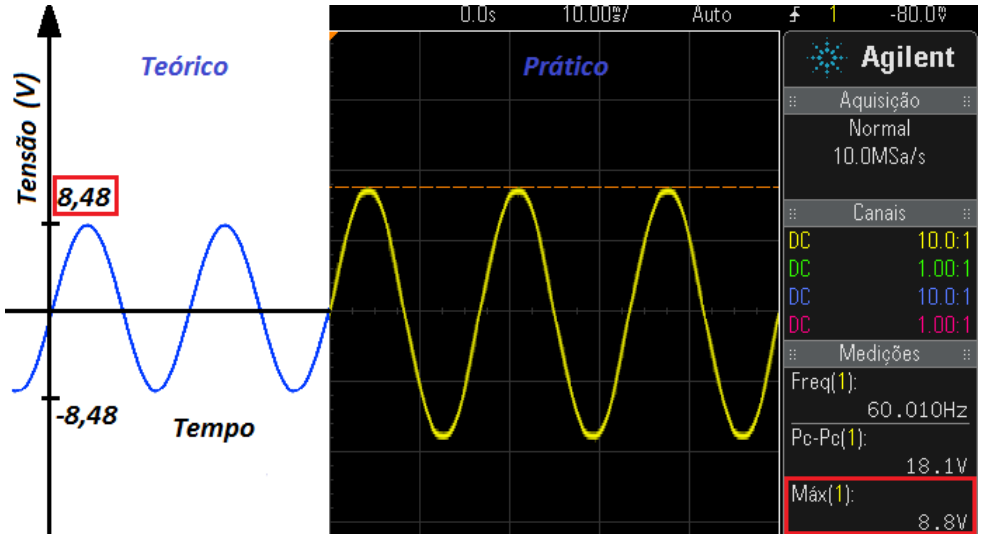

Figura 46 – Sinal de saída do transformador

Fonte: Elaborada pelo autor

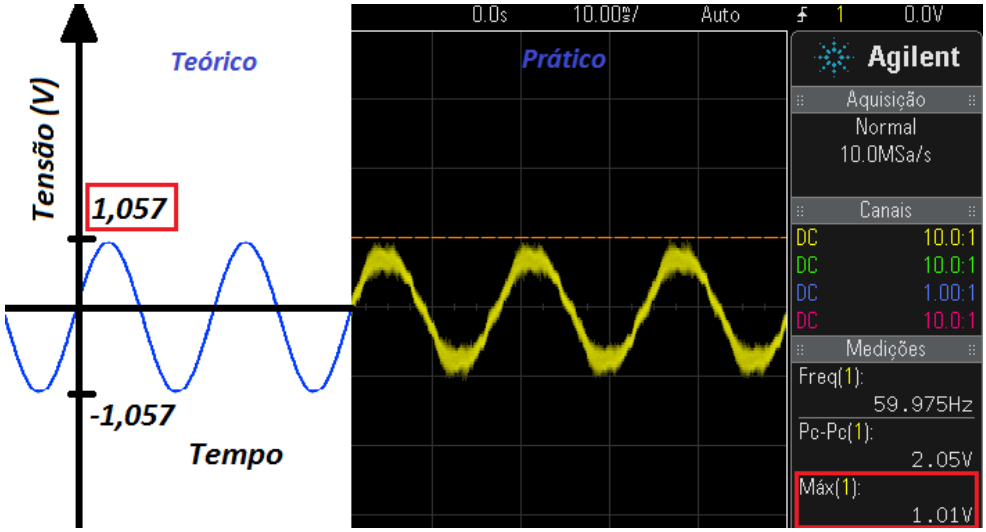

Figura 47 – Sinal de entrada do Arduino

Fonte: Elaborada pelo autor

Assim como apresentado no *capítulo 3 (Figura 29)*, o sinal teórico de saída do transformador é de 8,48V (Máximo), enquanto que o sinal analisado pelo osciloscópio apresenta um sinal de 8,8V. Esta diferença de 0,32V se dá por variações de fábrica e variação de carga nas espiras do primário e secundário do transformador, ocasionando em uma variação proporcional no sinal de entrada do *Arduino*. A análise do mesmo apresenta um sinal de 1,01V, em comparação ao valor teórico de 1,057V. Ambos os sinais oscilam na frequência da rede elétrica, de 60Hz.

4.1.2. Medição de equipamentos diversos

Para a demonstração do comportamento geral do leitor de consumo, foram utilizadas cargas de diferentes equipamentos. O leitor de consumo foi conectado em série com o analisador de consumo DDS238-2, por sua vez conectado à rede elétrica. O cabo USB padrão do *Arduino* conecta o mesmo ao computador, para que o monitor serial seja acessado (Porta COM34). A ligação do mesmo é apresentada na *Figura 48*.

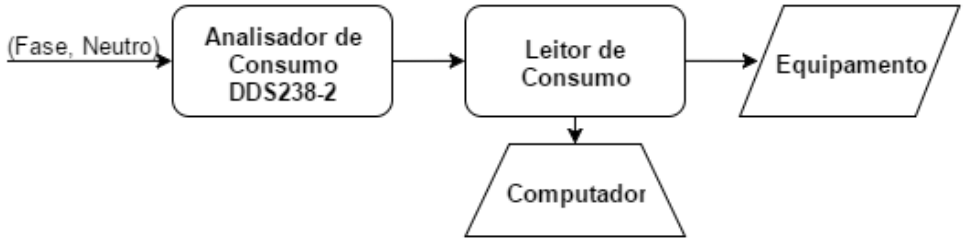

Figura 48 – Calibração do leitor com uma lâmpada incandescente de 60W

Fonte: Elaborada pelo autor, utilizando ferramenta Flow Chart Maker: www.draw.io

As especificações dos equipamentos são apresentadas na *Tabela 5*. A coluna *C* representa a corrente nominal classificada de fábrica, assim como a coluna *P* representa a potência nominal. A coluna *T* representa a tensão de operação dos equipamentos e *FP*, o fator de potência especificado.

Tabela 5 – Equipamentos utilizados para a medição

Fonte: Elaborada pelo autor

Equipamento	*C (A)*	*P (W)*	*T (V)*	*FP*
TV LCD 32'' LG	-	140	100-240	-
Mini ventilador FAME	-	15	220	-
Ventilador Mondial	-	50	220	-
Lâmpada Fluorescente FLC	0.124	15	220	0.55
Lâmpada Incandescente Osram	-	60	220	-

A *Tabela 6* apresenta os valores obtidos nos testes de medição (Medição do projeto/ medição do analisador de consumo). Para a medição de corrente, observa-se que quanto mais distante da carga de calibração mais divergentes são os valores. Este fato se torna claro na corrente da TV LCD, com um erro de aproximadamente 10%. O mesmo acontece com a potência ativa, com um erro de aproximadamente 8,3%. Tal comportamento limita a leitura de equipamentos sem a prévia calibração, porém o projeto em questão é baseado em um monitoramento de equipamentos fixos, não tendo grandes impactos de tal circunstância. As leituras de tensão não sofrem grande variação, refletindo em um erro reduzido. As cargas da lâmpada fluorescente e do mini ventilador são capacitiva e indutiva, respectivamente (refletindo o fator de potência de 0.63 e 0.59).

Tabela 6 – Medições em diversos equipamentos

Fonte: Elaborada pelo autor

Equipamento	*Tensão (V)*	*Corrente (A)*	*Potência ativa (W)*	*Fator de Potência*
TV LCD	221.62/ 222.8	0.46/ 0.51	99.90/ 109	0.99 / 0.96
Ventilador	223.21/ 224	0.22/ 0.21	49.58/ 48	0.97/ 0.99
Lâmpada F.	223.68/ 223.8	0.11/ 0.12	15.04/ 15	0.63/ 0.59
Lâmpada I.	222.19/ 221.5	0.27/ 0.27	59.28/ 60	0.99/ 1
Mini Ventilador	219.15/ 218.9	0.08/ 0.09	12.07/ 12	0.71/ 0.59

4.2. Testes do BLOCO 2

Para serem realizados os testes no sistema principal, é necessário definir-se um cenário específico, com resultados esperados e, ao final, comparar os mesmos. Como primeiro cenário, testa-se o sistema em pleno funcionamento com ambas as cargas ligadas (lâmpada e ventilador) por 30 minutos. São analisados os valores monitorados e os gráficos de consumo gerados. Em seguida são testados os acionamentos de carga específicos, em regime automático e manual. Por último, são testadas as notificações sonoras, visuais e entrada de limites de consumo e temperatura.

4.2.1. Sistema em funcionamento

Para o primeiro teste, os *Arduinos* dos leitores de consumo são conectados ao sistema principal através de duas portas USB. A plataforma *Raspberry* é ligada à rede elétrica e em todos

os seus periféricos. Os equipamentos são ligados aos leitores, por sua vez ligados à rede elétrica. Os passos a seguir são realizados em um intervalo de tempo mais próximo possível (excluindo-se a espera de 30 min), garantindo uma maior consistência de resultados. O sistema é aberto e o som de inicialização é tocado. A janela apresentada na *Figura 49* se torna visível. Para garantir que os valores das medições pertençam apenas ao período de 30 minutos, o botão '*Apaga dados'* é pressionado, eliminando qualquer valor armazenado anteriormente. Tal botão foi incluído no sistema para facilitar a manipulação dos dados armazenados. As leituras, neste momento, apresentam valores *0*, com as cargas desligadas. O monitoramento será realizado com ambas as cargas ligadas, pressionando-se os botões Liga/ Desliga no leitor 1 e 2.

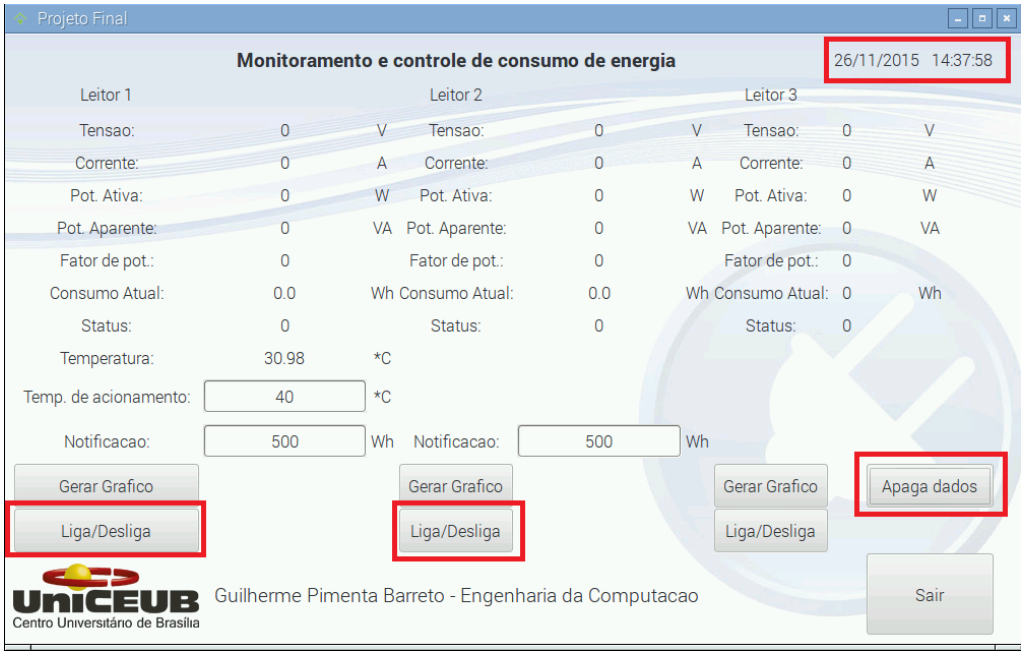

Figura 49 – Janela inicial do sistema em funcionamento

Fonte: Elaborada pelo autor

Com as cargas ligadas, o sistema mantém o monitoramento por 30 minutos, armazenando os valores mensurados. A *Figura 50* apresenta a janela apresentada antes dos 30 minutos. Os valores são atualizados dentro dos quadrados vermelhos delimitados.

