UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE SEMICONDUTORES INSTRUMENTOS E FOTÔNICA

ADELSON DUARTE DOS SANTOS

SISTEMA INTELIGENTE PARA MEDIÇÃO DO CONSUMO DE ENERGIA ELÉTRICA VIA POWER LINE COMMUNICATION

CAMPINAS
2018

ADELSON DUARTE DOS SANTOS

SISTEMA INTELIGENTE PARA MEDIÇÃO DO CONSUMO DE ENERGIA ELÉTRICA VIA *POWER LINE COMMUNICATION*

Dissertação de mestrado submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica na área de Eletrônica, Microeletrônica e Optoeletrônica.

Orientador: Prof. Dr. Elnatan Chagas Ferreira

Coorientador: Prof. Dr. Rodrigo Moreira Bacurau

Esse exemplar corresponde à versão final da dissertação defendida pelo aluno Adelson Duarte dos Santos, e orientada pelo Prof. Dr. Elnatan Chagas Ferreira

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

Sa59s

Santos, Adelson Duarte dos, 1992-
Sistema inteligente para medição de energia elétrica via Power Line Communication / Adelson Duarte dos Santos. – Campinas, SP : [s.n.], 2018.

Orientador: Elnatan Chagas Ferreira.
Coorientador: Rodrigo Moreira Bacurau.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Medidores elétricos. 2. Internet das coisas. 3. Hardware. I. Ferreira, Elnatan Chagas, 1955-. II. Bacurau, Rodrigo Moreira, 1988-. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Smart system for measuring electric consumption based on Power Line Communication
Palavras-chave em inglês:
Electrical meters
Internet of things
Hardware
Área de concentração: Eletrônica, Microeletrônica e Optoeletrônica
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Elnatan Chagas Ferreira [Orientador]
Leandro Tiago Manera
Alex Dante
Data de defesa: 31-08-2018
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Adelson Duarte dos Santos RA: 190710

Data da Defesa: 31 de agosto de 2018

Título da Tese: Sistema Inteligente para Medição do Consumo de Energia Elétrica via Power Line Communication

Prof. Dr. Elnatan Chagas Ferreira (Presidente, FEEC/UNICAMP)

Prof. Dr. Alex Dante (UFRJ)

Prof. Dr. Leandro Tiago Manera (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Agradecimentos

Agradeço aos meus pais, Adauto e Dalva, por todo o apoio e suporte fornecido durante a realização desse trabalho.

À minha companheira Jéssica Gonçalves Lobo pela paciência, amor, carinho e compreensão durante todo o desenvolvimento dessa pesquisa. Agradeço a minha cachorrinha, Preta, pelo carinho incondicional. Vocês foram fundamentais em todo esse trabalho.

Ao meu orientador Prof. Dr. Elnatan Chagas Ferreira, pela confiança, paciência e por todos os seus ensinamentos. Serei eternamente grato pela oportunidade que o senhor me forneceu de aprender um pouco de eletrônica.

Ao meu colega de laboratório e co-orientador Prof. Dr. Rodrigo Moreira Bacurau pela paciência e dedicação em sanar minhas dúvidas e por todos os seus ensinamentos.

Aos meus colegas de laboratório: Anderson Vedovetto e Marllon Schlischting pela amizade e companheirismo. Agradeço também ao professor Dr. Flávio José de Oliveira Morais por todo auxílio fornecido.

Aos colegas do LEPO, LSERF e DEB pela amizade e pelo empréstimo de equipamentos.

Aos meus amigos da FEI que, diretamente ou indiretamente, contribuíram com o desenvolvimento dessa pesquisa: Adilson Calixto, Felipe Hikichi, Jonatas de Cassio, Lucas Vieira, Lucas Molina, Rodrigo Carlos Mariani e Victor Hugo.

Aos meus amigos de longa data Carlos Henrique, Geovani Ventura, Guilherme Baldinotti, Patrick Lima, Leonardo Valvassori e Renato Vinicius.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), pelo financiamento.

"Se você vier me perguntar por onde andei
No tempo em que você sonhava
De olhos abertos lhe direi
Amigo, eu me desesperava."

(Belchior)

Resumo

Com o emergir do conceito IoT *(Internet of Things)*, aplicações de automação residencial, anteriormente conhecidas como domótica, ganharam maior visibilidade. Dentre as aplicações de automação residencial, o gerenciamento de recursos, como energia elétrica e água, vem cada dia ganhando mais importância. Nesse contexto, propôs-se neste trabalho o desenvolvimento de uma rede de medidores de energia inteligentes que permitam o controle e monitoramento de cargas residenciais em tempo real. Os módulos medidores foram construídos utilizando microcontroladores de baixo custo e consumo. A medição de corrente foi implementada utilizando resistor *shunt* e a de tensão através de divisor resistivo. O acionamento das cargas é controlado através de relés. A versão final do módulo medidor se mostrou capaz de medir energia elétrica com erros inferiores a 4 %. Foi desenvolvido um módulo mestre, em plataforma raspberry pi, responsável por solicitar os dados dos medidores, bem como enviá-los a um serviço web para acesso aos dados pela internet. Visando a construção de um sistema simples, de fácil instalação e com baixo custo, optou-se pela implementação do sistema de comunicação entre os módulos medidores através da rede elétrica (Power Line Communication - PLC). Como parte principal desse trabalho, foi proposto e construído um modem PLC de baixo custo para aplicações que requerem baixo fluxo de dados e alta latência, como monitoramento e controle de recursos em ambientes residenciais. O diferencial do modem PLC proposto é a utilização dos condutores de neutro e terra como meio de comunicação, eliminando a necessidade de circuitos de isolação complexos, e, consequentemente, diminuindo o custo do sistema. Outras decisões que contribuíram com a redução do custo do modem PLC foram: uso de protocolos e esquemas de modulação simples - não exigindo processamento digital do sinal tampouco filtros complexos; e a implementação do circuito eletrônico com componentes discretos de baixo custo: transistores, resistores, capacitores e indutores. Resultados experimentais mostraram que o modem PLC proposto é capaz de: transmitir dados a uma taxa de 9600 b/s com taxa de erro inferior a 10 % com ruído de densidade espectral de até 366,67 $\mu V_{RMS}/\sqrt{Hz}$; e transmitir dados a uma taxa de 19200 b/s com taxa de erro inferior a 10 % com ruído de densidade espectral de até 333,34 $\mu V_{RMS}/\sqrt{Hz}$.

Palavras-chave: IoT, PLC, medidor de energia.

Abstract

Due to the emergence of the IoT (Internet of Things) concept, home automation solutions earlier known as Domotics, have gained greater visibility. Among the home automation solutions, the management of resources, such as electricity and water, is becoming more important every day. In this context, it was proposed in this work the development of an intelligent energy meters network that allow the control and monitoring of residential loads in real time. The energy meters were developed using a low-cost and low-power microcontroller. Current sensing circuit was implemented using shunt resistor and voltage sensing by resistive divider. The loads are driven by relays. The final version of the energy meter can measure energy parameters with error rate less than 4 %. A master module has also been developed, in raspberry pi platform, responsible for requesting the data of the meters, as well as sending them to a web service for data access through the internet. In order to build a simple, low cost and easy to install system, we choose the implementation of energy meters communication system through the power lines (Power Line Communication - PLC). As a major part of this work, it was proposed and built a low-cost PLC modem for applications that requires low data rates, such as monitoring and control resources in residential environments. The main difference of the proposed PLC modem is the use of ground and neutral conductors as the communication channel, which eliminates the isolation circuits concern, and consequently, reduces the complexity and cost of the system. Other design decisions that contributed to the low-cost of the proposed PLC modem were: simple protocols and modulation techniques - not requiring digital signal processing or complex filters; and the implementation of the electronic circuit with inexpensive discrete components: transistors, resistors, capacitors and inductors. Experimental results showed that the proposed PLC modem is capable of: transmitting data at a rate of 9600 b/s with an error rate less than 10 % on a power line with a spectral density noise of 366,67 $\mu V_{RMS}/\sqrt{Hz}$; and transmitting data at a rate of 19200 b/s with an error rate less than 10 % on a power line with a spectral density noise of 333,34 $\mu V_{RMS}/\sqrt{Hz}$.

Keywords: IoT, PLC, energy meter.

Lista de Figuras

Lista de Abreviações

A/D	Analógico/ Digital
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
PBE	Programa Brasileiro de Etiquetagem
INMETRO	Instituto Nacional de Metrologia, Normalização e Qualidade Industrial
ACEEE	American Council for an Energy-Efficient Economy
D/A	Digital / Analógico
PLC	Power Line Communication
PCB	Printed Circuit Board
LISSE	Laboratório de Instrumentação, Sensores e Sistemas Embarcados
FEI	Centro Universitário da Fundação Educacional Inaciana "Padre Sabóia de Medeiros"
ESR	Equivalent Series Resistance
PWM	Pulse Width Modulation
RISC	Reduced Instruction Set Computer
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
THT	Through-Hole Technology
SMD	Surface Mounting Device
IDE	Integrated Development Environment
API	Application Programming Interface
CI	Circuito Integrado
FPGA	Field Programmable Gate Array
IoT	Internet of Things

NILM	Non-Intrusive Load Monitoring
ILM	Intrusive Load Monitoring
AES	Advanced Encryption Standard
WEB	World Wide Web
CENELEC	European Commitee for Electrotechnical Standardization
FCC	Federal Communications Commision
MBOK	m-ary bi-orthogonal keying
DSSS	Direct-sequence Spread Spectrum
CSS	Chirped spread spectrum
CDMA	Code Division Multiple Access
FSK	Frequency Shift Keying
ASK	Amplitude Shift Keying
OOK	On-off Keying
PSK	Phase Shift Keying
UART	Universal Asynchronous Receiver/Transmitter
EMI	Electromagnetic interference
PGA	Programmable Gain Amplifier
MCLK	Main Clock
CC	Corrente Contínua
SPST	Single Pole Single Throw
DCO	Digitally Controlled Oscillator
SMCLK	Sub-Main Clock
MQTT	Message Queuing Telemetry Transport
HTML	hypertext markup language
REST	Representational State Transfer

URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
M2M	Machine-to-Machine
ARM	Advanced RISC Machine
SDRAM	Synchronous dynamic random-access memory
HDMI	High-Definition Multimedia Interface
UNICAMP	Universidade Estadual de Campinas

Sumário

Capítulo 1

Introdução

Neste capítulo é apresentada uma breve introdução sobre o consumo de energia elétrica no Brasil e no mundo. São apresentados programas de conscientização do uso racional da energia elétrica realizados pelo governo brasileiro e a crescente proposta de sistemas inteligentes de medição de energia elétrica. Em seguida, é apresentado o objetivo deste trabalho.

1.1 Contextualização

O consumo de energia elétrica vem aumentando a cada ano. De 2006 a 2016, houve um aumento do consumo de 2,2 % ao ano, ao passo que, em 2016, foram gerados 24.816,4 TWh de energia elétrica em todo o mundo [1]. O aumento do consumo de energia elétrica é um assunto preocupante, uma vez que, em 2014, aproximadamente 70 % da produção de energia elétrica mundial foi produzida através da queima de combustíveis fósseis, sendo a queima responsável por lançar 13.625 milhões de toneladas de CO_2 na atmosfera [2].

Conforme dados do DataBank, o consumo de energia elétrica nacional per capita passou de 1.564,089 kWh em 1994 para 2.601,366 kWh em 2014, crescimento de 66 % em apenas 20 anos [3]. Esse aumento de demanda está relacionado com o aumento do PIB, uma vez que o consumo de energia é um dos principais indicadores do desenvolvimento econômico e do nível de qualidade de vida de qualquer sociedade [4].

A preocupação com o consumo de energia elétrica no Brasil teve início em meados dos anos 1980 com a criação do Programa Nacional de Conservação de Energia Elétrica (PROCEL) pelo Ministério de Minas e Energia com coordenação da Eletrobrás.

Esse programa teve como objetivo estimular o uso consciente de energia elétrica por meio de ações educativas e investimentos em equipamentos e instalações.

O Programa Brasileiro de Etiquetagem (PBE) criado em 1993 teve como objetivo fornecer informações sobre o consumo de energia elétrica dos equipamentos de modo a conscientizar os consumidores no momento da aquisição. Esse programa é resultado da colaboração da Eletrobrás em conjunto com o Instituto Nacional de Metrologia, Normalização e Qualidade Industrial (INMETRO) e ganhou grande importância após o racionamento de 2001, quando foram implementadas as quotas de consumo mensal com a criação da Lei 10.295 de 17 de outubro de 2001, conhecida como Lei da Eficiência Energética. Essa lei estabeleceu níveis máximos de consumo de energia de máquinas e aparelhos elétricos fabricados ou comercializados no país [4], [5].

As informações da Figura 1 apresentam a distribuição do consumo de eletricidade em diversos segmentos e sua projeção para 2026. Com exceção do setor industrial, os outros segmentos tendem a crescer na projeção realizada, sendo o setor residencial responsável por aproximadamente 30 % do consumo total de eletricidade do país e com projeção de crescimento de 3,9 % ao ano [6].