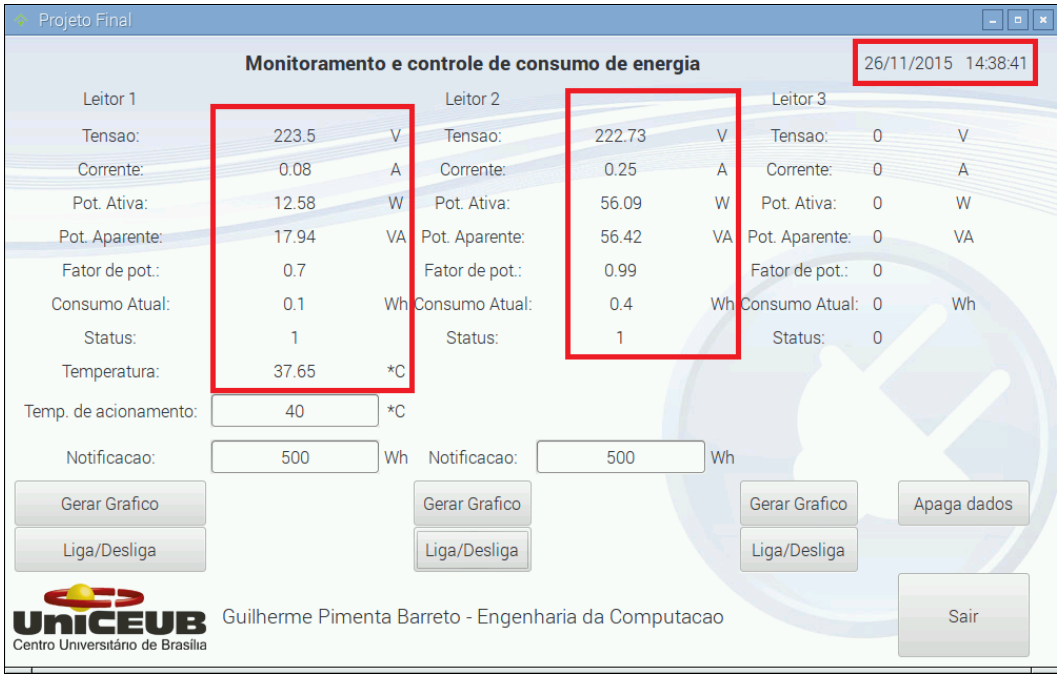

Figura 50 – Janela do sistema com as cargas acionadas

Fonte: Elaborada pelo autor

Após o período de monitoramento, as cargas foram desligadas pressionando-se os botões Liga/ Desliga novamente. Dessa forma, os valores de consumo apresentados são equivalentes ao período de 30 minutos. A janela após o desligamento das cargas é apresentada na *Figura 51*.

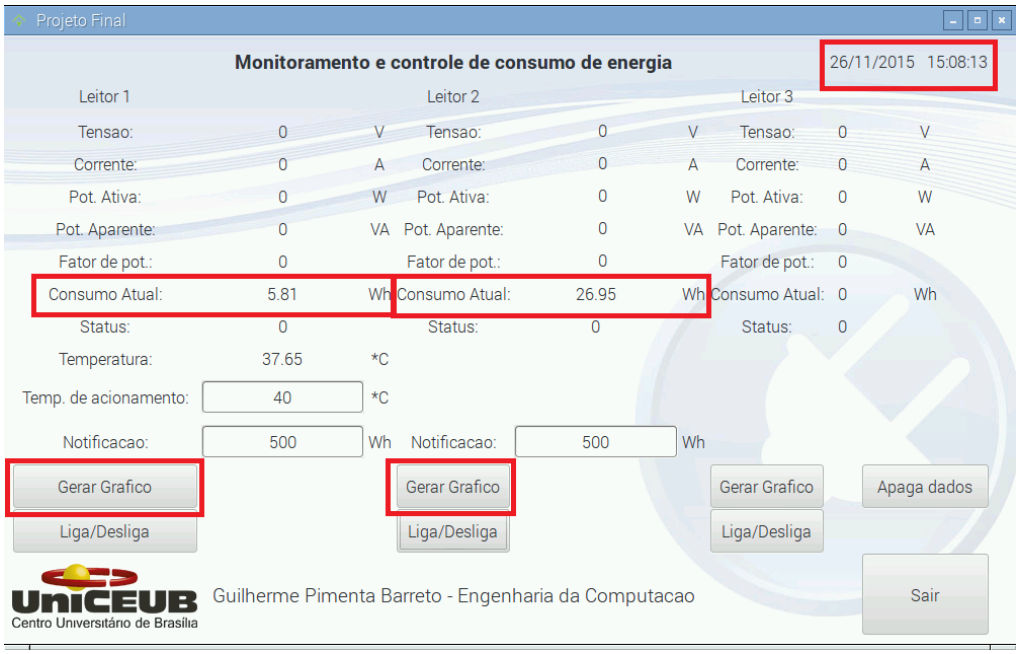

Figura 51 – Janela do sistema após 30 minutos

Fonte: Elaborada pelo autor

Em seguida, os gráficos de potência ativa, aparente e consumo de energia são gerados pressionando-se o botão '*Gerar gráfico'* em ambos os leitores. Para o leitor 1 (Mini Ventilador),

a *Figura 52* apresenta o gráfico criado com os aproximados 864 valores monitorados. O período do intervalo é apresentado pelo eixo *y*, abaixo do gráfico.

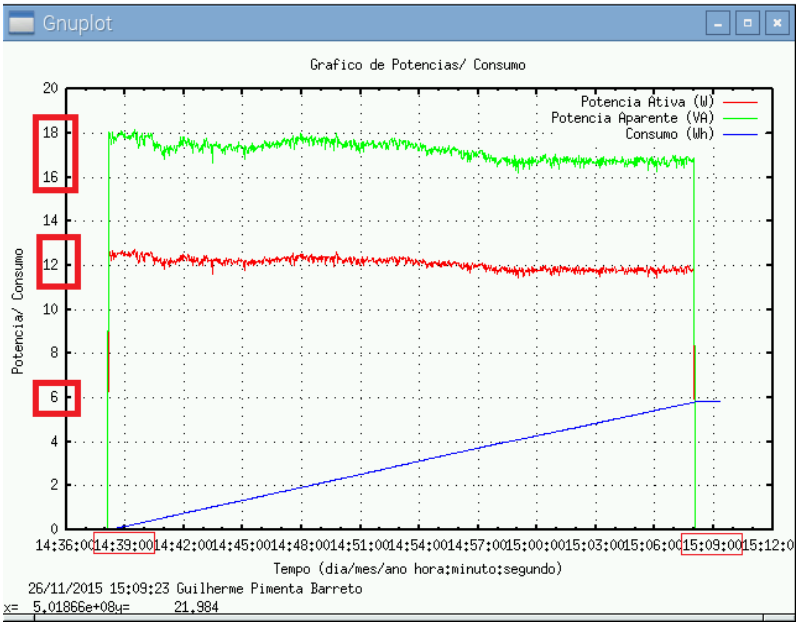

Figura 52 – Gráfico do leitor 1 após 30 minutos

Fonte: Elaborada pelo autor

Para a potência ativa (linha vermelha), a variação ocorreu dentro da margem de 11,5 a 12,5 W. A potência aparente (linha verde) atingiu valores dentro da margem de 16 a 18 VA. Esta diferença entre potência ativa e aparente é característica de cargas parcialmente indutivas, representando a parcela de potência reativa gerada através das bobinas existentes no ventilador. Para o consumo, a curva apresentada (linha azul) é praticamente linear, definida pelo uso sem interrupções da carga. O valor total de consumo atingido (5,81Wh) é equivalente ao cálculo de consumo de uma carga de 15W em funcionamento por 30 minutos, apresentada abaixo. Porém, foi observado no analisador de consumo DDS238-2 que a carga permanece a maior parte do tempo em 12W, não atingindo mais de 13W.

$$\boldsymbol{Consumo\ te\acute{o}rico = 12W\ x\ 0,5\ Hora = 6\ Wh} \qquad \textbf{(35)}$$

Similarmente ao leitor 1, a *Figura 53* apresenta o gráfico gerado com os valores monitorados do leitor 2 (Lâmpada incandescente).

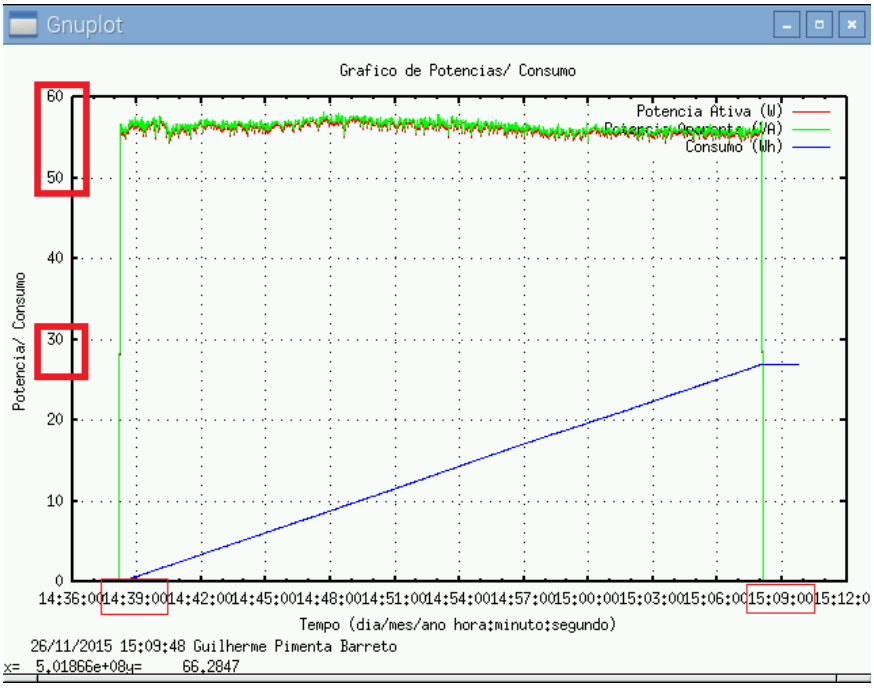

Figura 53 – Gráfico do leitor 2 após 30 minutos

Fonte: Elaborada pelo autor

Os valores de potência ativa (linha vermelha) alternaram na margem de 55W a 59W, assim como os valores de potência aparente (linha verde). Como a lâmpada incandescente representa uma carga puramente resistiva, toda a energia é consumida pela carga. A curva de consumo (linha azul) também possui uma progressão linear, possuindo uma amplitude maior que o gráfico anterior. Como a corrente que alimenta a lâmpada é superior ao ventilador, a mesma representa um maior consumo de energia (com uma tensão constante). O valor atingido (26,95Wh) equivale ao consumo ideal com a mesma lâmpada, apresentado abaixo.

$$\boldsymbol{Consumo\ te\acute{o}rico = 60W\ x\ 0,5\ Hora = 30\ Wh} \qquad \textit{(36)}$$

A *Figura 54* apresenta o Terminal em cada etapa do processo de testes. O momento seguinte à eliminação dos dados anteriores, após a inicialização do sistema. O momento seguinte ao acionamento das cargas 1 e 2, antes dos 30 minutos. O momento seguinte ao desligamento das cargas 1 e 2, após os 30 minutos e, por fim, o sistema é fechado pelo botão '*Sair'*.

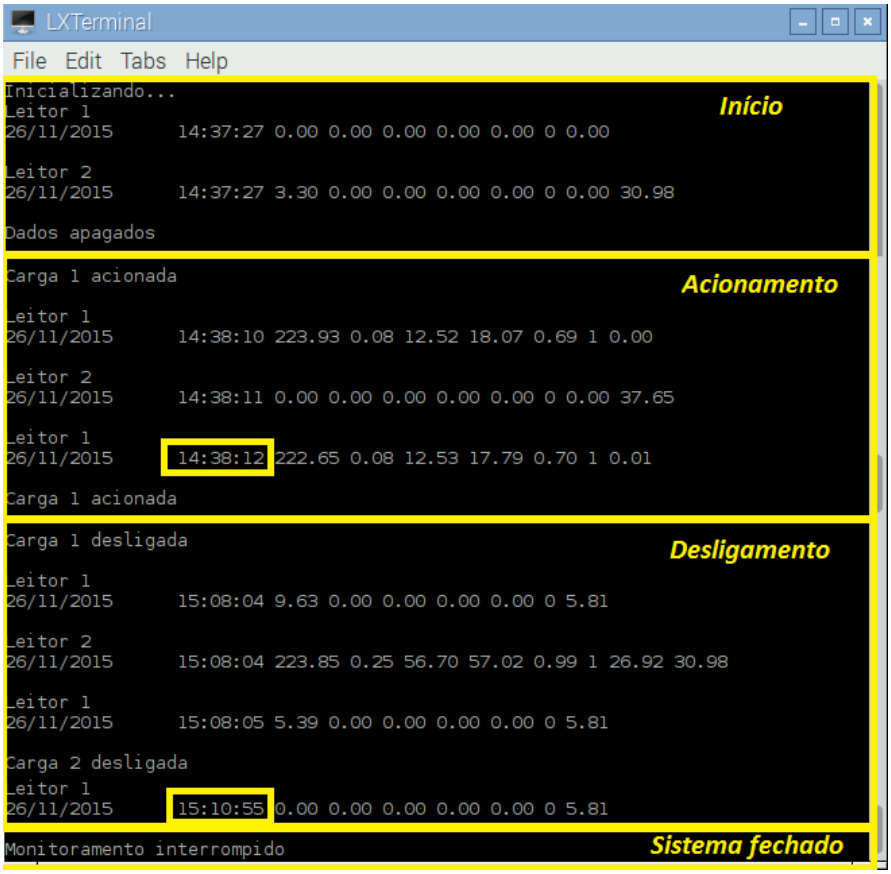

Figura 54 – Terminal em cada etapa do processo

Fonte: Elaborada pelo autor

O teste realizado no sistema principal apresentou resultados adequados ao erro exibido nos testes do BLOCO 1. O funcionamento do sistema não apresentou falhas no decorrer dos 30 minutos, realizando as etapas descritas de forma correta.