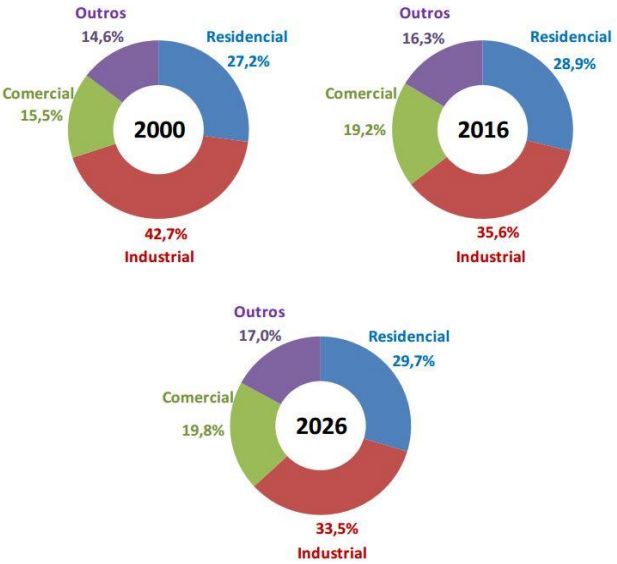


Figura 1. Consumo de eletricidade por segmento [6].

Dentre os equipamentos que constituem o consumo de energia elétrica residencial, estima-se que 3 % a 15 % do consumo de uma residência é oriundo do modo stand-by de equipamentos eletrônicos, segundo o estudo conduzido pelo Lawrence Berkeley National Laboratory [7]. O modo stand-by é definido como a energia consumida

por um equipamento eletrônico quando ele está no modo de espera. Por exemplo, um receptor de televisão mesmo desligado consome energia devido ao circuito para recebimento de sinais do controle remoto e para apresentar informação luminosa no visor, ou seja, gasta-se energia elétrica para manter circuitos receptores ativos e para alimentar alguma informação no visor do equipamento. Outros equipamentos como consoles de videogames, fornos microondas, modems de internet, impressoras, monitores, aparelhos de som, consomem energia no modo stand-by. Em 2013 o *European Union's Ecodesign Directive* estabeleceu que o consumo stand-by deve ser menor que 500 mW para aparelhos eletrônicos comercializados nos países membros. Essa ação resultou na redução da emissão de 39 milhões de toneladas de CO_2 e na economia de aproximadamente 35,5 TWh/ano [8].

O crescente interesse em *Internet of Things* (IoT) tem proporcionado o desenvolvimento de dispositivos inteligentes conectados à internet capazes de fornecer ao usuário informações em tempo real. Dispositivos IoT são formados por sensores, unidade de processamento e sistema de comunicação. Os sensores são responsáveis por realizar a aquisição de um fenômeno (temperatura, pressão, presença, etc.). A unidade de processamento, geralmente um microcontrolador, é responsável por processar o dado do sensor, controlar o dispositivo e aplicar o protocolo de comunicação adequado para transmitir a informação para outros dispositivos. Por fim, a informação adquirida é enviada para um servidor, o qual armazena as informações em um banco de dados, e possibilita que o usuário, com conexão à internet, acesse os dados remotamente através de uma interface de usuário. Trabalhos com sistemas de transportes inteligentes, casas inteligentes *(Smart Home*), rede elétrica inteligente *(Smart Grid),* iluminação pública, gerenciamento de resíduos e monitoramento ambiental têm sido desenvolvidos a partir do conceito emergente de IoT [9]–[11].

Medidores Inteligentes *(Smart Meters)* são dispositivos comumente utilizados em soluções de casas inteligentes. Esses dispositivos são capazes de obter dados de consumo de energia elétrica em tempo real e produzir informações de fácil interpretação para o usuário, cujo propósito é fornecer uma ferramenta para controle do consumo e, consequentemente, conscientizá-los sobre o impacto do seu comportamento no consumo de energia elétrica. Uma vez em posse dessa ferramenta, o consumidor tem a

possibilidade de conhecer seus hábitos de consumo e verificar quais ações podem ser tomadas para melhor aproveitamento da sua energia elétrica [12].

O Estudo dirigido pela *The American Council for an Energy-Efficient Economy* (ACEEE) indica que, em geral, sistemas de medição de energia elétrica em tempo real podem reduzir o consumo de energia elétrica em até 18 % [13]. O estudo de campo realizado na Irlanda em 2013, verificou que sistemas de medição de energia em tempo real resultam em economia de 2,9 % às famílias que os utilizam. Além disso, 48 % das famílias que utilizam sistemas de medição de energia em tempo real alegaram ter maior conhecimento sobre o consumo de energia geral da residência e 51 % das famílias declararam maior conhecimento sobre o consumo específico dos equipamentos de sua residência [14].

A Figura 2 apresenta algumas formas de feedback de sistemas de medição de energia elétrica em relação ao custo atrelado à tecnologia utilizada e à informação disponibilizada para o usuário. Ao contrário dos métodos convencionais de feedback, sistemas de monitoramento mais sofisticados necessitam de maior capacidade de processamento de dados e infraestrutura de comunicação para que a informação esteja disponível para o usuário em tempo real e de forma simples. O custo atrelado ao aumento da complexidade muitas vezes inviabiliza a implementação. Dessa forma, a viabilidade desses sistemas está em soluções que oferecem baixo custo de implementação e fácil interpretação para o usuário.

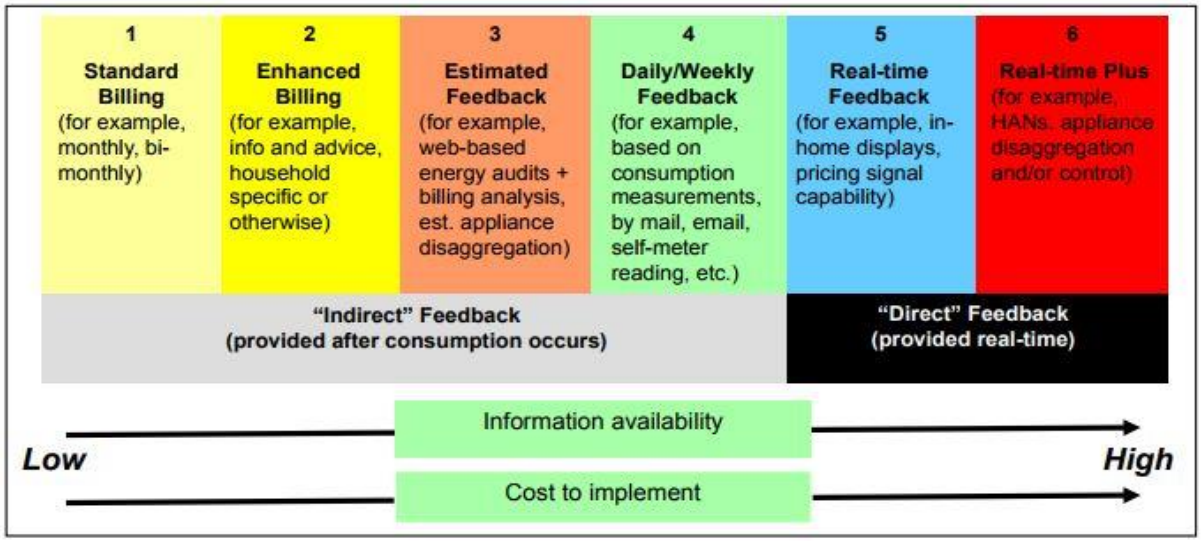


Figura 2. Diferentes sistemas de feedback [15].

Medidores de energia com comunicação sem fio (Wi-fi, *Zigbee*, Bluetooth) podem disponibilizar as informações do consumo de energia elétrica através de aplicações web, aplicativos de smartphones, ou hardware dedicado. Medidores de energia que utilizam comunicação cabeada como PLC e RS-485 podem, do mesmo modo, ser utilizados para monitoramento do consumo de energia de uma residência. Contudo, haverá a necessidade de um dos medidores ter a atribuição de *gateway* para realizar a interface da rede local (formada pelos medidores com comunicação cabeada) com a internet para disponibilização das informações através de sistemas web ou aplicativos de smartphones [16]. Os medidores de energia elétrica utilizados nesses sistemas inteligentes podem ser do tipo medidor geral de energia, geralmente localizado na entrada da caixa de distribuição, o qual realiza medição não intrusiva; ou medidores dedicados instalados nas cargas que se deseja monitorar, caracterizando-se como sistemas intrusivos.

Sistemas de medição não intrusivos, conhecidos como *Non-Intrusive Load Monitoring* (NILM), realizam a medição centralizada; apenas um medidor é necessário para o monitoramento de cargas e, portanto, são mais baratos. São geralmente instalados no quadro de distribuição e requerem a implementação de algoritmos de discriminação de cargas para avaliar o consumo individual das cargas.

Sistemas de medição intrusivos, conhecidos como *Intrusive Load Monitoring* (ILM), possibilitam a medição de cargas individuais, uma vez que cada carga monitorada possui um medidor conectado. Esses sistemas oferecem a vantagem de discriminação de carga de modo simples, maior precisão no monitoramento e possibilidade de controlar a carga. Estes medidores podem ser do tipo Tomada Inteligente *(Smart Plug)* como podem ser embutidos na própria carga [17]. Em um sistema ILM faz-se necessário a presença de diversos módulos para monitorar e controlar diferentes cargas. Em consequência disso, a principal desvantagem desse tipo de arquitetura é o custo vinculado à quantidade de módulos necessários e ao sistema de comunicação. Neste contexto, a utilização de sistemas ILM com enlace de comunicação via PLC (*Power Line Communication*) torna-se uma solução atrativa, uma vez que, intrisecamente, a solução PLC fornece uma redução de custo de infraestrutura e instalação, bastando apenas conectar os módulos à rede elétrica, sem necessidade de instalação de infraestrutura física adicional.

O PLC utiliza a rede elétrica existente para transmitir sinais digitais entre dispositivos conectados à rede elétrica. A grande vantagem dessa técnica é o baixo custo referente à infraestrutura de comunicação e instalação, uma vez que a rede elétrica já está presente nas construções. Em aplicações de casas inteligentes, o PLC possui a vantagem em relação às técnicas por rádio frequência de não sofrer atenuações devido a obstáculos (paredes, móveis, etc.) e, portanto, não necessitar de repetidores distribuídos pelo ambiente. Todavia, há desafios para a implementação do PLC devido à rede elétrica ser um ambiente hostil para transmissão de dados. Maiores detalhes sobre PLC serão apresentados no Capítulo 2 deste trabalho.

O desenvolvimento de sistemas ILM de baixo custo é de grande interesse para a adoção em aplicações de IoT que requerem controle e monitoramento de diferentes cargas em tempo real. Portanto, propõe-se nesse trabalho o desenvolvimento de módulos ILM de baixo custo com interface PLC que disponibilizem dados de consumo e permitam controle das cargas em tempo real.

1.2 Objetivos

O objetivo deste trabalho é desenvolver módulos medidores de energia elétrica, intrusivos e de baixo custo, para monitoramento e controle de cargas em ambientes residenciais. Os dispositivos devem ser capazes de viabilizar a comunicação pela rede elétrica (PLC) em diferentes instalações elétricas. Os dados de consumo devem ser disponibilizados para os usuários pela internet.

Para que esses objetivos sejam alcançados, será necessário:

- Projetar os circuitos de alimentação, aquisição, transmissão e recepção PLC.
- Projetar e fabricar Placas de Circuito Impresso (PCB).
- Desenvolver o *firmware* dos módulos medidores PLC.
- Desenvolver interface gráfica para testes e calibração.
- Desenvolver o software do sistema concentrador e interface gráfica.
- Validar o sistema em bancada.
- Validar o sistema em campo.

1.3 Organização do texto

O restante dessa dissertação está organizado da seguinte forma: no Capítulo 2 são apresentadas soluções de sistemas ILM para casas inteligentes e pesquisas relacionadas a modems PLC.

No Capítulo 3 é descrito o desenvolvimento dos módulos intrusivos. Nesse capítulo, são detalhados os componentes de hardware e firmware do sistema desenvolvido.

No Capítulo 4 é apresentado o desenvolvimento do sistema de concentrador de dados para validação dos módulos intrusivos desenvolvidos. Serão detalhados os componentes do sistema de aquisição de dados dos módulos medidores, a interface entre a rede local (PLC) e a internet, e a interface com usuário desenvolvida.

No Capítulo 5 são apresentados os resultados do sistema proposto.

No Capítulo 6 são apresentadas as conclusões e trabalhos futuros.

Capítulo 2

Desagregação do Consumo de Energia Elétrica Através de Rede de Sensores

Neste capítulo, são apresentadas pesquisas e soluções comerciais de sistemas de monitoramento de consumo de energia para aplicação em casas inteligentes. Em seguida, será apresentada a tecnologia PLC, suas características e desafios presentes na sua adoção. Por fim, implementações de modems PLC e suas características serão apresentadas.