4.2.2. Acionamentos das cargas

Para o teste de acionamento e desligamento das cargas automaticamente, no sistema principal, são explorados os sensores referentes a cada leitor de consumo, acionados em condições específicas. A mesma disposição de cargas do teste anterior é utilizada, bem como o processo de apagar os dados anteriores para uma maior consistência dos resultados. O LM35, sensor de temperatura, envia os valores de temperatura ao sistema principal. Este aciona a carga ligada ao leitor 1, caso a medição atinja o limite definido pelo usuário (ou 40°C inicial). Para o leitor 2, o sensor de presença HC-SR501 envia um sinal alto (*1*) para qualquer alteração dos níveis infravermelhos do ambiente. Assim, um movimento na frente do sensor de presença gera o acionamento da carga relacionada a ele (assim como na ausência de movimento a carga é

desligada).

Com o sistema em funcionamento, os dados anteriores armazenados são apagados e as cargas permanecem desligadas. Para que ocorra o acionamento da carga ligada ao leitor 1 (mini ventilador) é necessário que a temperatura medida seja superior ao limite definido pelo usuário (ou inicial). Na entrada de dados *Temp. de acionamento* o valor de 40°C é alterado para 28°C, assim como na *Figura 55*.

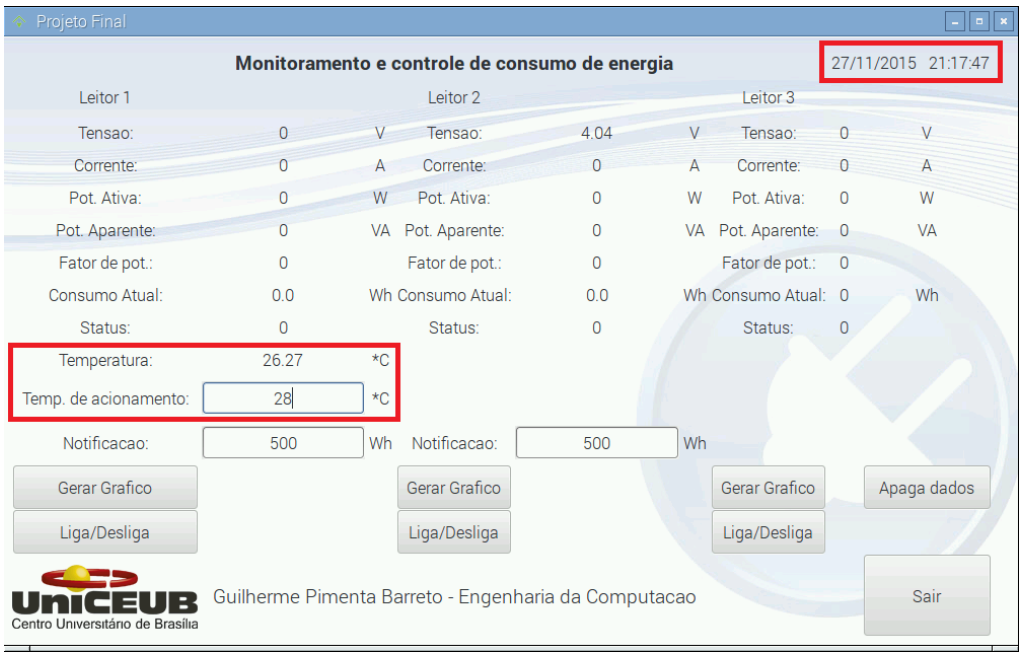

Figura 55 – Entrada de temperatura de acionamento

Fonte: Elaborada pelo autor

O sensor é aquecido com a temperatura da mão humana até ultrapassar o limite. A *Figura 56* apresenta a janela do sistema após o acionamento da carga, com a temperatura definida pelo usuário. A temperatura atinge o valor de 30,98°C, antes de voltar ao valor anterior de 26,27°C, gradualmente.

Figura 56 – Acionamento automático por temperatura

Fonte: Elaborada pelo autor

Para o teste acima, o sensor LM35 mostrou-se muito sensível a mudanças bruscas de temperatura, apresentando inconsistência de resultados com elevações acima de 5°C em um curto espaço de tempo. Como o sensor tem o objetivo de medir a temperatura ambiente, a mesma não demonstra variação tão acentuada em funcionamento normal, não apresentando risco de falha ao sistema.

Ainda com o sistema em funcionamento, para o acionamento do leitor 2 (lâmpada incandescente), a simulação de movimentação é realizada com uma mão humana, em frente ao sensor. A *Figura 57* apresenta a janela do sistema após o acionamento da carga.

Figura 57 – Acionamento automático por movimento

Fonte: Elaborada pelo autor

O sensor HC-SR501 possui dois potenciômetros para ajuste de sensibilidade e *delay*, por onde é possível realizar um ajuste fino de ambos os parâmetros. Para os testes utilizou-se um *delay* mínimo de aproximadamente 4 segundos. A sensibilidade foi mantida no mínimo em razão da proximidade dos testes.

O gráfico de acionamento do leitor 1 é apresentado na *Figura 58*. O monitoramento de potência ativa variou no intervalo de 12W a 14W, em aproximadamente 30 segundos. A potência aparente variou no intervalo de 17VA a 19VA, com um consumo acumulado de 0,1Wh. A diferença característica entre potência ativa e aparente foi mantida, como esperado.

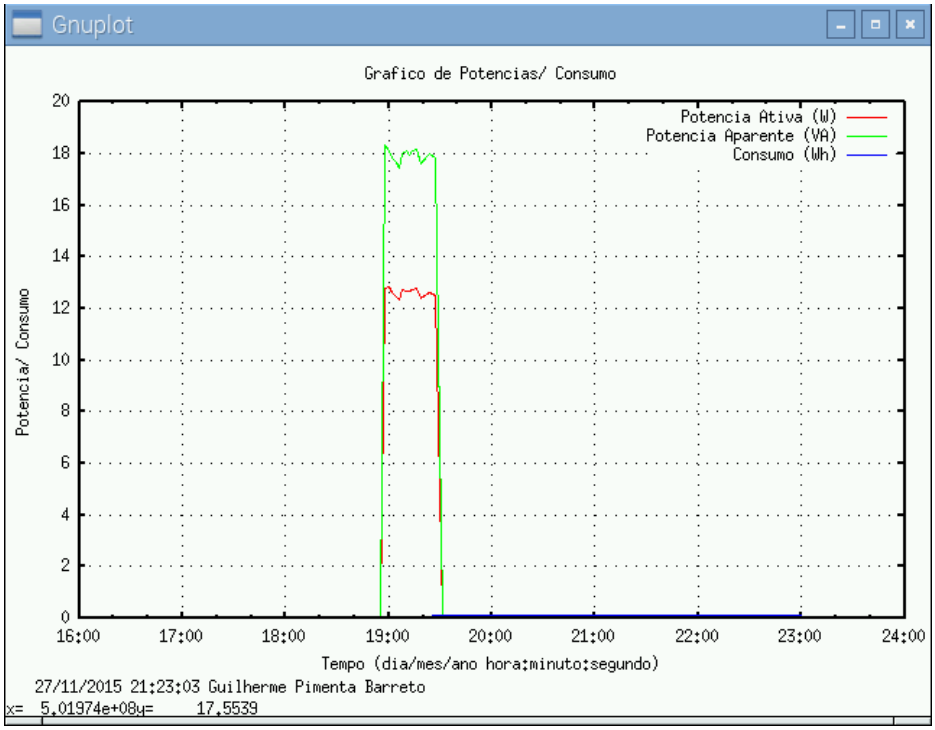

Figura 58 – Gráfico de acionamento por temperatura

Fonte: Elaborada pelo autor

Para o leitor 2, a *Figura 59* apresenta o gráfico de três acionamentos sucessivos em menos de 30 segundos. O período entre acionamentos não foi suficiente para mensurar a potência ativa, enquanto que a potência aparente variou no intervalo de 55VA a 60VA. O consumo acumulado representa 0,16Wh.

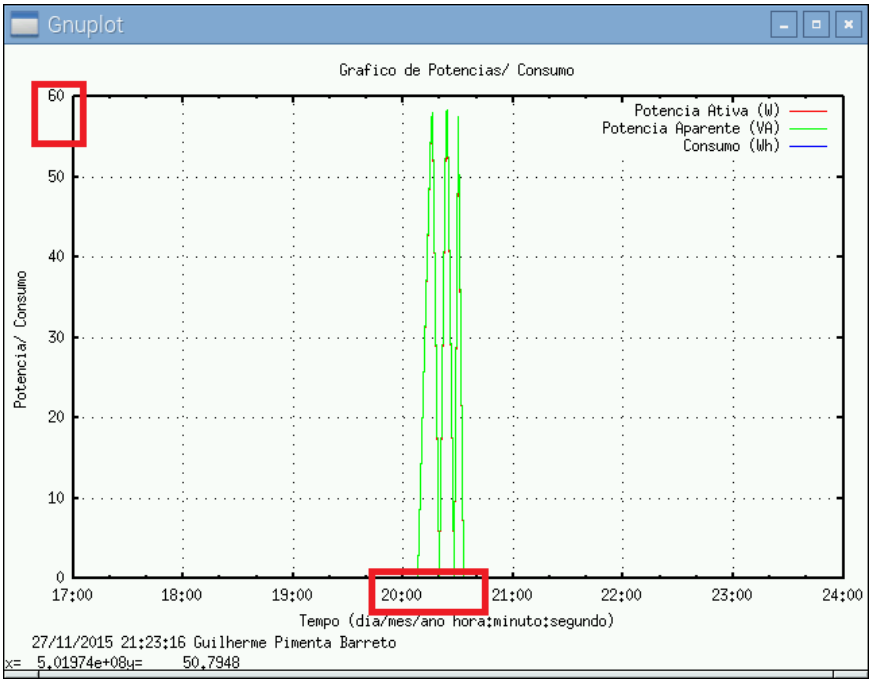

Figura 59 – Gráfico de acionamento por movimento

Fonte: Elaborada pelo autor

O Terminal, com todas as etapas acima, é apresentado na Figura 60. A inicialização e a eliminação dos dados iniciais são realizadas. O acionamento da carga no leitor 1 é efetuada com a ultrapassagem do limite definido, assim como o acionamento da carga no leitor 2, através da mudança de estado do sensor de presença.

O acionamento do controle por meio de sensores acontece automaticamente. Porém, pode ocorrer o acionamento manual enquanto o primeiro não houver cessado ainda. Nestes casos, os estado das cargas e dos sensores referentes às mesma são comparados. Se a carga foi acionada automaticamente, o estado do sensor foi alterado para *1*. Com o acionamento manual, o estado da carga também foi alterado para *1*. Para que a carga seja desligada, ambos os estados de carga e sensor devem ser alterados para *0*. Desta forma, se o acionamento da carga foi realizado manualmente, no intervalo em que o estado do sensor está em *1*, demonstra que o usuário deseja que a carga permaneça acionada após o término da condição de desligamento do sensor. Na alteração do mesmo para *0*, a carga é desligada com o desligamento manual, respeitando a decisão do usuário.

Figura 60 – Terminal dos testes de acionamento

Fonte: Elaborada pelo autor

Após a realização dos testes foi observado o correto funcionamento do sistema, com todas as características de acionamento apresentadas.

4.2.3. Notificações e entrada de dados

Além do Terminal auxiliar, já apresentado, o sistema possui alguns outros modos de interagir com o usuário. Sons de notificação emitidos pelo Buzzer, em algumas situações, e as entradas de limites de consumo de energia são utilizados para avisar o usuário de algum evento. Os textos de ajuda, apresentados ao se manter o cursor em um objeto da interface gráfica, também servem de auxílio na comunicação com o usuário. Os testes dos mesmos são descritos a seguir.

Em quatro situações específicas são reproduzidos os sons de notificação: Na inicialização do sistema, no acionamento das cargas, ao apagar os valores anteriores e quando os níveis de consumo atingem o limite definido. Para o primeiro, o sistema é iniciado normalmente e após o texto apresentado na *Figura 61*, o tom *1* da classe Buzzer é tocado. A função deste som de inicialização é puramente estética, indicando ao usuário o início do funcionamento do sistema.

Figura 61 – Som de notificação de inicialização do sistema

Fonte: Elaborada pelo autor

Com o acionamento de alguma das cargas (manualmente ou automaticamente) o tom *3* na classe Buzzer é tocado, após o texto de acionamento apresentado na *Figura 62*. Como a ideia geral do sistema também é o incentivo ao consumo de energia, o acionamento de uma carga, e o seu consequente gasto energético, sempre será avisado ao usuário.