2.1 Sistemas Intrusivos de Monitoramento de Energia - ILM

Sistemas ILM comerciais possuem um circuito de medição de energia elétrica e, na sua maioria, comunicação sem fio (Wi-fi, Bluetooth) e interface com usuário via web ou através de aplicativos em smartphones. Oferecem a capacidade de realizar o controle das cargas e monitoramento do consumo de energia elétrica em tempo real. Além disso, alguns desses sistemas comerciais são capazes de implementar algoritmos complexos de reconhecimento de padrões de consumo.

A tomada inteligente produzida pela empresa Zuli é apresentado na Figura 3. Ela possui 70 mm de comprimento, 41 mm de largura, 37 mm de altura e pesa 92 g. É compatível com redes com tensão nominal de 120 V_{RMS}, pode monitorar cargas resistivas de até 1800 W e capacitivas de até 240 W e possui a função de *dimmer* para cargas de até

240 W. A comunicação é realizada por bluetooth através de seu aplicativo para smartphones. São disponibilizadas para o usuário informações sobre o consumo de energia elétrica e a possibilidade de controle de cargas de forma instântanea ou por agendamento. Além disso, esse dispositivo possui sensores de presença que fornecem dados de comportamento de usuário e alimentam um algoritimo de reconhecimento de padrões cujo objetivo é ativar ou desativar cargas conforme a rotina do usuário. Contudo, essa funcionalidade requer a presença miníma de 3 dispositivos. A comunicação utilizada por este módulo limita o acesso aos dados, uma vez que a especificação bluetooth tem alcance máximo de 100 metros [18].

Figura 3. *Smartplug* Zuli [18]

O sistema ILM produzido pela Idevice é apresentado na Figura 4. Este dispositivo possui 42 mm de comprimento, 69 mm de largura, 40 mm de altura e pesa aproximadamente 93 g. É compatível com redes de 120 V_{RMS}, pode monitorar cargas resistivas de até 1800 W e indutivas de até 370 W. Permite o monitoramento de energia elétrica através de relatórios enviados ao usuário e controle de cargas a partir de aplicativo para smartphones. Esse aplicativo também permite o controle através de comandos de voz. A comunicação utilizada é Wi-Fi e, portanto, o acesso ao dispositivo pode ser realizado remotamente através da internet [19].

Figura 4. *Smartplug* Idevice [19]

A Figura 5 apresenta a tomada inteligente *DSP – W215 mydlink Wi-Fi Smart Plug* produzida pela D-LINK. Trata-se de um dispositivo de 61 mm de comprimento, 33 mm de largura e 90 mm de altura, alimentada com uma tensão de entrada de 100 V_{RMS} até 125 V_{RMS}, consumo máximo de 5 W. É capaz de monitorar cargas de até 1800 W/15 A e possui conexão Wi-Fi para controle e monitoramento por smartphone ou tablet. Dentre suas funcionalidades estão: Controle de carga remoto, relatório do consumo de energia elétrica, agendamento de controle de carga, proteção contra sobretemperatura [20].

Figura 5. *Smartplug* D-Link [20]

As características apresentadas pelos sistemas comerciais descritos são bastante semelhantes e percebe-se que a comunicação utiizada é predominantemente sem fio. Não foi encontrado um dispositivo ILM comercial que utilizasse comunicação PLC ou outra comunicação por cabos.

Os sistemas ILM também são amplamente explorados em pesquisas. Em [21], é apresentado o projeto de uma rede de medidores intrusivos de baixo consumo para monitoramento de cargas residenciais que possam transmitir informações de consumo de energia elétrica via radiofrequência. O microcontrolador utilizado possui um *transceiver* de rádio frequência que trabalha na banda de 868 MHz e capacidade de implementar criptografia AES de 128 bits, a qual foi utilizada para garantir a segurança da informação. O consumo máximo encontrado do módulo mestre é inferior a 25 mW com mais de 100 medidores na rede. O consumo do medidor avaliado é inferior a 20 mW. Nesse sistema, um *gateway* recebe os dados enviados pelos medidores e disponibiliza as informações através de uma interface web.

Em [22], é proposto um dispositivo para monitoramento e controle de cargas residenciais utilizando o protocolo de comunicação de baixo consumo *Zigbee*. As

informações aquisitadas pelo módulo medidor são visualizadas em uma aplicação para computador pessoal, sem possibilidade de acesso remoto pela internet.

Em [23], é proposto um medidor intrusivo que possui a capacidade de reconhecimento de carga através de sensores magnéticos conectados à carga monitorada. O sensor magnético fornece uma informação de 4 bits, a qual é lida pelo medidor. A partir de uma tabela embarcada no medidor, ele reconhece a carga que está sendo monitorada. É utilizada comunicação RS-485 para transmissão dos dados de consumo elétrico e do reconhecimento da carga para uma aplicação LabVIEW.

Em [24], é proposto um dispositivo que realiza a medição de energia elétrica e transfere as informações via PLC. Nessa implementação, o condicionamento de sinal para a inserção de dados na rede elétrica é realizado por um circuito integrado (CI) PLC, contudo, não há informações do mesmo. Os dados são recebidos por um *gateway* que disponibiliza a informação ao usuário através de uma interface WEB.

Nota-se nos trabalhos citados que diferentes soluções de comunicação foram utilizadas pelos sistemas ILM, tanto comunicações sem fio quanto cabeadas. Além disto, a preocupação em realizar feedback para o usuário é considerado em todos os trabalhos, acentuando a importância de uma boa interface com o usuário.

2.2 Power Line Communication

Como mencionado na seção anterior, a escolha do enlace de comunicação para a criação de uma rede de sensores é de extrema importância para o desempenho do sistema de medição e controle. Um protocolo de comunicação adequado garante a segurança e disponibilidade da informação. O custo e a eficiencia do sistema são critérios importantes na seleção de uma solução em detrimento das demais. A solução PLC residencial *(In-Home PLC)* torna-se atrativa uma vez que oferece a redução de custo de infraestrutura e, aplicando técnicas de modulação adequada, garante que os dados trafeguem com integridade pela rede elétrica. Além da aplicação residencial, o PLC pode ser utilizada em aplicações automotivas, navais, aeronáuticas e ferroviárias [25].

O PLC pode ser caracterizado pela sua largura de banda; são encontradas soluções de banda estreita *(narrowband)* e banda larga *(broadband)*. Os sistemas PLC *narrowband* utilizam frequências na ordem de centenas de kHz e estão presentes em implementações de baixo volume de dados, por exemplo: sistemas residenciais para

monitoramento e controle de temperatura, umidade, luminosidade etc. Da mesma forma, sistemas para monitoramento de consumo de energia elétrica também fazem o uso do PLC *narrowband* [26]. Os sistemas PLC *broadband* utilizam largura de banda maior para transmitissão de dados, trabalham com frequências de até 100 MHz, possibilitando taxas de transmissão na ordem de centenas de Mbps [25], [26]. Com essa taxa de transmissão, é possível transmitir informações que necessitam de grande volume de dados, tais como video e internet [27]. Altas taxas de transferência são possíveis graças à utilização de técnicas de modulações complexas baseadas em *spread-spectrum*, na qual a informação utiliza toda a largura de banda disponível. Contudo, a implementação dessas técnicas de modulação exige maior capacidade computacional e hardware mais complexo do que as soluções *narrowband*.

As frequências utilizadas por essas tecnologias são definidas por entidades regulatórias de várias partes do mundo, sendo as principais: *European Committee for Eletrotechnical Standardization* (CENELEC) na Europa e *Federal Communications Commision* (FCC) na América do Norte [25]. Outro padrão utilizado mundialmente é *Home Plug Power Alliance* [28]. No Brasil, a resolução normativa 375, de 25 de Agosto de 2009, regulamenta a utilização das instalações de distribuição elétrica para transporte de sinais [29], e a Resolução da Anatel 527, de 8 de Abril de 2009, aprova o regulamento sobre condições de uso da rede elétrica para transmitir sinais em banda larga [30]. Nenhuma das resoluções faz referência sobre condições de uso em ambientes residenciais e sobre a faixa de frequência utilizada.

A rede elétrica não foi concebida para tráfego de dados, o que traz desafios para a sua utilização como meio de comunicação. O ambiente ruidoso e com características diversas, tornam a caracterização e modelagem do canal de transmissão um desafio complexo para pesquisadores [26], [31]–[33]. As perturbações encontradas em um canal PLC podem ser representadas pela soma de diferentes tipos de ruídos [33]: ruído colorido com baixa densidade espectral e com frequência variável – causado pela soma de diversas fontes de ruído de baixa intensidade, podendo permanecer por minutos; ruído de banda estreita causado por sinais com modulação em amplitude oriundos de sinais de *broadcast* – sua amplitude pode variar ao decorrer do dia; ruído períodico impulsivo causado principalmente por fontes chaveadas; e ruídos impulsivos aperíodicos gerados por falhas na rede elétrica ou chaveamento de carga.

2.2.1 Modems PLC

Uma vez que as características do canal de transmissão são conhecidas, trabalhos foram propostos com o objetivo de superar os obstáculos inerentes ao canal e utilizar esse meio para transportar dados.

Em [34], foi desenvolvido um modem PLC *broadband* baseado em técnica de detecção de sinal adaptativa. O modem utiliza a modulação *chirped spread spectrum* (CSS) e possui sistema de retransmissão adaptativa baseada na seleção de frequências de portadora. Utilizando um conceito semelhante ao *direct digital synthesis* (DDS), o sistema possui uma tabela que contém dados que serão utilizados por um conversor D/A para formar a onda portadora com o objetivo de selecionar a melhor frequência para a transmissão do sinal. O sistema proposto é capaz de transmitir informações a uma distância de 50 metros com erro de aproximadamente 10 % em uma rede elétrica formada por eletrodomésticos comuns em residência (forno microondas, aspirador de pó, cargas capacitivas, secador elétrico).

A implementação em FPGA de um modem PLC *narrowband* proposto em [35] adota modulação digital de dois níveis com o objetivo de garantir o múltiplo acesso ao canal de transmissão (CDMA) e minimizar a interferência dos ruídos da rede elétrica. A primeira modulação utilizada é a *m-ary bi-orthogonal keying* (MBOK) utilizando *Walsh Codes* em conjunto com a *direct-sequence spread spectrum* (DSSS) e a segunda modulação usada é *frequency shift keying* (FSK).

A primeira modulação é responsável por garantir o múltiplo acesso ao canal de transmissão e espalhar a informação no espectro de frequência. A MBOK é uma técnica de modulação em quadratura em que é possível transmitir mais de 1 bit simultaneamente. *Walsh Codes* são códigos binários e ortogonais entre si. A DSSS é uma técnica de modulação utilizada para reduzir interferências no sinal, a qual consiste em modular a informação utilizando uma sequência de bit de curta duração. Dessa forma, foram utilizados 16 *Walsh Codes*, os quais foram indexados em 4 bits no diagrama de constelação da modulação em quadratura e são apresentados com 2 bits pela modulação MBOK. Cada *Walsh Code* é responsável por realizar a DSSS na informação e espalhá-la no espectro. Além disso, cada um dos 16 *Walsh Codes* garantem o mútiplo acesso ao canal de transmissão devido a sinais ortogonais não gerarem interferência entre si, portanto, até

16 comunicações simultâneas são permitidas nesse sistema. A segunda modulação é responsável por inserir a informação espalhada no espectro na rede elétrica. Foi utilizada a modulação FSK cujo valor lógico “0” é representado por 200 kHz e o valor lógico “1” é representado por 400 kHz. O sistema proposto permite taxa de transmissão de até 100 kbps.

A solução *narrowband* proposta em [27] é baseada em um protocolo de *smart home* chamado Insteon, o qual utiliza a modulação por deslocamento de fase (PSK). Esse protocolo realiza a transmissão sincronizada com o cruzamento de zero da rede elétrica, o que resulta em duas transmissões a cada 16,67 ms (60 Hz). Utilizam frames de 10 bytes e 24 bytes de dados de forma que necessitam de 6 a 13 cruzamentos de zero para enviar toda a informação. A máxima taxa de transmissão para esse protocolo é de 2880 bps.

Problemas com a entrega de pacote são solucionados por um sistema de retransmissão em que a mensagem é repetida, um número de vezes configurada em software, por todos os dispositivos conectados à rede. A repetição é sincronizada de modo a garantir que um sinal mais forte chegue no dispositivo destino. A rotina de repetição inicia 800 μs antes do cruzamento de zero e finaliza 1023 μs após o cruzamento de zero, desta forma a rotina de retransmissão não interfere na rotina principal de envio de mensagens.

O trabalho desenvolvido em [36] apresenta uma arquitetura de um modem PLC *broadband* implementado em FPGA que utiliza modulação MBOK com 16 *Walsh Codes*, semelhante ao trabalho [35]. Nessa implementação, os *Walsh Codes* são responsáveis por modular a informação e espalhá-la no espectro de frequência. Diferente da implementação de [35], não é adotado o segundo nível de modulação. A taxa de transmissão alcançada é de 128 kbps.