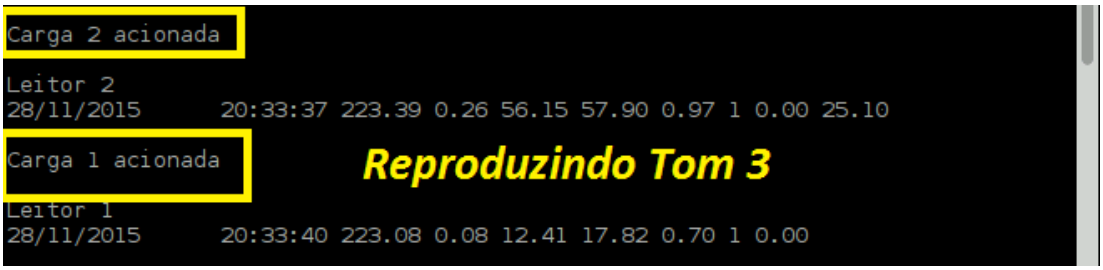

Figura 62 – Som de notificação de acionamento de cargas

Fonte: Elaborada pelo autor

Ao pressionar-se o botão *'Apagar dados'*, o comando *open(leitor,escrita).close()* em ambos

os leitores é executado e os valores apagados do arquivo *.dat*. Em seguida, o texto apresentado na *Figura 63* é exibido no Terminal, assim como o tom *2* é tocado no *Buzzer*. Como esta função é crítica para o sistema e causa a perda dos dados, uma notificação de sua execução se torna necessária.

Figura 63 – Som de notificação de eliminação de dados anteriores

Fonte: Elaborada pelo autor

O último teste, referente às notificações em questão, trata da entrada de dados pelos campos indicados pelo texto *Notificação*. Os valores inseridos no mesmo, seguidos do pressionamento da tecla *Enter* no teclado, são comparados com o valor atual acumulado de consumo atualizado pelo sistema na interface gráfica. Para o teste foi definido para o leitor 1 um consumo de 0,1Wh, assim como para o leitor 2 foi definido um consumo de 0.5Wh. Tais valores foram definidos antes que os mesmos fossem atingidos e na ocorrência do mesmo, as notificações textuais no Terminal, apresentados na *Figura 64*, foram seguidos da reprodução do tom *4* pelo Buzzer.

Figura 64 – Entrada de limites de consumo para notificação

Fonte: Elaborada pelo autor

Após os testes, foi constatado o correto funcionamento dos aspectos de notificação do sistema. Os sons foram reproduzidos nos momentos esperados e o texto no Terminal acompanhou os mesmos, com informações correspondentes.

Os textos de ajuda apresentados na *Figura 65* são exibidos na interface gráfica ao posicionar-se o ponteiro do mouse em cima de alguns objetos. O uso dos mesmos facilita a compreensão de algumas funções do sistema.

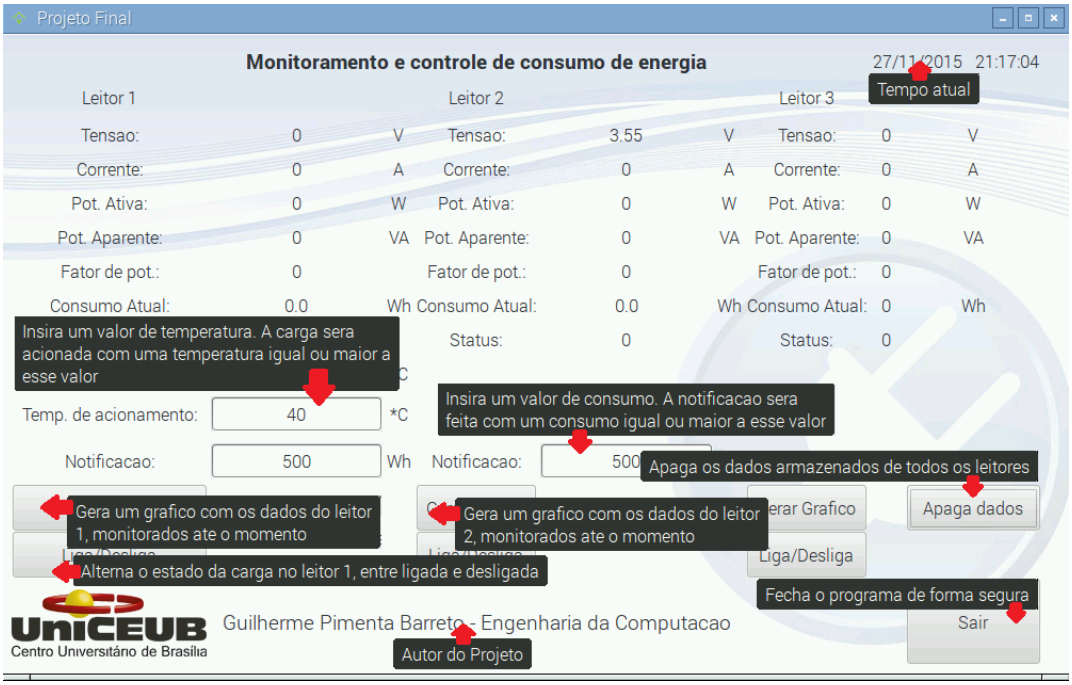

Figura 65 – Textos de ajuda da interface gráfica

Fonte: Elaborada pelo autor

4.3. Testes do BLOCO 3

Na calibração do leitor de consumo (*capítulo 3*), foi utilizada uma lâmpada incandescente de 60W e um mini ventilador de 15W, ambos presentes no protótipo final. Também utilizando o analisador de consumo DDS238-2 em série com o leitor de consumo, foram realizados os testes a seguir. O momento de comparação dos dados pode gerar uma pequena diferença nos valores. Um pequeno aumento na corrente medida no analisador de consumo é esperado, decorrente da corrente extra do leitor de consumo percorrendo-o.

4.3.1. Calibração da lâmpada incandescente

Para a calibração do primeiro parâmetro (tensão), obteve-se uma precisão de 99,7%. Tal exatidão foi possível através da utilização do transformador de 6V/200mA, assim como da biblioteca *EmonLib*. A baixa variação de tensão na rede elétrica contribuiu para uma maior precisão em equipamentos distintos, diferente do que ocorreu com a corrente e potência, que possuem variações muito mais acentuadas. A verificação da tensão é apresentada na Figura 66.

Figura 66 – Calibração da tensão

Fonte: Elaborada pelo autor

Para a calibração do segundo parâmetro (corrente), foi obtida uma precisão de 100%, atribuída a utilização do sensor ACS712 de 5A. Como a corrente atinge um valor máximo de 0,27A, o limite de 5A permite uma escala adequada de medição da mesma. A verificação da corrente é apresentada na Figura 67.

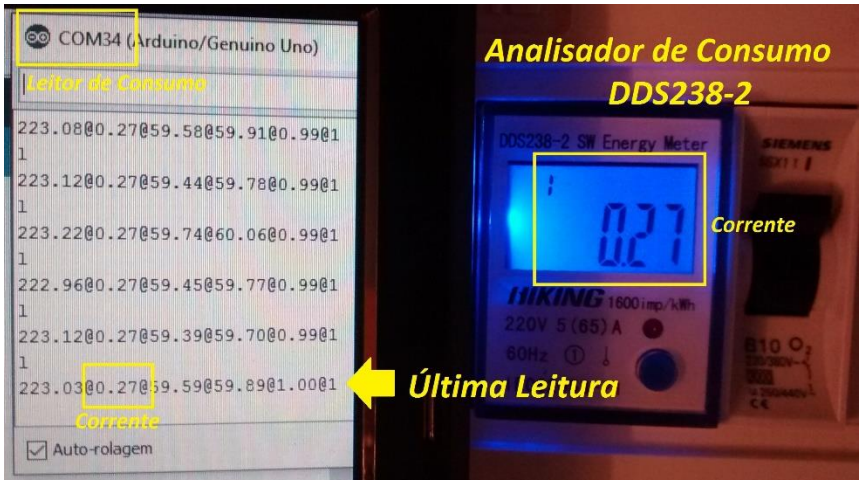

Figura 67 – Calibração da corrente

Fonte: Elaborada pelo autor

Na calibração da potência ativa, o bom dimensionamento do circuito sensor de tensão, assim como a precisão dos componentes e da biblioteca, acarretou numa precisão de 98,8% (4,8% a mais do que o esperado no início do projeto). A mesma precisão poderia ter sido alcançada nos testes com os outros equipamentos, caso a calibração individual fosse feita, mas como o foco do projeto é a utilização de uma lâmpada incandescente de 60W e um mini ventilador de 15W os resultados foram tolerados. A *Figura 68* apresenta a verificação da potência ativa descrita.

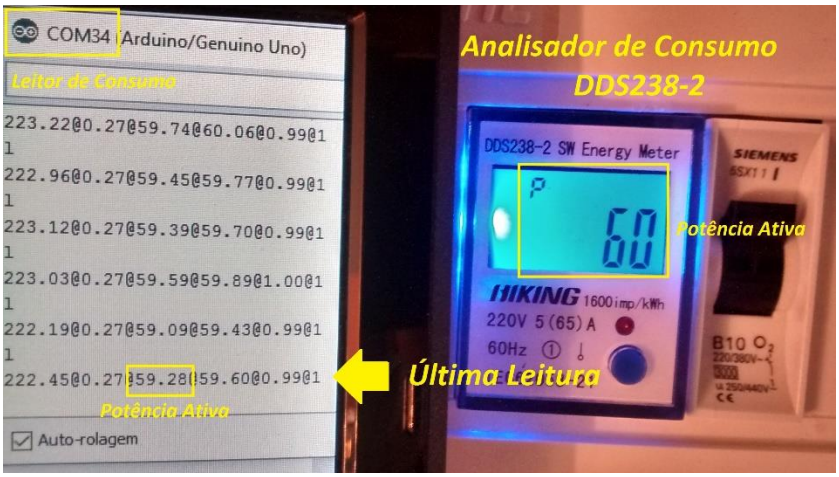

Figura 68 – Calibração da corrente

Fonte: Elaborada pelo autor

4.3.2. Calibração do mini ventilador

Similarmente à calibração do leitor de consumo com uma lâmpada incandescente, a calibração do mini ventilador foi realizada e será apresentada a seguir. Para a calibração de tensão foi alcançado o valor de *280*, alcançando uma precisão de 99,89%. Para a calibração de corrente, a precisão também foi de 99,89 com o valor de *1.8* e para a potência ativa, obteve-se uma exatidão de 99,42%. Tais resultados confirmam o impacto da calibração no valor final e suas verificações são apresentadas a seguir. A *Figura 69* representa a tensão, *Figura 70* representa a corrente e a potência ativa é exibida na *Figura 71*.

Figura 69 – Calibração da tensão

Fonte: Elaborada pelo autor

Figura 70 – Calibração da corrente

Fonte: Elaborada pelo autor

Figura 71 – Calibração da potência ativa

Fonte: Elaborada pelo autor

Após os testes realizados nos leitores de consumo, o funcionamento dos mesmos ocorreu de forma correta, com a precisão atingida nas medições próxima do esperado.

CAPÍTULO 5 – CONCLUSÃO

A energia elétrica é um bem indispensável na vida do ser humano e continuará sendo por muitos anos. O aumento na demanda deste recurso apresentará problemas de abastecimento e gerenciamento às cidades, com grandes impactos, principalmente, na população menos favorecida. O desenvolvimento das tecnologias renováveis mostra-se como uma boa proposta para a utilização de recursos em abundância no planeta, aumentando a diversidade de fontes sem agravar o impacto ambiental. Soluções mais próximas ao consumidor final traduzem-se no monitoramento e controle do seu próprio consumo, auxiliando numa melhor compreensão deste fluxo energético e permitindo o seu uso consciente.

A proposta do projeto, em questão, baseia-se no monitoramento dos níveis de consumo de energia elétrica em duas cargas monofásicas comumente utilizadas em uma residência, relacionadas à iluminação e condicionamento ambiental. Estes níveis de consumo são apresentados ao usuário através de uma interface simples e de fácil utilização, permitindo o controle das cargas de forma manual e automática.

Foram desenvolvidos dois leitores de consumo de energia, utilizando plataformas *Arduino* e sensores de tensão e corrente, capazes de medir os níveis de tensão, corrente, potência ativa, potência aparente e fator de potência das cargas conectadas a eles. Também foi desenvolvido um programa em linguagem *Python*, localizado na plataforma *Raspberry*, capaz de receber os valores de leitura dos sensores de temperatura, presença e leitores de consumo localizados nos cômodos das cargas. Os leitores e sistema principal comunicam-se por interfaces serial USB, trocando sinais de controle e leitura entre eles. A interface gráfica criada provê ao usuário os valores monitorados, além de possibilitar o controle das cargas e gerar gráficos de consumo das mesmas. Os protótipos finais criados são de pequeno porte, não ocupam muito espaço e o seu consumo não representa grande impacto no consumo da residência.