A maioria das soluções apresentadas utilizam complexas técnicas de processamento de sinais para superar a degradação do sinal oriundo da hostilidade do canal, sendo necessário empregar técnicas de processamento de sinais digitais para implementar o modem PLC.

Capítulo 3

Módulo Medidor PLC

Neste capítulo, é apresentado o projeto do módulo medidor de energia com interface PLC proposto nesse trabalho. Será apresentada uma visão geral do sistema e em seguida serão mostrados detalhes de cada um de seus componentes.

3.1 Descrição do Sistema

A rede de medidores de energia elétrica é composta por módulos medidores capazes de comunicar-se via PLC com o módulo mestre, o qual será apresentado em detalhes no Capítulo 4. Esses módulos utilizam sensores de tensão e corrente responsáveis pela aquisição dos dados utilizados pela unidade de processamento para o cálculo da energia consumida pela carga. A unidade de processamento também é responsável por controlar o acionamento da carga monitorada e controlar a transmissão e recepção de dados no modem PLC. A comunicação da unidade de processamento com o modem PLC é realizada via interface serial UART. A unidade de processamento também é responsável por enviar ao modem PLC um sinal quadrado (PWM com *duty-cycle* de 50%) utilizado para modulação. A Figura 6 apresenta o diagrama de blocos do medidor proposto.

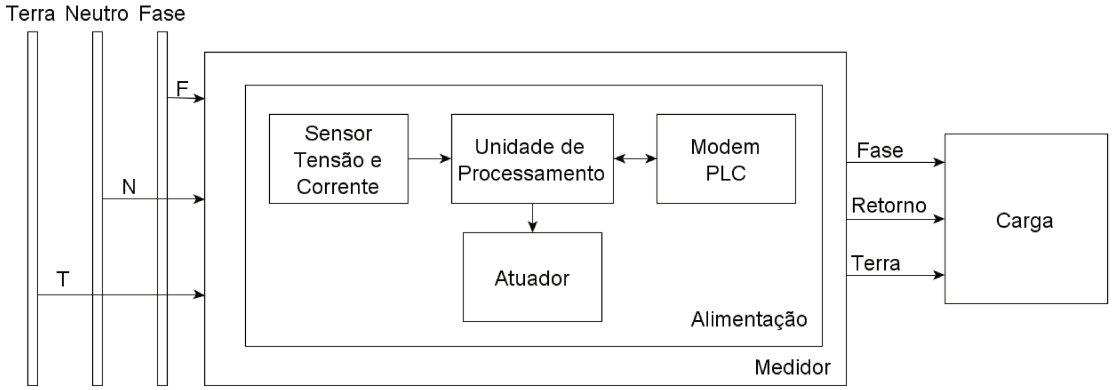


Figura 6. Diagrama de blocos do módulo medidor.

A unidade de processamento é baseada no microcontrolador MSP430AFE253 da Texas Instruments [37]. Este microcontrolador de baixo consumo possui uma unidade de processamento RISC de 16 bits com frequência de operação máxima de 12 MHz, 16 kB de memória *flash*, 512 B de memória SRAM, interfaces de comunicação UART e SPI, multiplicador em hardware de 16 bits, timer de 16 bits, 11 pinos de entrada e saída digital e três conversores A/D sigma-delta independentes de até 24 bits. Os conversores A/D possuem capacidade de sobreamostragem de 1024 vezes, ganho programável de 32 vezes e entradas diferenciais [37]. O esquemático da unidade de processamento é apresentado na Figura 7.

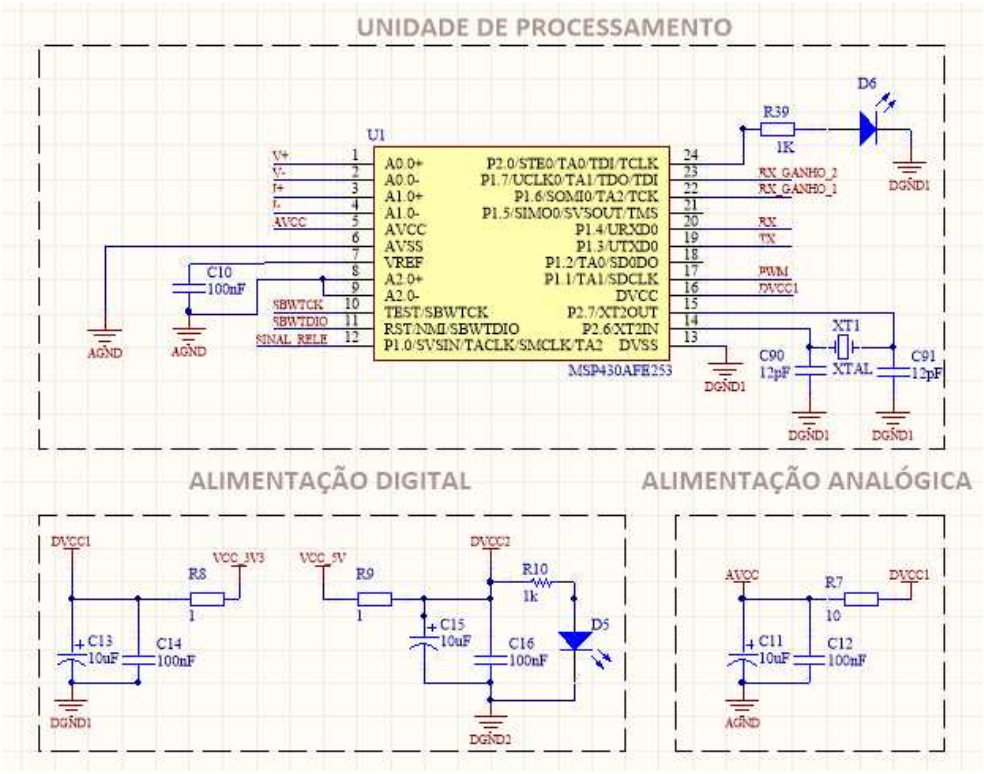


Figura 7. Diagrama esquemático da unidade de processamento.

Os pinos 1 e 2 (V+ e V-) e pinos 3 e 4 (I+ I-) são entradas diferenciais utilizadas pelo conversor A/D para receber amostras de tensão da rede e corrente da carga, respectivamente. O pino 12 (SINAL_RELÉ) é utilizado para controlar o relé. O pino 17 (PWM) é utilizado para gerar um sinal quadrado para o modem PLC. Os pinos 19 e 20 (TX e RX) são usados para enviar e receber informações do modem PLC pela UART. Os pinos 22 e 23 (RX_GANHO_1 e RX_GANHO_2) são utilizados para controlar a chave analógica TS5A2066, responsável pelo ajuste de ganho do estágio de amplificação do modem PLC [38]. Foram utilizadas duas alimentações: 5 V (DVCC2) e 3,3 V (DVCC1). A

primeira é responsável pela alimentação do modem PLC e do relé, a segunda alimenta o microcontrolador, chave analógica e parte do modem PLC. Maiores detalhes sobre o receptor PLC e a chave analógica serão apresentados na Seção 3.4.3 deste capítulo.

Para o controle de acionamento da carga foi utilizado um relé JQC-3F(T73) [39]. Este relé possui tensão de alimentação de 5 V e corrente de alimentação de aproximadamente 60 mA. Sua saída suporta tensões de até 125 V_{RMS} e correntes de até 10 A_{RMS}. O circuito para acionamento do relé consiste em um transistor NPN, resistor e diodo de proteção. Este circuito é apresentado na Figura 8.

A porta de saída do microcontrolador para o acionamento do relé é indicada pelo rótulo SINAL_RELÉ. Quando a porta SINAL_RELÉ é colocada em nível alto, ocorre a saturação do transistor Q7 e a fonte DVCC2 fornece a corrente necessária para acionar o relé, conectando as linhas SHUNT e RETORNO. Quando conectadas, as linhas SHUNT e RETORNO fecham o circuito que conecta a carga ao neutro da instalação. A corrente que flui por essas linhas é obtida pelo circuito de medição baseado em resistor *shunt*, conforme será discutido na Seção 3.3 deste capítulo.

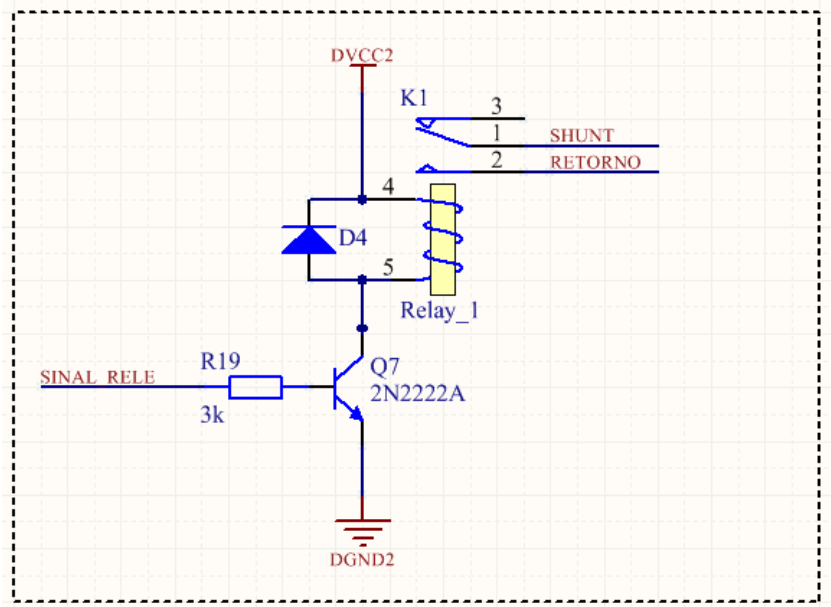


Figura 8. Circuito de acionamento de relé

3.2 Circuito de Alimentação

Para a alimentação de todo o circuito, foi desenvolvida uma fonte chaveada de topologia *flyback*, não isolada, com potência de 2,5 W, saída de 5 V e 500 mA, utilizando o CI UCC28910 da Texas Instruments [40]. O CI UCC28910 possui chaveamento por MOSFET, o qual suporta tensões de até 700 V. Ele é utilizado em topologias *flyback*, com frequência de chaveamento máxima de 115 kHz. Possui proteções contra aquecimento, redução da tensão da rede *(Brownout)* e sobretensão da saída. Além disso, oferece regulação de tensão e corrente com ± 5 % de acurácia sem o uso de foto-acopladores.

O processo de feedback é realizado através de um método de controle chamado *Primary – Side Regulation*. Esse método consiste em utilizar um terceiro enrolamento do transformador, chamado de enrolamento auxiliar, para sentir a tensão de saída através da relação de espira entre o enrolamento secundário e auxiliar. Essa informação é enviada para a malha de controle do CI UCC28910, garantindo controle preciso de tensão e corrente. Esse CI dispõe de algoritmos de controle para inicializar o processo de chaveamento, além de controle de frequência de chaveamento e modulação da corrente no circuito primiário com objetivo de atender às especificações EMI vigentes. Atua no modo descontínuo com objetivo de reduzir as perdas por chaveamento. De acordo com a condição de carga, o controlador pode atuar no modo *tensão constante:* quando é necessário controlar a tensão de saída; e no modo *corrente constante:* quando é necessário controlar a corrente de saída. Suas principais aplicações são em tomadas inteligentes, medidores de energia e carregadores de celular [40].

Em conjunto com o UCC28910, o regulador linear *Ultra-Low Dropout Voltage* TPS73633 [41] da Texas Instruments foi utilizado com o objetivo de regular a alimentação em 3,3 V.

3.2.1 Topologia *Flyback*

A topologia *flyback* é uma das configurações utilizadas em fontes chaveadas isoladas. É derivada da topologia *buck-boost*, conforme apresentado na Figura 9 e Figura 10. O indutor utilizado na configuração *buck-boost* é substituido por um transformador de alta frequência, também conhecido como *flyback transformer*, o qual possui polaridade invertida para garantir saída positiva no secundário, e relação de transformação

apropriada para contribuir para a performance da conversão. O uso desta topologia também oferece a vantagem de não utilizar indutores no filtro de saída devido à presença do transformador de alta frequência. Devido à isolação provida pelo transformador, esta topologia permite a implementação de diversas saídas no secundário, como apresentado na Figura 11. Apesar de se tratar de uma topologia para fontes isoladas, esta topologia foi implementada de forma não isolada devido à limitação de potência encontrada na utilização deste CI em topologia *buck*, conforme discutido em [42].

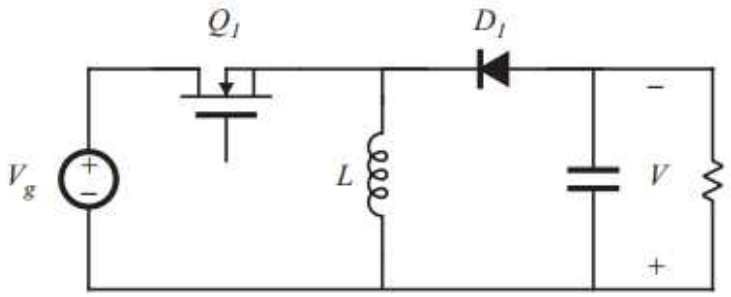


Figura 9. Topologia *buck-boost* [43].