Dois tipos diferentes de plataformas, *Raspberry* e *Arduino*, são integradas utilizando a comunicação serial, tratando as especificidades de cada dispositivo de forma individual. Duas linguagens de programação diferentes, *C++* e *Python*, são desenvolvidas para o único objetivo de tornar o sistema funcional, agregando conhecimentos relacionados ao módulo *GTK* de desenvolvimento de interfaces, ferramenta *gnuplot* para geração de gráficos e biblioteca *Emonlib* de leitura de energia. São abordados temas relacionados à análise de energia elétrica, eletrônica, controle e automação e eficiência energética, além de várias outras abordagens menores.

Os objetivos definidos no início do planejamento do projeto foram atingidos e os

resultados dos testes comprovaram o correto funcionamento dos componentes envolvidos. Problemas e dificuldades encontradas no decorrer do projeto foram apresentadas e justificadas.

5.1. Trabalhos futuros

Nesta seção encontram-se diversas sugestões de aperfeiçoamento do sistema proposto, com ideias de implementação não realizadas, relacionadas ao sistema de controle e monitoramento de consumo de energia.

A) Utilização de tecnologia *wireless* na comunicação entre leitores de consumo, sistema principal e usuário (como *bluetooth* e rádio frequência), possibilitando o acesso do mesmo ao sistema em qualquer lugar da residência, reduzindo o consumo de energia relacionado ao monitor conectado ao *Raspberry* e aumentando a quantidade possível de leitores de consumo conectados a ele.

B) Criação de um sistema de banco de dados para gerenciamento dos valores de monitoramento armazenados, facilitando a pesquisa da informação por período de monitoramento e melhorando a eficiência do processo.

C) Utilização de microcontroladores PIC em substituição às plataformas *Arduino*, reduzindo custos drasticamente, bem como o tamanho total do circuito leitor de consumo.

D) Substituição do sensor de temperatura LM35 por um sensor equivalente digital, removendo a necessidade de um conversor A/D externo e reduzindo as oscilações bruscas de temperatura caracterizadas no projeto.

E) Utilização de contatoras no acionamento das cargas, possibilitando que correntes superiores a 7A sejam chaveadas e aumentando a gama de equipamentos potencialmente controláveis.

F) Realizar o estudo de qualidade de energia sobre a rede elétrica da residência, mensurando características de continuidade, suprimento e de conformidade através do monitoramento do sistema desenvolvido.

G) Adicionar novas funcionalidades ao controle, como o uso de *timers* para acionamento de cargas por tempo de operação.

REFERÊNCIAS

ADEMARO A. M. B. COTRIM. **Instalacoes Elétricas**. 5ª. ed. São Paulo: Pearson Prentice Hall, 2008. 1,3, 29-32 p.

ANEEL. BIG - Banco de Informações de Geração. **Agencia Nacional de Energia Elétrica**, 2015. Disponivel em: <http://www.aneel.gov.br/aplicacoes/capacidadebrasil/capacidadebrasil.cfm>. Acesso em: 06 Agosto 2015.

ANEEL. Fontes de Energia Exploradas no Brasil. **Agência Nacional de Energia Elétrica**, 2015. Disponivel em: <http://www.aneel.gov.br/aplicacoes/capacidadebrasil/FontesEnergia.asp>. Acesso em: 06 Agosto 2015.

ANEEL. Matriz de Energia Elétrica. **Agência Nacional de Energia Elétrica**, 2015. Disponivel em: <http://www.aneel.gov.br/aplicacoes/capacidadebrasil/OperacaoCapacidadeBrasil.cfm>. Acesso em: 06 Agosto 2015.

ARDUINO. Arduino - Serial. **Arduino**, 2015. Disponivel em: <https://www.arduino.cc/en/Reference/Serial>. Acesso em: 25 Setembro 2015.

ARDUINO. ArduinoBoardUno. **Arduino**, 2015. Disponivel em: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 30 Agosto 2015.

ARDUINO. Begin. **Arduino**, 2015. Disponivel em: <https://www.arduino.cc/en/Serial/Begin>. Acesso em: 21 Novembro 2015.

ARDUINO. DisablingAutoResetOnSerialConnection. **Arduino**, 2015. Disponivel em: <http://playground.arduino.cc/Main/DisablingAutoResetOnSerialConnection>. Acesso em: 20 Novembro 2015.

AYA, T. H. I.; EDGAR, M. M. U. **ANÁLISE DOS IMPACTOS AMBIENTAIS NA PRODUÇÃO DE ENERGIA**. Instituto de Eletrotécnica e Energia da Universidade de São Paulo (IEE/USP). São Paulo, p. 5. 2005.

BORGES, L. E. **Python para Desenvolvedores**. 2ª. ed. Rio de Janeiro: Edição do autor, 2010.

BOYLESTAD, R.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos**. 6ª. ed. Rio de Janeiro: LTC, 1999. 11, 88 p.

CHARLES , K.; MATTHEW, S. N. O. **Fundamentos de Circuitos Elétricos**. 5ª. ed. Porto Alegre: AMGH, 2013. 2-10,262,330,406-408,417-421, 493,511,528, 550 p.

DECKMANN, S. M.; POMILIO, J. A. **Avaliação da Qualidade da Energia Elétrica**. UNICAMP/FEEC/DSCE. Campinas, p. 5. 2010.

DEITEL, P.; HARVEY, D. **Java como programar**. 8ª. ed. Sao Paulo: Pearson, 2010. 5,6 p.

EIA. International Energy Statistics - EIA. **Independent Statistics & Analysis U.S. Energy Information Administration**, 2012. Disponivel em: <http://www.eia.gov/cfapps/ipdbproject/IEDIndex3.cfm?tid=2&pid=2&aid=2>. Acesso em: 20 Agosto 2015.

EIA. **International Energy Outlook 2014: World Petroleum and Other Liquid Fuels**. U.S. Energy Information Administration. Washington. 2014.

ELETROPAULO. Entenda o aumento na conta de energia. **AES Eletropaulo**, 2015. Disponivel em: <https://www.aeseletropaulo.com.br/para-sua-casa/prazos-e-tarifas/conteudo/entenda-o-aumento-na-conta-de-energia-(mar%C3%A7o2015)>. Acesso em: 07 Agosto 2015.

EPE. **Eficiência energética na indústria e nas residências no horizonte decenal (2010-2019)**. Empresa de Pesquisa Energética. Rio de Janeiro, p. 2,26. 2010. (NOTA TÉCNICA DEA 14/10).

EPE. **Consumo de Energia no Brasil: Análises Setorias**. Empresa de Pesquisa Energética. Rio de Janeiro, p. 11,89. 2014. (NOTA TÉCNICA DEA 10/14).

EPE. **Demanda de Energia 2050**. Empresa de Pesquisa Energética. Rio de Janeiro, p. 198. 2014. (NOTA TÉCNICA DEA 13/14).

EXXONMOBIL. **Panorama Energético: Perspectivas para 2040**. Exxon Mobil Corporation. Irving, p. 4. 2014.

FACCIN, F. **Abordagem Inovadora no Projeto de Controladores PID - Dissertação de Mestrado**. Universidade Federal do Rio Grande do Sul. Porto Alegre, p. 2. 2004.

FERREIRA, J. B. **Análise de Formas de Medição de Consumo de Energia Elétrica no Setor Residencial**. Universidade Federal de Pernambuco. Recife, p. 22-24. 2012. (Trabalho de Graduação).

GNUPLOT. gnuplot homepage. **Gnuplot**, 2015. Disponivel em: <http://www.gnuplot.info/>.

Acesso em: 19 Outubro 2015.

HENRIQUES, A. M. et al. **Eficiência Energética Teoria & Prática**. 1ª. ed. Itajubá: Eletrobrás/ PROCEL EDUCAÇÃO, v. editor, 2007. 1,2, 238 p.

HUDSON, G.; LEA, T. AC Power Theory - Arduino maths. **OpenEnergyMonitor**, 2015. Disponivel em: <openenergymonitor.org/emon/buildingblocks/ac-power-arduino-maths>. Acesso em: 28 Setembro 2015.

HUDSON, G.; LEA, T. Calibration. **OpenEnergyMonitor**, 2015. Disponivel em: <http://openenergymonitor.org/emon/buildingblocks/calibration>. Acesso em: 08 Outubro 2015.

HUDSON, G.; LEA, T. Measurement implications of ADC resolutions at low current values. **OpenEnergyMonitor**, 2015. Disponivel em: <http://openenergymonitor.org/emon/buildingblocks/measurement-implications-of-adc-resolution-at-low-current-values>. Acesso em: 27 Novembro 2015.

INTEL. USB – Universal Serial Bus Overview. **Intel**, 2015. Disponivel em: <http://www.intel.com/content/www/us/en/io/universal-serial-bus/universal-serial-bus.html>. Acesso em: 19 Outubro 2015.

INTEL et al. **Universal Serial Bus Specification Revision 2.0**. Compaq Computer Corporation,Hewlett-Packard CompanyIntel Corporation, Lucent Technologies Inc,Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V. [S.l.], p. 17,102. 2000.

KONDO, M. M. Energia é fator determinante no desenvolvimento da sociedade. **Folha de S.Paulo**, São Paulo, 16 Fevereiro 2005. Disponivel em: <http://vestibular.uol.com.br/resumo-das-disciplinas/geografia/energia-e-fator-determinante-no-desenvolvimento-da-sociedade.htm>. Acesso em: 2015.

KUROSE, J. F. **Redes de Computadores e a Internet**. 5ª. ed. São Paulo: Pearson, 2009. 326 p.

LAMBERTS, R.; DUTRA, L.; PEREIRA, F. **Eficiencia Energetica na Arquitetura**. 3ª. ed. Rio de Janeiro: ELETROBRÁS, 2014. 161 p.

LIECHTI, C. pySerial 2.7 documentation. **PYSERIAL**, 2013. Disponivel em: <https://pythonhosted.org/pyserial/>. Acesso em: 20 Outubro 2015.

LIMA, J. E. F. W. **Recursos Hídricos no Brasil e no Mundo**. Embrapa. Planaltina, p. 20. 2001.

NILSSON, J. W.; RIEDEL, S. A. **Circuitos Elétricos**. 8ª. ed. São Paulo: Pearson Prentice Hall, 2009. 7-9,230-232, 234 p.

O. SIDDIQUI. **The Green Grid: Energy Savings and Carbon Emissions Reductions Enabled by a Smart Grid**. Eletric Power Reserch Institute. Palo Alto, p. 1-1. 2008. (1016905).

OGATA, K. **Engenharia de controle moderno**. 4ª. ed. Rio de Janeiro: Pearson Prentice Hall, 2003. 1,4-6, 170-173 p.

ONS. Geração de Energia - Hidráulica -. **Operador Nacional do Sistema Elétrico**, 2015. Disponivel em: <http://www.ons.org.br/historico/geracao_energia.aspx>. Acesso em: 07 Agosto 2015.

PEACOCK, C. USB in a NutShell - Chapter 3 - USB Protocols. **Beyond Logic**, 2010. Disponivel em: <http://www.beyondlogic.org/usbnutshell/usb3.shtml>. Acesso em: 29 Novembro 2015.

PINTO, J. F. **Magnometria por efeito Hall**. Universidade Federal de Pernambuco. Recife, p. 10. 2010.

PRATYAKSA, D. How to Play Piezo Buzzer Tunes on RaspberryPi GPIO With PWM. **LinuxCircle.com**, 2015. Disponivel em: <http://www.linuxcircle.com/2015/04/12/how-to-play-piezo-buzzer-tunes-on-raspberry-pi-gpio-with-pwm/>. Acesso em: 23 Novembro 2015.

PROCEL. **Metodologia de Realização de Diagnóstico Energético**. ELETROBRÁS. Brasília, p. 41. 2009. (3).

PYGTK TEAM. PyGTK: GTK+ for Python. **PyGTK**, 2009. Disponivel em: <http://www.pygtk.org/>. Acesso em: 19 Outubro 2015.

PYTHON. Getting started with python. **Python**, 2015. Disponivel em: <http://thepythonguru.com/getting-started-with-python/>. Acesso em: 31 Agosto 2015.

RASPBERRY. USB - Raspberry Pi Documentation. **Raspberry**, 2015. Disponivel em: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/usb/README.md>. Acesso em: 20 Outubro 2015.

RASPBERRY FOUNDATION. RASPBERRY PI 2 MODEL B. **Raspberry PI**, 2015. Disponivel em: <https://www.raspberrypi.org/>. Acesso em: 07 Setembro 2015.

SANTOS, A. H. et al. **Conservação de energia Eficiência energética de Equipamentos e**

Instalacoes. 3ª. ed. Itajubá: Eletrobrás/ PROCEL educação, 2006. 17,29-33,149,160-162,225-231 p.

SMUTS, J. PID Controllers Explained. **Control Notes**, 2011. Disponivel em: <http://blog.opticontrols.com/archives/date/2011/03/page/2>. Acesso em: 31 Agosto 2015.

THE WORLD BANK. World Development Indicators The World Bank. **The World Bank**, 2010. Disponivel em: <http://wdi.worldbank.org/table/3.7>. Acesso em: 20 Agosto 2015.

UNEP. **Global Trends in Renewable Energy Investment 2015**. Frankfurt School. Frankfurt, p. 13. 2015.

US GAO. **Nuclear Reactors: Status and challenges in development and deployment of new commercial concepts**. Center for Science, Technology, and Engineering Natural Resources and Environment. [S.l.], p. 6. 2015.

VIANA, N. A. et al. **Eficiência Energetica Fundamentos e Aplicações**. 1ª. ed. Campinas: Elektro - Eletricidade e Serviços S.A., 2012. 13-27,31,49,58-59, 86-89,293,300 p.

APÊNDICES

APÊNDICE A – CÓDIGO DO LEITOR DE CONSUMO DE ENERGIA

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Trabalho de Conclusao de Curso do Centro Universitario de Brasilia
// Ra: 21136802
// Aluno: Guilherme Pimenta Barreto
// Curso: Engenharia da computação
// Periodo: 11/2015
// Programa do Leitor de Consumo de energia, responsavel por realizar a leitura
// dos valores de tensao, corrente, potencia ativa, potencia aparente, fator
// de potencia e estado das cargas conectadas, alem de realizar o acionamento
// das mesmas, junto a uma comunicacao serial.
////////////////////////////////// - Definicoes - ////////////////////////////////////////////////////////////////
//Biblioteca de leitura dos valores EmonLib
#include "EmonLib.h"
//Cria uma instancia do monitor de energia especifico da biblioteca EmonLib
EnergyMonitor emon1;
//Aloca o valor da porta analogica 4 na variavel rele
int rele=A4;
//Declara a variavel boleana estado com o valor de 0
boolean estado=1;
//Variavel que recebe a leitura da serial
char controleLeitura='0';
//Declara a variavel responsavel por receber todos os valores de leitura como uma string
String valores;
//Declara a variavel responsavel por delimitar cada valor individual no envio da serial
char const* separador="@";
////////////////////////////////// - Setup - ///////////////////////////////////////
void setup()
{
  //Inicia a serial com a taxa em bauds de 115200
  Serial.begin(115200);
  // Define parâmetros da medição de tensão (Pino, calibracao, angulo de fase)
  //emon1.voltage(2, 269, -0.10);   // Sensor de 5A com lampada incandescente
  emon1.voltage(2, 280, -1.1);   // Sensor de 5A com ventilador
  // Define parâmetros da medição de corrente (Pino, calibracao)
  //emon1.current(1, 4.5);          // Sensor de 5A com lampada incandescente
```

```
  emon1.current(1, 1.8);          // Sensor de 5A com ventilador
  //Define o comportamento do pino do rele como saida de dados
  pinMode (rele, OUTPUT);
  //Envia um sinal baixo ao pino do rele, desligando-o no inicio do programa
  digitalWrite(rele,LOW);
}

////////////////////////////////// - Loop - /////////////////////////////////////
void loop(){
  //Chama a funcao de controle dentro da recurcao
  controle();
}

////////////////////////////////// - Funcao de controle - ///////////////////////////
void controle(){
    //Verificacao de conexao
    if(Serial.available() > 0){
      //Aloca o valor da leitura do sinal de controle
      controleLeitura=Serial.read();
      //Retorna o sinal de controle
      Serial.println(controleLeitura);
      //Selecao do controle
      switch (controleLeitura) {
        //Sinal alto aciona a carga com o rele e modifica o estado para 1
        case '1':
        estado=1;
        digitalWrite(rele,HIGH);
        break;
        //Sinal baixo desliga a carga com o rele e modifica o estado para 0
        case '0':
        estado=0;
        digitalWrite(rele,LOW);
        break;
        //Sinal com a string 'l' chama a funcao leitura
        case 'l':
        leitura();
        break;
      }
    }
}
```

```
//////////////////////////////////// - Funcao de leitura - ////////////////////////////
 void leitura(){
  //Realiza a leitura dos valores e define seus parametros (crossings, time-out)
  emon1.calcVI(60,2000);
  //Aloca os valores das leituras em suas respectivas variaveis de tipo float
  float tensao=emon1.Vrms;
  float corrente=emon1.Irms;
  float ativa=emon1.realPower;
  float aparente=emon1.apparentPower;
  float fator=emon1.powerFactor;
  //Cria a string na estrutura de envio, concatenando os valores e separadores

valores=String(tensao)+separador+String(corrente)+separador+String(ativa)+separador+String(aparente)+separado
r+String(fator)+separador+String(estado);
  //Envia a string valores
  Serial.println(valores);
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