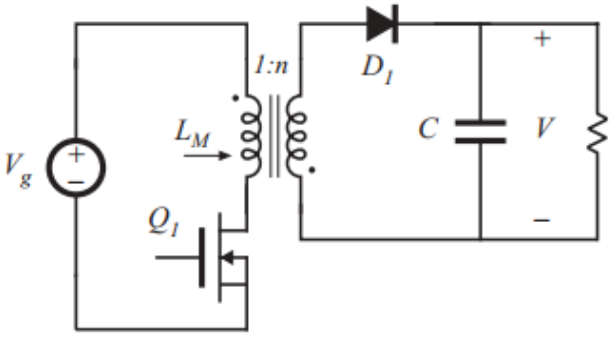


Figura 10. Topologia *flyback* [43].

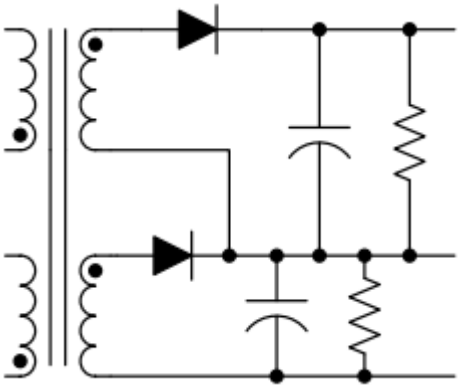


Figura 11. Topologia *flyback* com múltiplas saídas em configuração de minimização de problema de regulação para saída projetada *(cross-regulation)* [40]

3.2.1.1 Análise da Topologia *Flyback* em Modo Descontínuo

Como mencionado anteriormente, o CI utilizado trabalha no modo descontínuo. Este modo de operação é usualmente utilizado para projetos com potência inferior a 60 W. As principais características desse modo são [44]:

- Transformadores simples: Uma vez que neste modo de operação a energia armazenada nos enrolamentos do transformador é pequena, transformadores com baixa indutância são utilizados. A necessidade de poucas espiras dimiui as perdas nos enrolamentos. Essas características contribuem para a redução do tamanho e do custo do transformador.
- Diodo de retificação: A velocidade do diodo de retificação não é tão crítica, uma vez que nesse modo o retificador de saída está operando com corrente zero imediatamente antes de se tornar polarizado inversamente.
- MOSFET: O fechamento da chave ocorre quando a corrente do primário é igual à zero, portanto o tempo de fechamento não é um parametro crítico para o projeto. Esta característica também resulta na redução de EMI e na redução de perdas por chaveamento.

Em contrapartida, esse modo apresenta as seguintes desvantagens [44]:

- Corrente de pico: A bobina do primário, o MOSFET e o diodo de retificação estão sujeitos a valores de corrente de pico elevada. Portanto, a seleção desses componentes deve levar em consideração essa característica do circuito.
- Regulagem para múltiplas saídas: O fluxo magnético e o valor da indutância parasita são altos neste modo, resultando em problemas de regulação de múltiplas saídas. O circuito que minimiza este problema foi apresentado na Figura 10.
- Capacitor de saída: Altos valores de *ripple* de corrente requerem capacitores com ESR baixos. Capacitâncias elevadas são utilizadas para alcançar ESR adequados.

Para efeito de análise, vamos considerar o comportamento desse circuito em três etapas: chave fechada (Q-ON), chave aberta (Q-OFF) e modo descontínuo (MD).

A Figura 12 apresenta o circuito equivalente da topologia *flyback* quando o MOSFET está conduzindo (Q-ON). A corrente I_P oriunda da fonte V_g circula pela bobina primária do transformador L_P e pelo MOSFET a uma taxa descrita por:

$$\frac{dI_p}{dt} = \frac{V_g}{L_p}. \tag{1}$$

A energia armazenada W pelo indutor no período Q-ON é dada por:

$$W = \frac{1}{2} L_p \, {I_{P_{Pico}}}^2 . \tag{2}$$

A corrente de pico do primário $I_{P_{Pico}}$ é encontrada aplicando o limite na Equação (1) para $dt \rightarrow T_{Q-on}$, resultando em:

$$I_{P_{PICO}} = \frac{V_g T_{Q-ON}}{L_P}, \tag{3}$$

em que T_{Q-ON} é o tempo em que o MOSFET está acionado.

No estado estacionário, a tensão do indutor é aproximadamente V_g, uma vez que a tensão entre os terminais do MOSFET é próxima de zero.

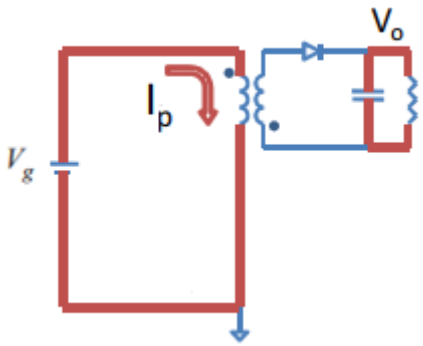


Figura 12. Circuito equivalente da topologia *flyback* durante a condução do MOSFET (Q-ON) [45].

Na Figura 13 é apresentado o circuito equivalente da topologia *flyback* quando o MOSFET está desligado (Q-OFF). Uma diferença de potencial V_s surge no enrolamento secundário com valor definido pela relação de espiras N e com sentido oposto a V_g, gerando passagem de corrente I_s no secundário do transformador. Esta corrente é responsável por carregar o capacitor e alimentar a carga. A energia armazenada no

primário do transformador durante o período Q-ON, é transferida, através das bobinas do transformador, para o filtro e para a carga.

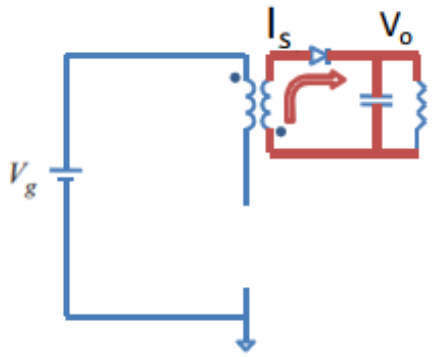


Figura 13. Circuito equivalente da topologia *flyback* durante a não condução do MOSFET (Q-OFF) [45].

Na Figura 14 é apresentado o circuito equivalente no modo de descontinuidade (MD). No momento em que a corrente do secundário é zero, o sistema entra em modo de descontínuidade, isto é, $V_S = 0$, $I_S = 0$, $I_P = 0$ e $W = 0$. Neste cenário, o capacitor é responsável por alimentar a carga até o momento em que a chave é fechada novamente (Q-ON).

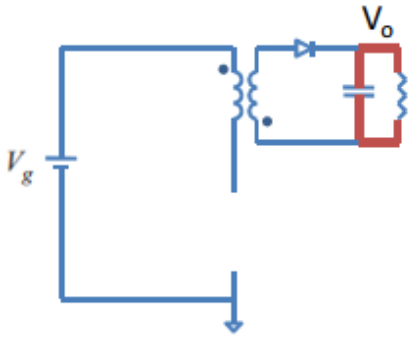


Figura 14. Circuito equivalente da topologia *flyback* no modo descontínuo (MD) [45]

A potência média entregue à carga é dada por [44]:

$$P = \frac{W}{T} = \frac{L_P \, {I_{P_{Pico}}}^2}{2T}, \tag{4}$$

em que, T é o período de chaveamento.

A potência dissipada na carga pode ser escrita como:

$$P = \frac{{V_0}^2}{R}. \tag{5}$$

Substituindo as equações (3) e (5) na Equação (4):

$$\frac{V_0}{V_g} = \frac{T_{Q-on}}{T}\sqrt{\frac{RT}{2L_P}}, \tag{6}$$

onde a relação $\frac{T_{Q-on}}{T}$ é o duty cycle D do sistema.

Observando a Equação (6), nota-se que no modo descontínuo a tensão de saída V_0 possui dependência dos parâmetros de circuito N, L_S e T; e do ponto de operação D, V_g e R [45].

3.2.2 Projeto da Fonte Chaveada

Na seção anterior, foram apresentados os conceitos fundamentais de funcionamento de uma fonte chaveada na topologia proposta. O entendimento do princípio de funcionamento desse circuito é de grande importância para melhor compreensão dos parâmetros de dimensionamento da fonte construída.

O projeto da fonte utilizada foi baseado no *Application Note* do UCC28910, presente no *datasheet* do CI [40], o qual apresenta os cálculos de dimensionamento dos componentes necessários à fonte chaveada. Além disso, fornece orientação para um bom roteamento de PCB de modo a evitar interferências em outros circuitos. Na Figura 15 é apresentado o diagrama esquemático da fonte desenvolvida.

Apesar da topologia *flyback* prover isolação galvânica entre primário e secundário, foi necessário romper essa isolação, conectando a referência do primário com a referência do secundário através do resistor R00 de aproximadamente 0 Ω, apresentado na Figura 15. A primeira motivação para realizar este rompimento foi permitir a utilização de um resistor *shunt* para medição de corrente. Essa escolha está relacionada à redução de custo do sistema, pois, transformador de corrente, ou sensor de efeito *Hall*, são consideravelmente mais caros do que resistores *shunt*. Ademais, o CI utilizado não permite utilizar topologia *buck* para tensões de saída inferiores a 7,5 V, conforme apresentado em [42]. Mais detalhes sobre o circuito de medição de corrente serão apresentados na Seção 3.3 desse capítulo.

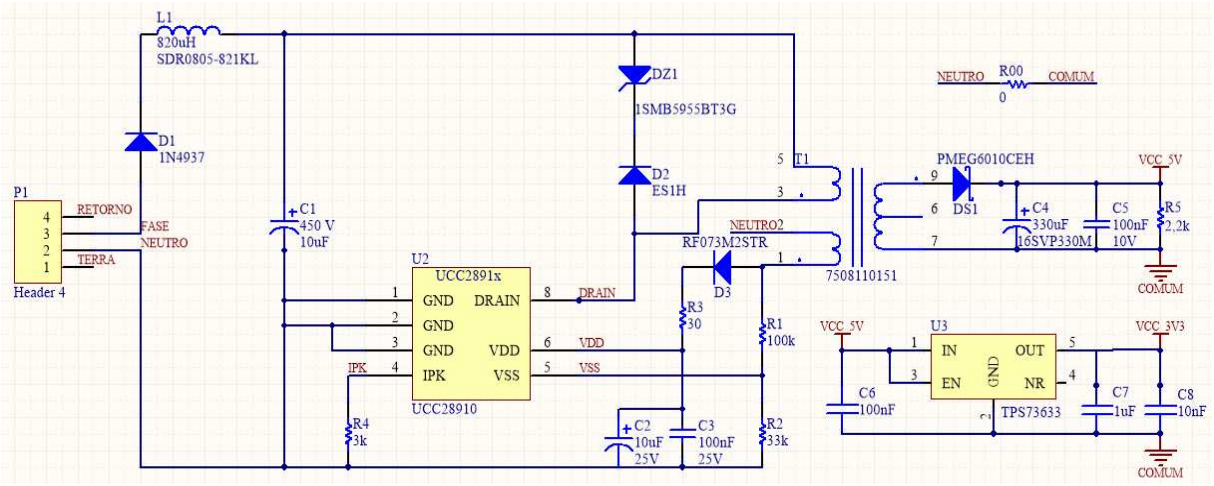


Figura 15. Diagrama esquemático da fonte projetada.

3.3 Sensores de Tensão e Corrente

O primeiro passo para aquisição da tensão da rede, é reduzí-la para valores compatíveis com a entrada do conversor A/D utilizado. Considerando que o valor máximo de tensão do conversor A/D do microcontrolador utilizado (MSP430AFE253) é de 353,55 mV_{RMS} (500 mV_{PICO}) [37], e que o circuito deve operar em redes com tensão nominal de 127 V_{RMS}, foi implementado um divisor resistivo composto por resistores de 180 kΩ e 1,5 kΩ. Dessa forma, o divisor resistivo terá uma tensão nominal de saída de 352 mV_{RMS} (497 mV_{PICO}). O conversor A/D utilizado possui 16 bits efetivos e o canal de leitura de tensão foi configurado para trabalhar com fundo de escala de 500 mV_{PICO} (PGA = 1), permitindo a medição da tensão da rede com resolução de 3,88 mV_{RMS}. O circuito de condicionamento de sinal de tensão é mostrado na Figura 16.