APÊNDICE B – CÓDIGO DO SISTEMA PRINCIPAL

```
#!/usr/bin/env python

# Trabalho de Conclusao de Curso do Centro Universitario de Brasilia
# Ra: 21136802
# Aluno: Guilherme Pimenta Barreto
# Curso: Engenharia da computacao
# Periodo: 11/2015
# Programa do Sistema Principal, responsavel por receber os valores de
# leitura de tensao, corrente, potencia ativa, potencia aparente, fator
# de potencia e estado das cargas conectadas nos modulos leitores de consumo.
# Estes valores sao atualizados na interface grafica (GUI) e seus graficos,
# gerados pelo botao 'Gerar Grafico'. As cargas sao controladas internamente.

###################### - Modulos - ##########################################
#Modulo GTK
import gtk
#Modulo serial
import serial
#Modulo de manipulacao de data e hora
```

```
from datetime import datetime
#Modulo de criacao de processos
import subprocess
#Modulo que adiciona atributos adicionais aos objetos GTK
import gobject
#Modulo de manipulacao das GPIOs
import RPi.GPIO as GPIO
#Modulo de tempo
import time
#Subconjunto do protocolo I2C
from smbus import SMBus

###################### - Definicao de GPIO - ################################
#Define o modo de numeracao dos pinos para BCM (Broadcom SOC Channel)
GPIO.setmode(GPIO.BCM)
#Desativa mensagens de aviso da GPIO
GPIO.setwarnings(False)

#Artibui o valor do pino do sensor de presenca com o valor 07 (GPIO 07)
pinoPIR=07
#Artibui o valor do pino do sensor de temperatura com o valor 24 (GPIO 24)
pinoDHT=24
#Artibui o valor do pino do buzzer com o valor 12 (GPIO 12)
pinoBuzzer=12

#Define o pino do sensor de presenca como entrada de dados (Input)
GPIO.setup(pinoPIR,GPIO.IN)
#Define o pino do sensor de temperatura como entrada de dados (Input)
GPIO.setup(pinoDHT,GPIO.IN)
#Define o pino do buzzer como saida de dados (Output)
GPIO.setup(pinoBuzzer, GPIO.OUT)

#Define a porta I2C 1 (/dev/i2c-1) e cria uma instancia dela
i2cBus = SMBus(1)
# Endereco padrao na I2C (0x48) e registrador de controle 0
i2cBus.write_byte(0x48,0)

###################### - Definicao da serial - ##############################
# Cria uma instancia da serial para cada leitor de consumo
# Primeiro parametro e o caminho da porta
# Segundo e a taxa da comunicacao em bauds
```

```
leitor1 = serial.Serial('/dev/ttyACM0', 115200)
leitor2 = serial.Serial('/dev/ttyACM1', 115200)
time.sleep(1)
###################### - Classe da GUI - ####################################
#Cria uma classe com heranca do objeto window (janela)
class GUI(gtk.Window):
	#Metodo construtor responsavel por inicializar parametros
	def __init__(self):
		super(GUI, self).__init__()
		############## - Criacao da janela - ##################
		#Define o titulo da janela
		self.set_title("Projeto Final")
		#Define a posicao da janela
		self.set_position(gtk.WIN_POS_CENTER)
		#Define o tamanho da janela
		self.set_size_request (1000,600)
		#Define a largura da borda da janela
		self.set_border_width(8)
		#Define imagem do icone da janela
		self.set_icon_from_file("icone.png")
		#Conecta o evento de fechar a janela ao clicar no x
		self.connect("destroy", gtk.main_quit)

		#Cria um objeto contendo uma imagem
		self.logo = gtk.gdk.pixbuf_new_from_file("ceub.png")
		#Construtor para criacao de uma imagem vazia
		ceub=gtk.Image()
		#Define a imagem
		ceub.set_from_pixbuf(self.logo)

		############## - Variaveis de alocacao e controle- ##################
		#Variavel de alocacao do estado da carga
		self.controleCarga1=0
		self.controleCarga2=0
		#Variavel de alocacao do estado do sensor PIR
		self.controlePIR=0;
		self.PIRAnterior=0;
		#Variavel de alocacao do estado do sensor de temperatura
		self.controleLM35=0;
		self.LM35Anterior=0;
		#Variavel de alocacao da temperatura
```

```
self.temp=0;
#Variavel de alocacao da temperatura definida
self.tempDefinida=40;
#Variavel de alocacao do consumo definido
self.consumoDefinido1=500;
self.consumoDefinido2=500;
#Variavel de controle do estado anterior da carga
self.cargaAnterior1=0
self.cargaAnterior2=0
#Variavel de alocacao da soma de consumo do leitor 1
self.somaConsumo1=0
#Variavel de alocacao da soma de consumo do leitor 2
self.somaConsumo2=0
#Variavel de selecao da porta serial
self.controleLeitor=1

############ - Criacao da entrada de dados - ################
#Construtor para criacao de uma caixa de texto de entrada
self.entradaTemp=gtk.Entry()
#Define alinhamento do texto dentro da caixa de texto de entrada
self.entradaTemp.set_alignment(xalign=0.5)
#Define o texto inicial dentro da caixa de texto de entrada
self.entradaTemp.set_text(str(self.tempDefinida))
#Conecta o evento ao pressionar 'Enter' dentro da caixa de texto
self.entradaTemp.connect('activate', self.defineTemp)

#Construtor para criacao de uma caixa de texto de entrada
self.entradaConsumo1=gtk.Entry()
#Define alinhamento do texto dentro da caixa de texto de entrada
self.entradaConsumo1.set_alignment(xalign=0.5)
#Define o texto inicial dentro da caixa de texto de entrada
self.entradaConsumo1.set_text(str(self.consumoDefinido1))
#Conecta o evento ao pressionar 'Enter' dentro da caixa de texto
self.entradaConsumo1.connect('activate', self.defineConsumo1)

#Construtor para criacao de uma caixa de texto de entrada
self.entradaConsumo2=gtk.Entry()
#Define alinhamento do texto dentro da caixa de texto de entrada
self.entradaConsumo2.set_alignment(xalign=0.5)
#Define o texto inicial dentro da caixa de texto de entrada
self.entradaConsumo2.set_text(str(self.consumoDefinido2))
```

```
#Conecta o evento ao pressionar 'Enter' dentro da caixa de texto
self.entradaConsumo2.connect('activate', self.defineConsumo2)

############### - Criacao de labels - ###################
#Cria labels para apresentacao dos valores
#Valores iniciais definidos como '0'
self.tensao1 = gtk.Label(str(0))
self.corrente1=gtk.Label(str(0))
self.ativa1=gtk.Label(str(0))
self.aparente1=gtk.Label(str(0))
self.fator1=gtk.Label(str(0))
self.consumo1=gtk.Label(str(0))
self.estado1=gtk.Label(str(0))
self.tempLabel=gtk.Label(str(0))

self.tensao2=gtk.Label(str(0))
self.corrente2=gtk.Label(str(0))
self.ativa2=gtk.Label(str(0))
self.aparente2=gtk.Label(str(0))
self.fator2=gtk.Label(str(0))
self.consumo2=gtk.Label(str(0))
self.estado2=gtk.Label(str(0))

self.tensao3=gtk.Label(str(0))
self.corrente3=gtk.Label(str(0))
self.ativa3=gtk.Label(str(0))
self.aparente3=gtk.Label(str(0))
self.fator3=gtk.Label(str(0))
self.consumo3=gtk.Label(str(0))
self.estado3=gtk.Label(str(0))

#Cria a label data
self.data = gtk.Label()
#Cria a label nome
nome = gtk.Label()
#Define parametros da label nome
nome.set_markup("<big>Guilherme Pimenta Barreto - Engenharia da Computacao</big>")
#Cria a label titulo
title = gtk.Label()
#Define parametros da label titulo
title.set_markup("<big><b>Monitoramento e controle de consumo de energia</b></big>")
```

```
############## - Criacao de containers - ###################
#Construtor para criacao de um conteiner widget vertical
vbox = gtk.VBox(False,2)
#Construtor para criacao de um conteiner widget tabela
table = gtk.Table(10, 15, False)
#Define espacamento entre colunas da tabela
table.set_col_spacings(3)
#Define espacamento entre linhas da tabela
table.set_row_spacing(1, 3)

############## - Criacao de botoes - ###################
#Cria os botoes com texto especifico
grafico1 = gtk.Button("Gerar Grafico")
liga1 = gtk.Button("Liga/Desliga")
grafico2 = gtk.Button("Gerar Grafico")
liga2 = gtk.Button("Liga/Desliga")
grafico3 = gtk.Button("Gerar Grafico")
liga3 = gtk.Button("Liga/Desliga")
apaga = gtk.Button("Apaga dados")
sair = gtk.Button("Sair")

#Conecta eventos ao pressionar os botoes
#Metodos especificos sao chamados
grafico1.connect("clicked", self.gnuplot1)
liga1.connect("clicked", self.acionaCarga1)
grafico2.connect("clicked", self.gnuplot2)
liga2.connect("clicked", self.acionaCarga2)
apaga.connect("clicked", self.apagaDados)
sair.connect("clicked", self.sairConfirma)

############## - Criacao de texto de ajuda - #############
#Define o texto de ajuda ao manter o cursor no objeto
self.entradaTemp.set_tooltip_text('Insira um valor de temperatura. A carga sera acionada com uma
temperatura igual ou maior a esse valor')
self.data.set_tooltip_text('Tempo atual')
nome.set_tooltip_text('Autor do Projeto')
grafico1.set_tooltip_text('Gera um grafico com os dados do leitor 1, monitorados ate o momento')
liga1.set_tooltip_text('Alterna o estado da carga no leitor 1, entre ligada e desligada')
grafico2.set_tooltip_text('Gera um grafico com os dados do leitor 2, monitorados ate o momento')
liga2.set_tooltip_text('Alterna o estado da carga no leitor 2, entre ligada e desligada')
```

```
        grafico3.set_tooltip_text('Gera um grafico com os dados do leitor 3, monitorados ate o momento')
        liga3.set_tooltip_text('Alterna o estado da carga no leitor 3, entre ligada e desligada')
        apaga.set_tooltip_text('Apaga os dados armazenados de todos os leitores')
        sair.set_tooltip_text('Fecha o programa de forma segura')
        self.entradaConsumo1.set_tooltip_text('Insira um valor de consumo. A notificacao sera feita com um consumo igual ou maior a esse valor')
        self.entradaConsumo2.set_tooltip_text('Insira um valor de consumo. A notificacao sera feita com um consumo igual ou maior a esse valor')

        ############## - Insercao na tabela - ###################
        #Insere a label titulo na tabela com a posicao especificada
        table.attach(title,1 , 6, 0, 1)
        #Insere a label data na tabela com a posicao especificada
        table.attach(self.data,7,9,0,1)

        #Insere os widgets na tabela com a posicao especificada. Primeira coluna
        table.attach(gtk.Label("Leitor 1"),0,1,1,2)
        table.attach(gtk.Label("Tensao:"),0,1,2,3)
        table.attach(gtk.Label("Corrente:"),0,1,3,4)
        table.attach(gtk.Label("Pot. Ativa:"),0,1,4,5)
        table.attach(gtk.Label("Pot. Aparente:"),0,1,5,6)
        table.attach(gtk.Label("Fator de pot.:"),0,1,6,7)
        table.attach(gtk.Label("Consumo Atual:"),0,1,7,8)
        table.attach(gtk.Label("Status:"),0,1,8,9)
        table.attach(gtk.Label("Temperatura:"),0,1,9,10)
        table.attach(gtk.Label("Temp. de acionamento:"),0,1,10,11)
        table.attach(gtk.Label("Notificacao:"),0,1,11,12)
        table.attach(grafico1, 0, 1, 12, 13)
        table.attach(liga1, 0, 1, 13, 14)
        table.attach(ceub, 0, 1, 14, 15)

        #Insere os widgets na tabela com a posicao especificada. Segunda coluna
        table.attach(self.tensao1,1,2,2,3)
        table.attach(self.corrente1,1,2,3,4)
        table.attach(self.ativa1,1,2,4,5)
        table.attach(self.aparente1,1,2,5,6)
        table.attach(self.fator1,1,2,6,7)
        table.attach(self.consumo1,1,2,7,8)
        table.attach(self.estado1,1,2,8,9)
        table.attach(self.tempLabel,1,2,9,10)
        table.attach(self.entradaTemp,1,2,10,11)
```

```
table.attach(self.entradaConsumo1,1,2,11,12)
table.attach(nome, 1, 6, 14, 15)

#Insere os widgets na tabela com a posicao especificada. Terceira coluna
table.attach(gtk.Label("V"),2,3,2,3)
table.attach(gtk.Label("A"),2,3,3,4)
table.attach(gtk.Label("W"),2,3,4,5)
table.attach(gtk.Label("VA"),2,3,5,6)
table.attach(gtk.Label("Wh"),2,3,7,8)
table.attach(gtk.Label("*C"),2,3,9,10)
table.attach(gtk.Label("*C"),2,3,10,11)
table.attach(gtk.Label("Wh"),2,3,11,12)

#Insere os widgets na tabela com a posicao especificada. Quarta coluna
table.attach(gtk.Label("Leitor 2"),3,4,1,2)
table.attach(gtk.Label("Tensao:"),3,4,2,3)
table.attach(gtk.Label("Corrente:"),3,4,3,4)
table.attach(gtk.Label("Pot. Ativa:"),3,4,4,5)
table.attach(gtk.Label("Pot. Aparente:"),3,4,5,6)
table.attach(gtk.Label("Fator de pot.:"),3,4,6,7)
table.attach(gtk.Label("Consumo Atual:"),3,4,7,8)
table.attach(gtk.Label("Status:"),3,4,8,9)
table.attach(gtk.Label("Notificacao:"),3,4,11,12)
table.attach(grafico2, 3, 4, 12, 13)
table.attach(liga2, 3, 4, 13, 14)

#Insere os widgets na tabela com a posicao especificada. Quinta coluna
table.attach(self.tensao2,4,5,2,3)
table.attach(self.corrente2,4,5,3,4)
table.attach(self.ativa2,4,5,4,5)
table.attach(self.aparente2,4,5,5,6)
table.attach(self.fator2,4,5,6,7)
table.attach(self.consumo2,4,5,7,8)
table.attach(self.estado2,4,5,8,9)
table.attach(self.entradaConsumo2,4,5,11,12)

#Insere os widgets na tabela com a posicao especificada. Sexta coluna
table.attach(gtk.Label("V"),5,6,2,3)
table.attach(gtk.Label("A"),5,6,3,4)
table.attach(gtk.Label("W"),5,6,4,5)
table.attach(gtk.Label("VA"),5,6,5,6)
```

```
table.attach(gtk.Label("Wh"),5,6,7,8)
table.attach(gtk.Label("Wh"),5,6,11,12)

#Insere os widgets na tabela com a posicao especificada. Setima coluna
table.attach(gtk.Label("Leitor 3"),6,7,1,2)
table.attach(gtk.Label("Tensao:"),6,7,2,3)
table.attach(gtk.Label("Corrente:"),6,7,3,4)
table.attach(gtk.Label("Pot. Ativa:"),6,7,4,5)
table.attach(gtk.Label("Pot. Aparente:"),6,7,5,6)
table.attach(gtk.Label("Fator de pot.:"),6,7,6,7)
table.attach(gtk.Label("Consumo Atual:"),6,7,7,8)
table.attach(gtk.Label("Status:"),6,7,8,9)
table.attach(grafico3, 6, 7, 12, 13)
table.attach(liga3, 6, 7, 13, 14)

#Insere os widgets na tabela com a posicao especificada. Oitava coluna
table.attach(self.tensao3,7,8,2,3)
table.attach(self.corrente3,7,8,3,4)
table.attach(self.ativa3,7,8,4,5)
table.attach(self.aparente3,7,8,5,6)
table.attach(self.fator3,7,8,6,7)
table.attach(self.consumo3,7,8,7,8)
table.attach(self.estado3,7,8,8,9)

#Insere os widgets na tabela com a posicao especificada. Nona coluna

table.attach(gtk.Label("V"),8,9,2,3)
table.attach(gtk.Label("A"),8,9,3,4)
table.attach(gtk.Label("W"),8,9,4,5)
table.attach(gtk.Label("VA"),8,9,5,6)
table.attach(gtk.Label("Wh"),8,9,7,8)
table.attach(apaga, 8, 9, 12, 13)
table.attach(sair, 8, 9, 14, 15)
#########################################################

#Insere a tabela no final do widget container vertical

vbox.pack_end(table,True,True,0)
#Conecta o evento de expansao da janela a imagem de fundo
vbox.connect('expose-event', self.imagemFundo)
#Adiciona o widget conteiner vertical a janela
```

```
        self.add(vbox)
        #Apresenta a janela
        self.show_all()

        #Chama a funcao timeout de 1 em 1 segundo
        gobject.timeout_add(1000, self.timeout)

############## - Metodo de leitura e atualizacao - ###################
def timeout (self):
        #Chama os metodos
        self.verificaPIR ()
        self.verificaLM35 ()
        self.controleCarga ()

        #Caso para leitor 1
        if (self.controleLeitor==1):
                #Envia uma solicitacao de leitura
                leitor1.write("l")
        #Caso para leitor 2
        elif (self.controleLeitor==2):
                #Envia uma solicitacao de leitura
                leitor2.write("l")

        ############## - Leitura dos valores - ###################
        #Variavel de controle para garantir a consistencia dos dados
        controle=False
        #Recursao de leitura
        while controle != True:
                #Leitura do leitor 1
                if (self.controleLeitor==1):
                        #Armazena o dado lido da serial na variavel
                        lido= leitor1.readline()
                #Leitura do leitor 2
                elif (self.controleLeitor==2):
                        #Armazena o dado lido da serial na variavel
                        lido= leitor2.readline()
                #Remove line Breakers
                lido=lido.replace('\r\n','')
                #Quebra a string pelos separadores @ e cria uma lista de valores
                listaValores=lido.split('@')
```

```
		#Atribui os valores da lista em variaveis
		#Estrutura try sai com algum erro
		try:
			#Converte valor da lista em real
			v=float(listaValores[0])
			#Caso o valor seja muito baixo, descarta
			if (v<3):
				v=0;
			#Converte valor da lista em real
			c=float(listaValores[1])
			#Caso o valor seja muito baixo, descarta
			if (c<0.04):
				c=0;
			#Converte valor da lista em real
			p=float(listaValores[2])
			#Caso o valor seja muito baixo, descarta
			if (p<0.01):
				p=0;
			#Converte valor da lista em real
			s=float(listaValores[3])
			#Caso o valor seja muito baixo, descarta
			if (s<0.10):
				s=0;
			#Converte valor da lista em real
			f=float(listaValores[4])
			#Caso o valor seja muito baixo, descarta
			if (f<0.10):
				f=0;
			#Converte valor da lista em inteiro
			e=int(listaValores[5])
			#Define o controle como verdadeiro, sem erros
			controle=True

		#Caso algum erro ocorra, repete a leitura
		except Exception,e:
			pass

############## - Atualizacao dos dados - ###################
#Armazena o valor atual de data
now=datetime.now()
#Define o formato da data
```

```
                dataAtual=now.strftime("%d/%m/%Y        %H:%M:%S")
                #Atualiza o valor de data na interface
                self.data.set_text(str(dataAtual))
                #Caso para leitor 1
                if (self.controleLeitor==1):
                        #Armazena a soma de consumo      dividido pelo periodo de leitura em horas
                        consumoAtual1= round(self.somaConsumo1/1800,2)
                        #Caso o consumo seja maior que o definido
                        if (consumoAtual1 >= self.consumoDefinido1):
                                #Apresenta texto no terminal
                                print 'Consumo limite atingido no leitor 1'
                                #Som de notificacao
                                buzzer.toca(4)
                        #Adiciona o valor de potencia ativa ao somatorio de consumo
                        self.somaConsumo1=self.somaConsumo1+p
                        #Atualiza texto das labels da interface
                        self.tensao1.set_text(str(v))
                        self.corrente1.set_text(str(c))
                        self.ativa1.set_text(str(p))
                        self.aparente1.set_text(str(s))
                        self.fator1.set_text(str(f))
                        self.estado1.set_text(str(e))
                        self.consumo1.set_text(str(consumoAtual1))

                        #Abre o arquivo especificado no modo 'adicionar'
                        a = open('leitura1.dat', 'a')
                        #Apresenta texto no terminal
                        print "Leitor 1\t"
                        #Cria uma variavel com os valores lidos pelo leitor
                        data      =      "{}      {:.02f}      {:.02f}      {:.02f}      {:.02f}      {:.02f}      {}
{:.02f}\n".format(dataAtual,v,c,p,s,f,e, consumoAtual1 )
                        #Apresenta texto no terminal
                        print "{} {:.02f} {:.02f} {:.02f} {:.02f} {:.02f} {} {:.02f}\n".format(dataAtual,v,c,p,s,f,e,
consumoAtual1)
                        #Adiciona ao arquivo a variavel com os valores
                        a.write(data)
                        #Fecha o arquivo
                        a.close()

                        #Altera o leitor
                        self.controleLeitor=2
```

```
#Caso para leitor 2
elif (self.controleLeitor==2):
        #Armazena a soma de consumo      dividido pelo periodo de leitura em horas
        consumoAtual2= round(self.somaConsumo2/1800,2)
        #Caso o consumo seja maior que o definido
        if (consumoAtual2 >= self.consumoDefinido2):
                #Apresenta texto no terminal
                print 'Consumo limite atingido no leitor 2'
                #Som de notificacao
                buzzer.toca(4)
        #Leitura da porta I2C
        leituraI2C=i2cBus.read_byte(0x48)
        #Armazena o valor de temperatura dividido pela escala do conversor A/D
        self.temp=round((leituraI2C/2.5500),2)
        #Adiciona o valor de potencia ativa ao somatorio de consumo

        self.somaConsumo2=self.somaConsumo2+p
        #Atualiza texto das labels da interface
        self.tensao2.set_text(str(v))
        self.corrente2.set_text(str(c))
        self.ativa2.set_text(str(p))
        self.aparente2.set_text(str(s))
        self.fator2.set_text(str(f))
        self.estado2.set_text(str(e))
        self.consumo2.set_text(str(consumoAtual2))
        self.tempLabel.set_text(str(self.temp))

        #Abre o arquivo especificado no modo 'adicionar'
        a = open('leitura2.dat', 'a')
        #Apresenta texto no terminal
        print "Leitor 2\t"
        #Cria uma variavel com os valores lidos pelo leitor
        data = "{} {:.02f} {:.02f} {:.02f} {:.02f} {:.02f} {} {:.02f}
{:.02f}\n".format(dataAtual,v,c,p,s,f,e, consumoAtual2, self.temp )
        #Apresenta texto no terminal
        print "{} {:.02f} {:.02f} {:.02f} {:.02f} {:.02f} {} {:.02f}
{:.02f}\n".format(dataAtual,v,c,p,s,f,e, consumoAtual2, self.temp)
        #Adiciona ao arquivo a variavel com os valores
        a.write(data)
        #Fecha o arquivo
```

```
                a.close()

                #Altera o leitor
                self.controleLeitor=1

            return True

        ############### - Metodo de controle - ###################
        #Cria o metodo de controle
        def controleCarga       (self):
            #Caso para leitor 1
            if (self.controleLeitor==1):
                #Caso o estado da carga ou do sensor tenham mudado
                if (self.cargaAnterior1 != self.controleCarga1 or self.controleLM35 !=
self.LM35Anterior):
                    #Caso o estado da carga ou do sensor sejam 1
                    if (self.controleCarga1==1 or self.controleLM35==1):
                        #Envia um sinal de controle 1
                        leitor1.write("1")
                        #Apresenta texto no terminal
                        print 'Carga 1 acionada\n'
                        #Som de notificacao
                        buzzer.toca(3)
                    #Caso o estado da carga ou do sensor sejam 0
                    elif (self.controleCarga1==0 and self.controleLM35==0):
                        #Envia um sinal de controle 0
                        leitor1.write("0")
                        #Apresenta texto no terminal
                        print 'Carga 1 desligada\n'

                #Atribui o estado atual da carga a variavel
                self.cargaAnterior1 = self.controleCarga1
                #Atribui o estado atual do sensor a variavel
                self.LM35Anterior = self.controleLM35

            #Caso para leitor 2
            elif (self.controleLeitor==2):
                #Caso o estado da carga ou do sensor tenham mudado
                if (self.cargaAnterior2 != self.controleCarga2 or self.controlePIR != self.PIRAnterior):
                    #Caso o estado da carga ou do sensor sejam 1
                    if (self.controleCarga2==1 or self.controlePIR==1):
```

```
                                        #Envia um sinal de controle 1
                                        leitor2.write("1")
                                        #Apresenta texto no terminal
                                        print 'Carga 2 acionada\n'
                                        #Som de notificacao
                                        buzzer.toca(3)
                                #Caso o estado da carga ou do sensor sejam 0
                                elif (self.controleCarga2==0 and self.controlePIR==0):
                                        #Envia um sinal de controle 0
                                        leitor2.write("0")
                                        #Apresenta texto no terminal
                                        print 'Carga 2 desligada\n'

                        #Atribui o estado atual da carga a variavel cargaAnterior
                        self.cargaAnterior2 = self.controleCarga2
                        self.PIRAnterior = self.controlePIR

############### - Metodos da classe GUI - ####################
#Cria o metodo de definicao de imagem de fundo
def imagemFundo(self,widget, event):
        #Define uma variavel com o caminho da imagem
        path = '/home/pi/Desktop/ProjetoFinal/background.jpg'
        #Cria um objeto contendo uma imagem
        pixbuf = gtk.gdk.pixbuf_new_from_file(path)
        #Apresenta um retangulo com a imagem ao fundo
        widget.window.draw_pixbuf(widget.style.bg_gc[gtk.STATE_NORMAL], pixbuf, 0, 0, 0,0)

#Cria o metodo de confirmacao de saida
def sairConfirma(self, widget):
        #Cria uma janela de dialogo informativo com dois botoes
        dialog = gtk.MessageDialog(self, gtk.DIALOG_MODAL, gtk.MESSAGE_INFO,
                gtk.BUTTONS_YES_NO, "Tem certeza que deseja sair?")
        #Define texto adicional da janela de dialogo
        dialog.format_secondary_text(
                "O monitoramento sera intermopido na leitura atual")
        #Cria uma variavel com a resposta do dialogo
        response = dialog.run()
        #Caso a resposta seja sim
        if response == gtk.RESPONSE_YES:
                #Apresenta texto no terminal
                print("Monitoramento interrompido")
```

```
                #Limpa as portas utilizadas da GPIO
                GPIO.cleanup()
                #Envia um sinal de controle de desligamento
                leitor1.write("0")
                leitor2.write("0")
                #Fecha as portas seriais
                leitor1.close()
                leitor2.close()
                #Fecha a janela
                gtk.main_quit()
        #Caso contrario
        else:
                #Fecha a janela de dialogo
                dialog.destroy()

#Cria o metodo de verificacao do sensor de presenca
def verificaPIR (self):
        #Caso o pino esteja em HIGH
        if GPIO.input(pinoPIR):
                #Define o controle do sensor para 1
                self.controlePIR=1
        #Caso contrario
        else:
                #Define o controle do sensor para 0
                self.controlePIR=0

#Cria o metodo de verificacao do sensor de temperatura
def verificaLM35 (self):
        #Caso a temperatura esteja igual ou mais alta do que a definida
        if      (self.temp >= self.tempDefinida):
                #Define o controle do sensor para 1
                self.controleLM35=1
        #Caso contrario
        else:
                #Define o controle do sensor para 0
                self.controleLM35=0

#Cria o metodo de definicao de temperatura
def defineTemp (self, widget):
        #Define a temperatura pelo texto inserido
        self.tempDefinida=float(widget.get_text())
```

```
#Cria o metodo de definicao de consumo do leitor 1
def defineConsumo1 (self, widget):
        #Define o consumo pelo texto inserido
        self.consumoDefinido1=float(widget.get_text())

#Cria o metodo de definicao de consumo do leitor 2
def defineConsumo2 (self, widget):
        #Define o consumo pelo texto inserido
        self.consumoDefinido2=float(widget.get_text())

#Cria o metodo de criacao de grafico do leitor 1
def gnuplot1 (self,button):
        #Executa o comando de abertura do script do gnuplot pelo terminal,com persistencia
        proc = subprocess.Popen(['gnuplot','-p'], shell=False,stdin=subprocess.PIPE,)
        proc.communicate("""load \'leitura1.p\'
        """)

#Cria o metodo de criacao de grafico do leitor 2
def gnuplot2 (self,button):
        #Executa o comando de abertura do script do gnuplot pelo terminal,com persistencia
        proc = subprocess.Popen(['gnuplot','-p'], shell=False,stdin=subprocess.PIPE,)
        proc.communicate("""load \'leitura2.p\'
        """)

#Cria o metodo de eliminacao de dados armazenados
def apagaDados(self, button):
        #Abre o arquivo no modo escrever e fecha logo em seguida
        a = open('leitura1.dat', 'w').close()
        a = open('leitura2.dat', 'w').close()
        #Apresenta texto no terminal
        print "Dados apagados\n"
        #Som de notificacao
        buzzer.toca(2)

#Cria o metodo de mudanca de estado da carga do leitor 1
def acionaCarga1 (self,button):
        #Armazena o valor invertido da carga na variavel
        self.controleCarga1 = not self.controleCarga1

#Cria o metodo de mudanca de estado da carga do leitor 2
```

```
        def acionaCarga2 (self,button):
                #Armazena o valor invertido da carga na variavel
                self.controleCarga2 = not self.controleCarga2

###################### - Classe do Buzzer - ###################################
class Buzzer(object):
        #Metodo construtor responsavel por inicializar parametros
        def __init__(self):
                #Define variaveis com as frequencias das notas musicais basicas
                self.c = 261
                self.d = 294
                self.e = 329
                self.f = 349
                self.g = 392
                self.a = 440
                self.b = 493
                self.C = 523

        #Cria o metodo de definicao de parametros das notas
        def buzz(self,tom, duracao):
                #Caso o tom seja 0
                if(tom==0):
                        #Esperar o tempo da duracao
                        time.sleep(duracao)
                        return
                #Define o periodo do tom
                periodo = 1.0 / tom
                #Define o tempo de intervalo
                intervalo = periodo / 2
                #Define o numero de ciclos
                ciclos = int(duracao * tom)
                #Recursao para reproducao do som
                for i in range(ciclos):
                        #Aciona o buzzer
                        GPIO.output(pinoBuzzer, True)
                        #Aguarda o intervalo
                        time.sleep(intervalo)
                        #Desliga o buzzer
                        GPIO.output(pinoBuzzer, False)
                        #Aguarda o intervalo
                        time.sleep(intervalo)
```

```
#Cria o metodo de construcao das melodias
def toca(self, tom):
        #Define o contador com o valor inicial 0
        x=0
        #Caso o tom seja 1
        if(tom==1):
                #Define as notas da melodia

tons=[self.e,self.e,self.f,self.g,self.g,self.f,self.e,self.d,self.c,self.c,self.d,self.e,self.e,self.d,self.d]
                #Define a duracao das notas
                duracao=[0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1]
                #Pelo numero de notas na melodia
                for p in tons:
                        #Chama o metodo de reproducao
                        self.buzz(p, duracao[x])
                        #Aguarda pelo periodo da duracao
                        time.sleep(duracao[x] *0.5)
                        #Incrementa o contador
                        x+=1
        #Caso o tom seja 2
        elif(tom==2):
                #Define as notas da melodia
                tons=[self.e,self.d,self.c]
                #Define a duracao das notas
                duracao=[0.2,0.2,0.5]
                #Pelo numero de notas na melodia
                for p in tons:
                        #Chama o metodo de reproducao
                        self.buzz(p, duracao[x])
                        #Aguarda pelo periodo da duracao
                        time.sleep(duracao[x] *0.5)
                        #Incrementa o contador
                        x+=1
        #Caso o tom seja 3
        elif(tom==3):
                #Define as notas da melodia
                tons=[self.a,self.b]
                #Define a duracao das notas
                duracao=[0.2,0.5]
                #Pelo numero de notas na melodia
```

```
                    for p in tons:
                        #Chama o metodo de reproducao
                        self.buzz(p, duracao[x])
                        #Aguarda pelo periodo da duracao
                        time.sleep(duracao[x] *0.5)
                        #Incrementa o contador
                        x+=1
                #Caso o tom seja 4
                elif(tom==4):
                    #Define as notas da melodia
                    tons=[self.b]
                    #Define a duracao das notas
                    duracao=[0.2]
                    #Pelo numero de notas na melodia
                    for p in tons:
                        #Chama o metodo de reproducao
                        self.buzz(p, duracao[x])
                        #Aguarda pelo periodo da duracao
                        time.sleep(duracao[x] *0.5)
                        #Incrementa o contador
                        x+=1

#Executa o main
if __name__ == "__main__":
    #Apresenta texto no terminal
    print 'Inicializando...'
    #Cria uma instancia da classe buzzer
    buzzer = Buzzer()
    #Reproduz a melodia 1 da classe buzzer

    #Cria uma instancia da classe buzzer
    gui=GUI()
    #Recursao da interface grafica
    gtk.main()
```

APÊNDICE C – SCRIPT DO GNUPLOT

```
#Script leitura1.p

set   autoscale                 # Escala eixos automaticamente
unset log                       # Remove log-scaling
unset label                     # Remove etiquetas anteriores
set xtic auto                   # Define cada xtic automaticamente
set ytic auto                   # Define cada ytic automaticamente
set grid                        # Desenha linhas de grid
set time

# Definicoes de tempo
set xdata time
set timefmt "%d/%m/%Y\t%H:%M:%S"
#set format x "%d%m%Y\n%H%M%S"
set label "Data: %d/%m/%Y %H:%M:%S" at 1,1
set timestamp "%d/%m/%Y %H:%M:%S Guilherme Pimenta Barreto"

set title "Grafico de Potencias/ Consumo"
set xlabel "Tempo (dia/mes/ano hora:minuto:segundo)"
set ylabel "Potencia/ Consumo"
plot [:][:] 'leitura1.dat' using 1:5 title 'Potencia Ativa (W)' with lines , \
     'leitura1.dat' using 1:6 title 'Potencia Aparente (VA)' with lines , \
     'leitura1.dat' using 1:9 title 'Consumo (Wh)' with lines
set key below
set output
```