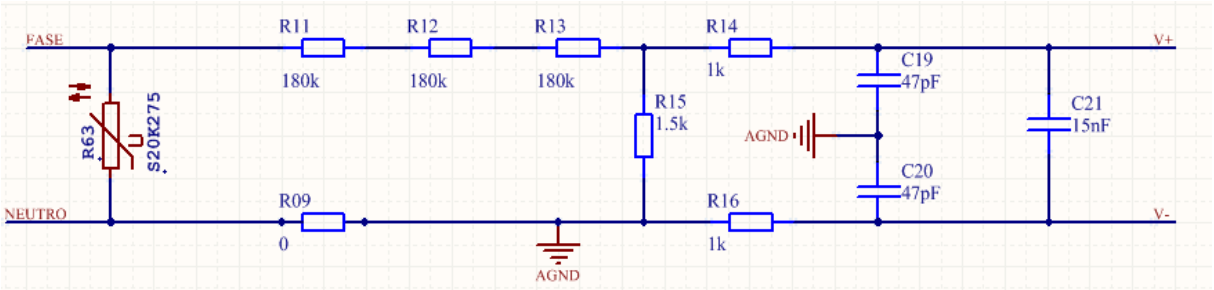


Figura 16. Circuito de condicionamento de tensão e filtro *anti-aliasing*.

O sinal de corrente é obtido através da diferença de potêncial gerada pela passagem de corrente da carga por um resistor *shunt* de 5 mΩ e 2 W [46]. A especificação do *shunt* limita a passagem de corrente em até 14 A_{RMS} (20 A_{PICO}). Utilizando o mesmo conversor A/D de 16 bits efetivos, com o canal de leitura de corrente configurado para

trabalhar com fundo de escala de 125 mV_{PICO} (PGA = 4), é possível medir correntes de até 17,68 A_{RMS} (25 A_{PICO}) com resolução de 539,48 μA_{RMS}. O circuito de condicionamento de sinal de corrente é mostrado na Figura 17.

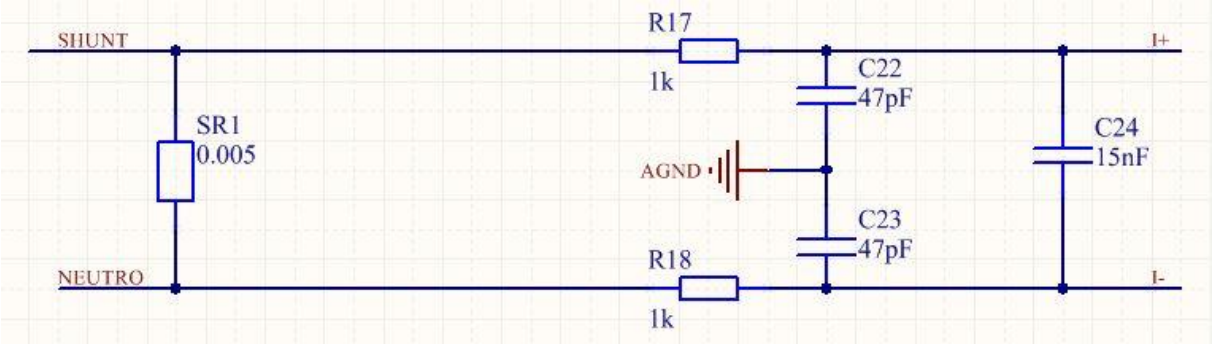


Figura 17. Cicuito de condicionamento de corrente e filtro *anti-aliasing*.

Tanto o circuito de condicionamento de sinal de tensão quanto o de corrente, possui filtro *anti-aliasing,* apresentados nas Figura 16 e Figura 17. Devido à alta taxa de amostragem utilizada pelo conversor sigma-delta, na ordem de 1 MS/s, o filtro *anti-aliasing* pode ser reduzido a um simples circuito RC de primeira ordem. A frequência de corte do filtro utilizado é de 10 kHz. A resposta em frequência do filtro é apresentada na Figura 18.

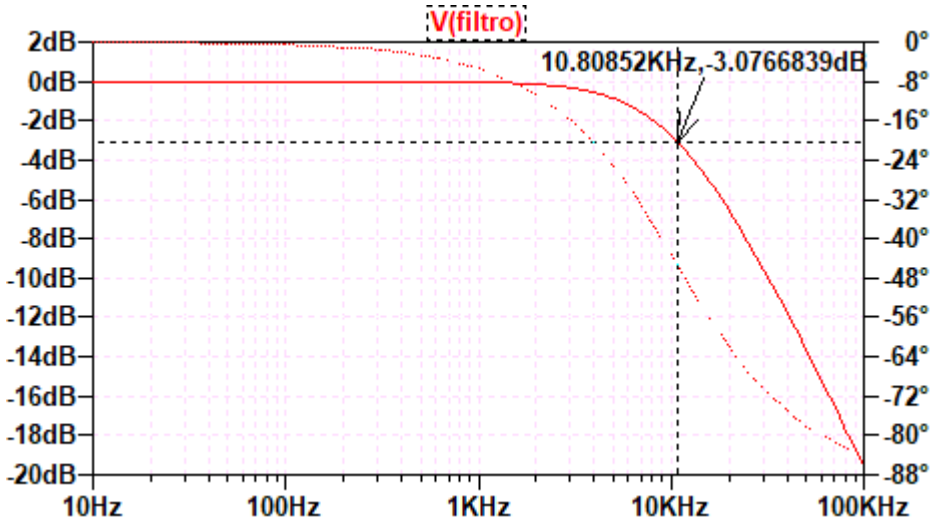


Figura 18. Resposta em frequência do filtro *anti-aliasing*.

Para realizar a aquisição do sinal de tensão e corrente para a conversão A/D, foi necessário receber 65 amostras do ciclo da rede. Esse valor é resultado da frequência utilizada pelo conversor A/D. O conversor A/D é configurado para utilizar a fonte de clock principal do microcontrolador (MCLK), cujo valor é 12 MHz. Esse valor é dividido por 6

no bloco de conversão A/D, resultando na taxa de amostragem efetiva de 1 MS/s. Devido ao processo de sobreamostragem utilizada por conversores A/D sigma-delta, foi configurado uma taxa de sobreamostragem de 256 vezes. Portanto, a taxa de amostragem efetiva de 1 MS/s resulta em uma taxa de amostragem de aproximadamente 3906 S/s (1 MS/s / 256). Dessa forma, para a leitura de sinais da rede elétrica em 60 Hz, são necessários obter aproximadamente 65 amostras por ciclo de rede (3906 S/s / 60 Hz).

3.4 Modem PLC

Diferentemente da maioria dos modems PLC, que utilizam os condutores de neutro e fase para transmissão de dados, o sistema desenvolvido utiliza os condutores de neutro e terra. Dessa forma, o circuito de isolação se tornou simples, reduzindo seu custo. O modem PLC é responsável por modular o sinal a ser transmitido e acoplá-lo à rede elétrica, bem como, aquisitar os sinais recebidos e decodificá-los. No modem desenvolvido, foram aplicadas técnicas simples de modulação, filtragem, amplificação e demodulação no sinal.

O modem proposto é capaz de transmitir e receber informações utilizando um único canal, sendo possível visualizar a informação transmitida (eco) no circuito de recepção. Essa característica é utilizada para realizar calibração do sistema de transmissão, como será discutido na Seção 3.6 A informação (sinal modulante) é modulada por um sinal quadrado de alta frequência (portadora) e inserida entre terra e neutro através de um filtro passa–altas. Na Figura 19 é apresentado o diagrama de blocos do sistema proposto.

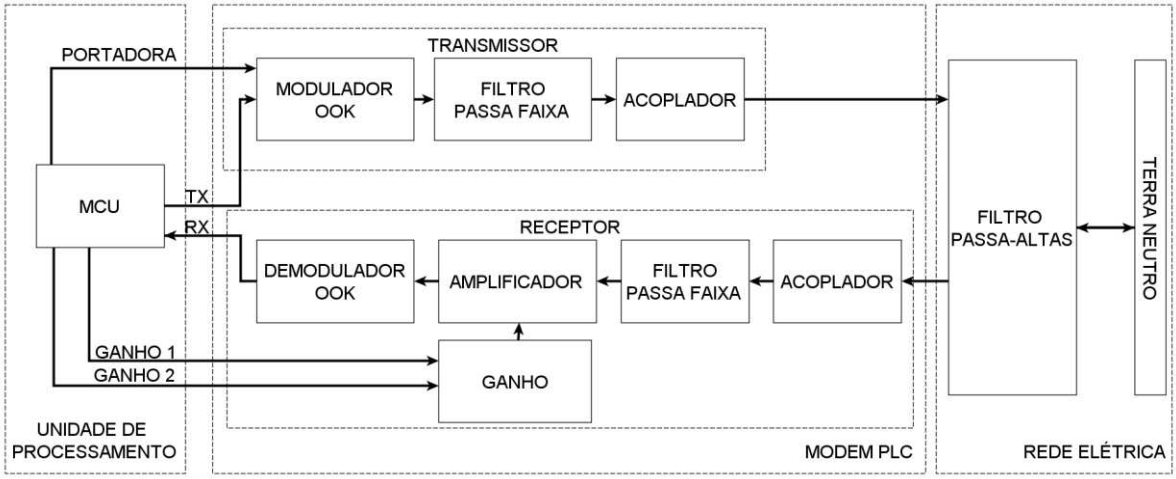


Figura 19. Diagrama de blocos do sistema proposto.

3.4.1 Transmissor

O transmissor PLC proposto é formado por um modulador OOK (*on-off keying*), filtro passa-faixa e acoplador, conforme apresentado na Figura 20.

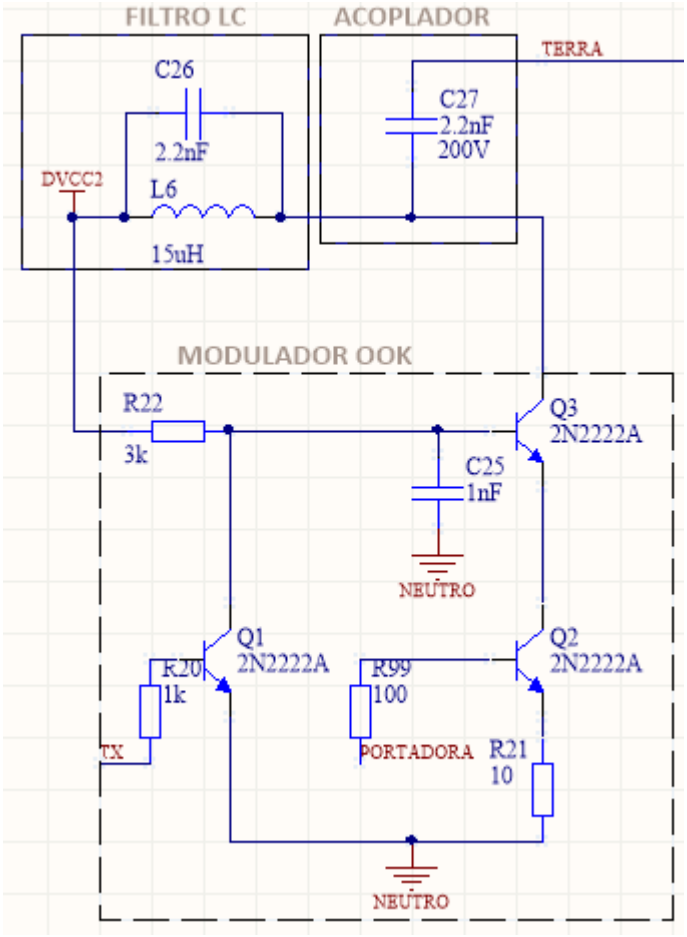


Figura 20. Circuito transmissor do PLC proposto.

No circuito de transmissão proposto, o sinal modulado possui fase deslocada em 180° em relação ao sinal modulante. Este comportamento pode ser entendido pela análise do circuito da Figura 19. Para o sinal modulante em nível lógico alto, o transistor Q1 estará saturado, e, consequentemente, Q3 estará em corte e Q2 estará flutuando, pois seu coletor estará aberto. Portanto, o chaveamento do transistor Q2 pela portadora não terá influência na modulação do sinal. No momento em que o sinal modulante está em nível lógico baixo, o transistor Q1 estará em corte, e, consequentemente, Q3 estará saturado devido à passagem de corrente oriunda da fonte DVCC2, e Q2 será chaveado pela portadora. Portanto, haverá portadora para o nível lógico baixo do dado digital e não haverá portadora para o nível alto do lado digital.

A modulação OOK é uma forma simplificada da modulação ASK (*amplitude-shift keying*) e consiste na representação do sinal digital em função da presença ou ausência da portadora. Por exemplo, o nivel lógico alto pode ser representado pela ausência de portadora enquanto o nível lógico baixo seria representado pela presença da portadora. Essa modulação é facilmente implementada em hardware com poucos componentes: resistores, capacitores e transistores.

Após a modulação, a informação passa por uma etapa de filtragem. A implementação desse filtro é realizada com um circuito LC com frequência de ressonância próxima à frequência da portadora do sinal. O objetivo desse filtro é sintonizar a frequência da portadora, limitando a banda do sistema. A frequência de ressonância f do filtro LC é dada por:

$$f = \frac{1}{2\pi\sqrt{LC}}. \tag{7}$$

A frequência de ressonância utilizada no circuito de transmissão proposto é de aproximadamente 900 kHz.

Por fim, a informação modulada é inserida no condutor terra através de um capacitor. Além de desacoplar o circuito de transmissão com o terra da instalação elétrica, a presença deste capacitor contribui com o filtro LC de maneira a atenuar sinais de baixa frequência.

3.4.2 Filtro passa-altas terra neutro

A diferença de potêncial entre terra e neutro é limitada por um filtro passa-altas, mostrado na Figura 21. Visto que não há cargas entre neutro e terra, o filtro passa-altas oferece alta impedância para a frequência da rede e baixa impedância para o sinal modulado. Dessa forma, o sinal modulado é introduzido adequadamente nos condutores terra e neutro.

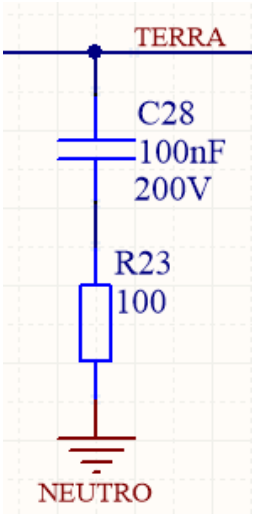


Figura 21. Filtro passa-altas terra neutro.

3.4.3 Simulação do Transmissor

Utilizando a ferramenta de simulação LTSpice, foi possível simular o circuito transmissor A Figura 22 apresenta as formas de onda observadas em cada ponto do circuito.

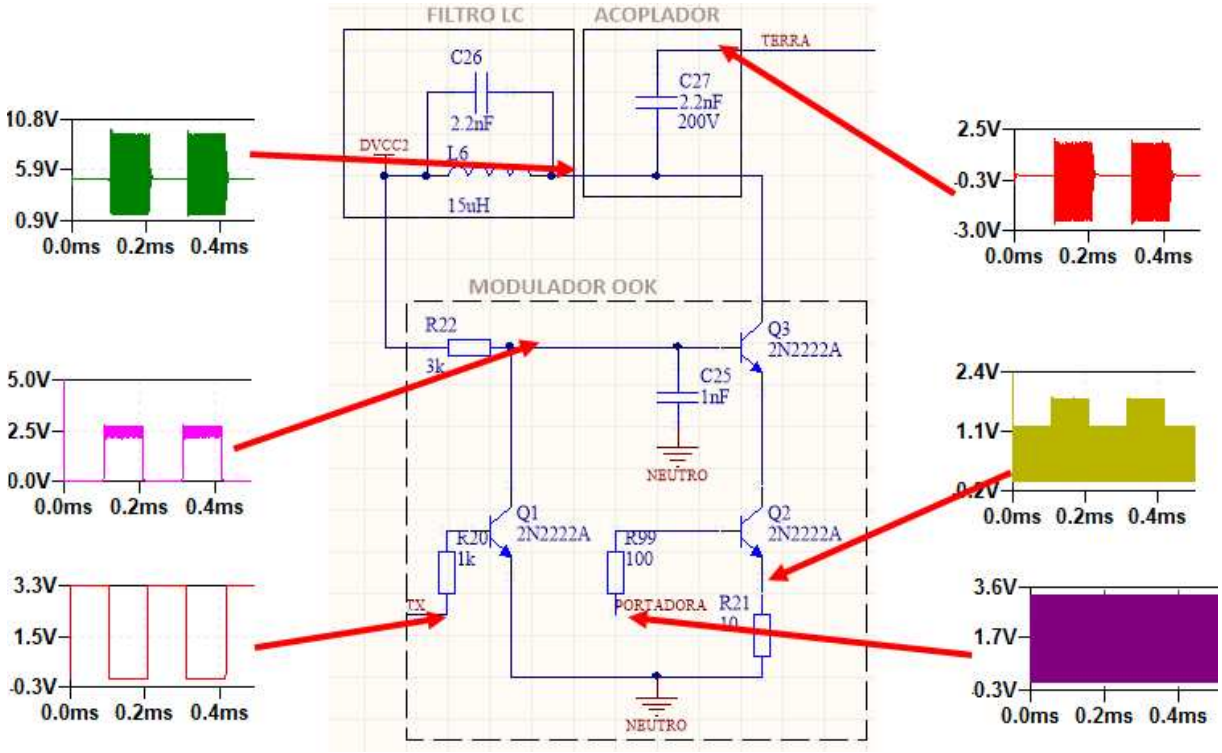


Figura 22. Simulação do circuito transmissor proposto.

3.4.4 Receptor

O receptor é formado por um acoplador, filtro passa-faixas, amplificador a transistor na configuração emissor comum e demodulador OOK, conforme apresentado na Figura 23. O acoplador e o filtro passa-faixa do circuito receptor possuem estrutura e finalidades idênticas ao descrito na Seção 3.4.1: limitar a banda do sistema e desacoplar o circuito receptor da rede elétrica.

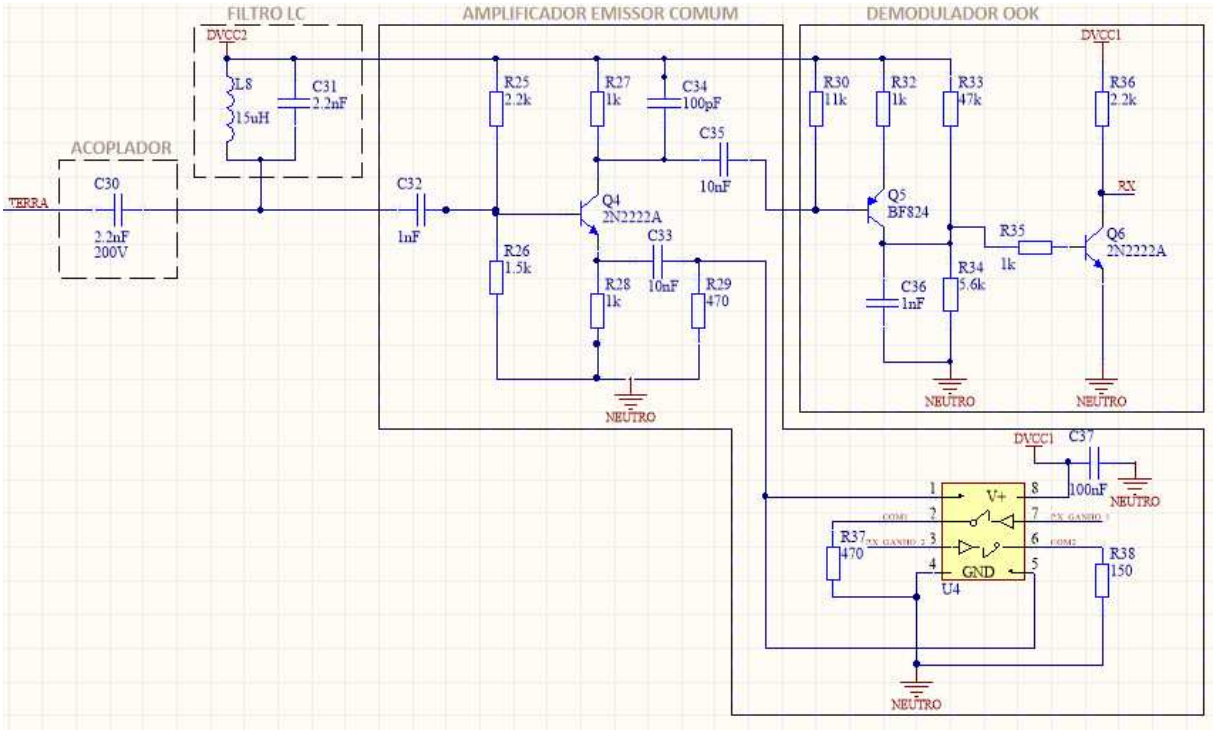


Figura 23. Circuito receptor do modem PLC proposto.

A resposta em frequência da entrada do receptor é mostrada na Figura 24.

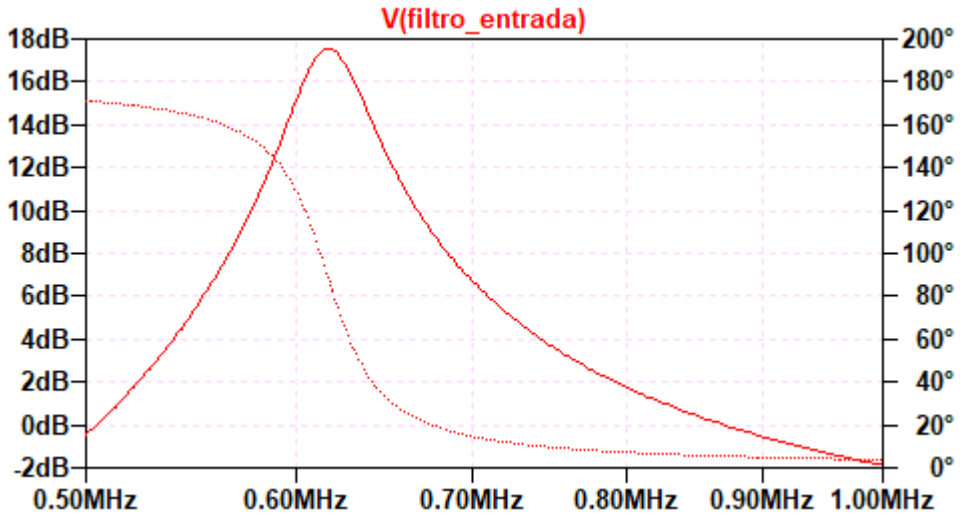


Figura 24. Resposta em frequência do filtro de entrada do circuito receptor.

Após filtrado, o sinal passa por um estágio de amplificação. Para manter a simplicidade da solução, foi utilizado amplificador a transistor na configuração emissor comum com ajuste de ganho definido por software. Esta configuração apresenta ganho de tensão A_V dado por:

$$A_V = -\frac{R_C}{r_e + R_E}, \tag{8}$$

em que, R_C é a resistência conectada ao coletor do transistor; R_E é a resistência conectada ao emissor do transistor; e r_e é a resistência definida pela operação CC do amplificador, dada por:

$$r_e = \frac{V_T}{I_C}, \tag{9}$$

em que, $V_T = 25$ mV à temperatura de 20 °C e I_C é a corrente do coletor.

Os capacitores utilizados nesses amplificadores são responsáveis por limitar a largura de banda do amplificador; sinais fora da banda de interesse não são amplificados por este estágio.

O ajuste do ganho do amplificador é realizado pela alteração da resistência R_E da Equação (8). Esta alteração é realizada utilizando o circuito integrado TS5A2066 da Texas Instruments, uma chave analógica *single pole single throw* (SPST) com seleção para 4 saídas a partir de 2 bits de controle [38]. O algoritmo responsável pela seleção do ganho do estágio de amplificação será discutido na Seção 3.6.3.

Após filtragem e amplificação, o sinal é enviado para o circuito de demodulação. Esse circuito é responsável por recuperar o dado enviado, separando a portadora de alta frequência do sinal modulante. Neste estágio, o sinal é filtrado por um filtro passa-baixas.

No último estágio, o sinal modulante recupera sua fase através de um transistor atuando como inversor lógico. As formas de onda do estágio de amplificação e demodulação simulados são apresentadas na Figura 25.

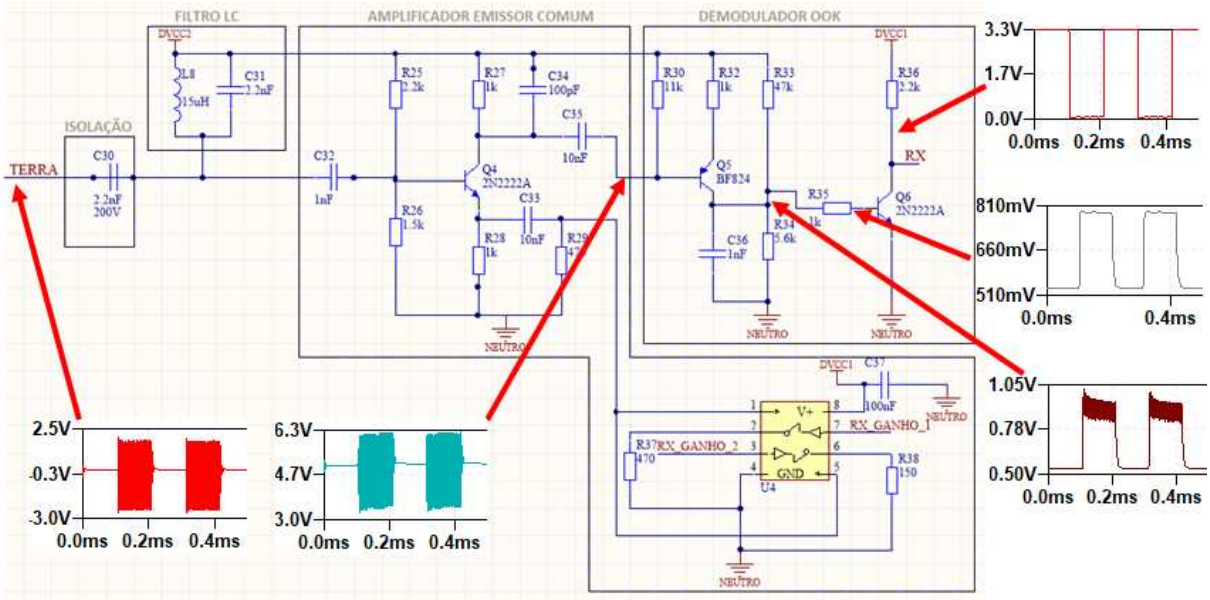
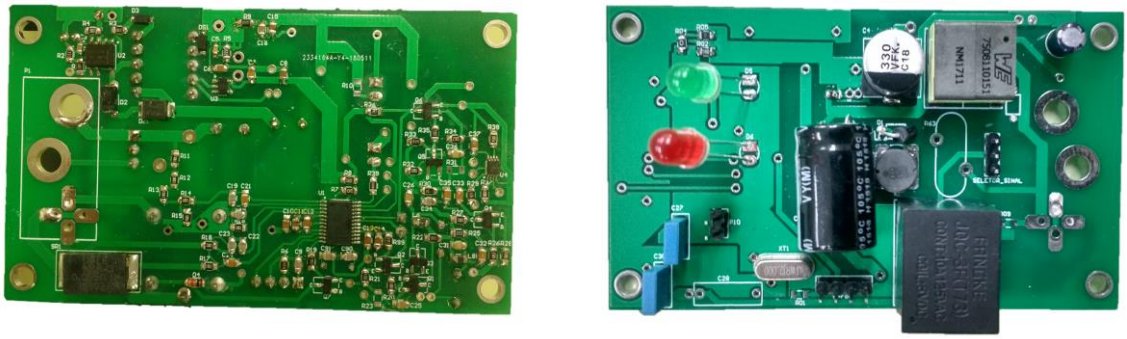


Figura 25. Formas de onda do receptor proposto.

3.5 Hardware Desenvolvido

O módulo medidor PLC foi desenvolvido no Laboratório de Instrumentação, Sensores e Sistemas Embarcados – LISSE da Universidade Estadual de Campinas – UNICAMP. O desenvolvimento dos esquemáticos e do projeto da PCB foi realizado utilizando o software *Altium Design*. A fabricação dos protótipos foi realizada na China. A Figura 26 apresenta o protótipo desenvolvido.



(a) (b)

Figura 26. Hardware desenvolvido. (a) Face superior e (b) Face Inferior.

Foi utilizada uma PCB dupla face com dimensões de 9,2 cm por 5,4 cm, componentes SMD e THT, e técnicas de roteamento de maneira a separar os circuitos de alimentação, medição e modulação.

Na face superior foram colocados os componentes SMD , como apresentado na Figura 26a. Estão presentes nessa face os circuitos de medição de tensão e corrente, a unidade de processamento, o modem PLC e parte do circuito de alimentação.

Na face inferior estão presentes componentes THT. A maioria dos componentes presentes nesta face são referentes ao circuito de alimentação (transformador, capacitores, diodo). Nessa face também estão presentes o relé e os capacitores de acoplamento do modem PLC. Alguns componentes SMDs foram alocados nessa face para facilitar o roteamento da PCB, como o indutor de entrada do circuito de alimentação, como apresentado na Figura 26b.

Durante o roteamento da PCB, teve-se o cuidado de separar os circuitos de modo a minimizar o ruído no circuito de medição, segundo o conceito de *star ground*. A referência do circuito do modem PLC e do circuito de medição são distintas e se conectam apenas na referência do circuito de alimentação. Além disso, o chaveamento da alimentação pode ser fonte de ruído irradiado no circuito de medição, para tanto, teve-se o cuidado de manter distância entre o circuito de alimentação e o circuito de medição de modo a minimizar este efeito.

O posicionamento dos pinos de conexão com a tomada foi realizada baseando-se na norma NBR 14136 [47] para uma carga de até 10 A_{RMS}. A conexão dos pinos de fase e terra são vazadas, ou seja, ligadas diretamente na rede elétrica. O pino de neutro, contudo, possui uma interrupção necessária para medir a corrente da carga.

3.6 Firmware do Módulo Medidor

O firmware do módulo medidor PLC é responsável por: calcular os parâmetros elétricos da carga monitorada (tensão e corrente eficazes, potência ativa e aparente, energia ativa consumida e fator de potência); selecionar a frequência da portadora adequada para ser utilizada no circuito PLC; e ajustar o ganho do circuito de recepção baseado no nível de ruído do canal de transmissão. O fluxograma simplificado do firmware do sistema é apresentado na Figura 27. Os algoritmos foram separados em Medição de Energia Elétrica, Seleção de Frequência e Ajuste de Ganho, e serão apresentados detalhadamente nas seções 3.6.1, 3.6.2 e 3.6.3.

Todos os programas do modulo medidor foram escritos em linguagem C ANSI C99 (ISO/IEC 9899:1999) usando o compilador Code Composer Studio v 7.4.0.00015.

Após a inicialização do microcontrolador, são realizadas as configurações do hardware: *clocks*, portas de I/O, comunicação UART e conversor A/D. O clock principal do microcontrolador (MCLK) é configurado para usar o sinal proveniente de um oscilador externo de 12 MHz. O *clock* secundário (SMCLK), utilizado pelos periféricos, é configurado para usar o sinal gerado internamente pelo Oscilador Controlado Digitalmente (DCO). O DCO é a fonte de *clock* utilizada na geração do sinal da portadora do Modem PLC. Ele foi configurado para trabalhar na máxima frequência possível, aproximadamente 21 MHz, permitindo a configuração da frequência da portadora com resolução que varia de 10 kHz à 50 kHz. O conversor A/D é configurado com a fonte de *clock* estável, oriunda do MCLK, cujo valor é dividido por 12 para resultar na taxa de amostragem efetiva de 1 MS/s.

Após a configuração dos *clocks* e do módulo UART, o conversor A/D é inicializado e o sistema entra em *loop* infinito. No *loop* infinito, o sistema fica aguardando interrupções do conversor A/D, indicando o fim de uma conversão e a obtenção da amostra de tensão e corrente. Ao fim de cada conversão, os valores obtidos são armazedos em vetores e acumulados para a compensação de *offset* da medida.

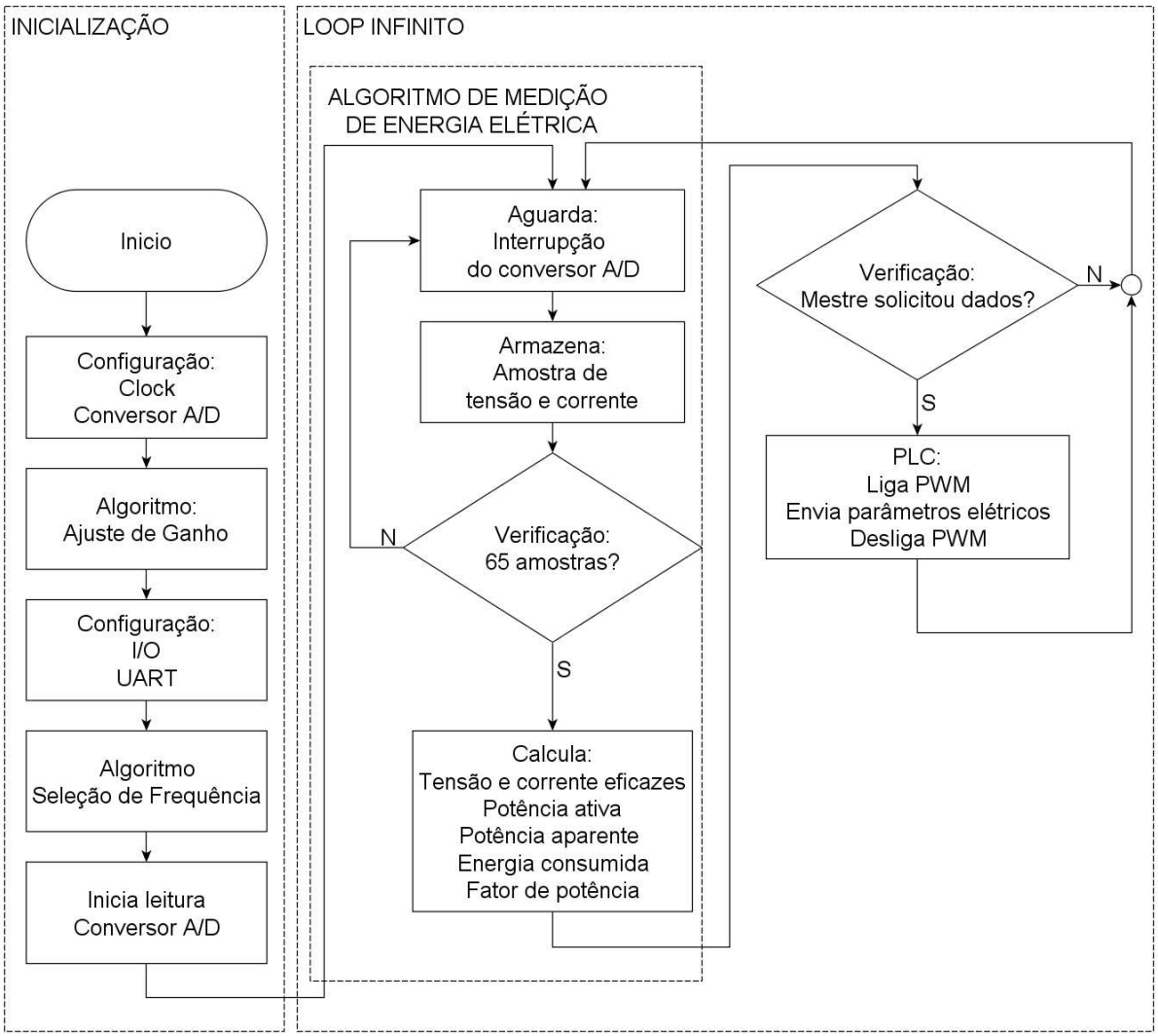


Figura 27. Fluxograma simplificado do módulo medidor PLC.

3.6.1 Medição de Energia Elétrica

Os conversores A/D operam a uma taxa de aproximadamente 3906 S/s. Com essa taxa de amostragem, em redes de 60 Hz, é necessário receber 65 amostras para se obter um ciclo completo da rede. O cálculo dos parâmetros elétricos é iniciado somente após a aquisição de 65 amostras. Durante o cálculo das grandezas elétricas, os conversores A/D são desligados e somente ligados novamente ao fim do cálculo. Com as amostras obtidas são calculadas tensão e corrente eficazes, potência ativa e aparente, energia ativa consumida e fator de potência, conforme apresentado nas equações a seguir.

Os valores de tensão e corrente eficazes são calculados, respectivamente, através das seguintes equações:

$$V_{RMS} = K_V \sqrt{\sum_{n=1}^{N} \frac{v[n]^2}{N}}, \quad (10)$$

$$I_{RMS} = K_A \sqrt{\sum_{n=1}^{N} \frac{i[n]^2}{N}}, \quad (11)$$

em que, V_{RMS} é o valor eficaz da tensão, K_V é o ganho de tensão, $v[n]$ a enésina amostra de tensão, I_{RMS} é o valor eficaz da corrente, K_A é o ganho de corrente, $i[n]$ a enésina amostra de corrente, n é o índice da amostra e N é o número total de amostras.

O valor da potência ativa é calculado por:

$$P = K_V * K_A \sum_{n=1}^{N} \frac{v[n] * i[n]}{N}, \quad (12)$$

onde, P é o valor de potência ativa, K_V é o ganho de tensão, K_A é o ganho de corrente, $v[n]$ é a enésina amostra de tensão, $i[n]$ é a enésina amostra de corrente, n é o índice da amostra e N é o número total de amostras.

O valor de potência aparente S é calculado utilizando os valores encontrados nas equações (10) e (11), conforme apresentado na Equação (13) :

$$S = V_{RMS} * I_{RMS}. \quad (13)$$

O valor do fator de potência FP é calculado pela razão entre a potência ativa e a potência aparente, conforme Equação (14) :

$$FP = \frac{P}{S}. \quad (14)$$

Na Figura 28, é apresentado o fluxograma do módulo medidor de energia elétrica. Após a aquisição das amostras, os valores de tensão e corrente são armazenados em vetores para processamento futuro. Em seguida, os valores das amostras são acumulados para compensação o *offset* da medida. Após a aquisição de 65 amostras, o equivalente a um ciclo da rede elétrica, os valores de *offset* de tensão e corrente são calculados a partir da média dos valores acumulados. Os *offsets* calculados são subtraídos

dos valores de cada amostra de tensão e corrente. Finalmente, os valores das grandezas elétricas são calculados conforme as equações descritas anteriormente nesta seção.

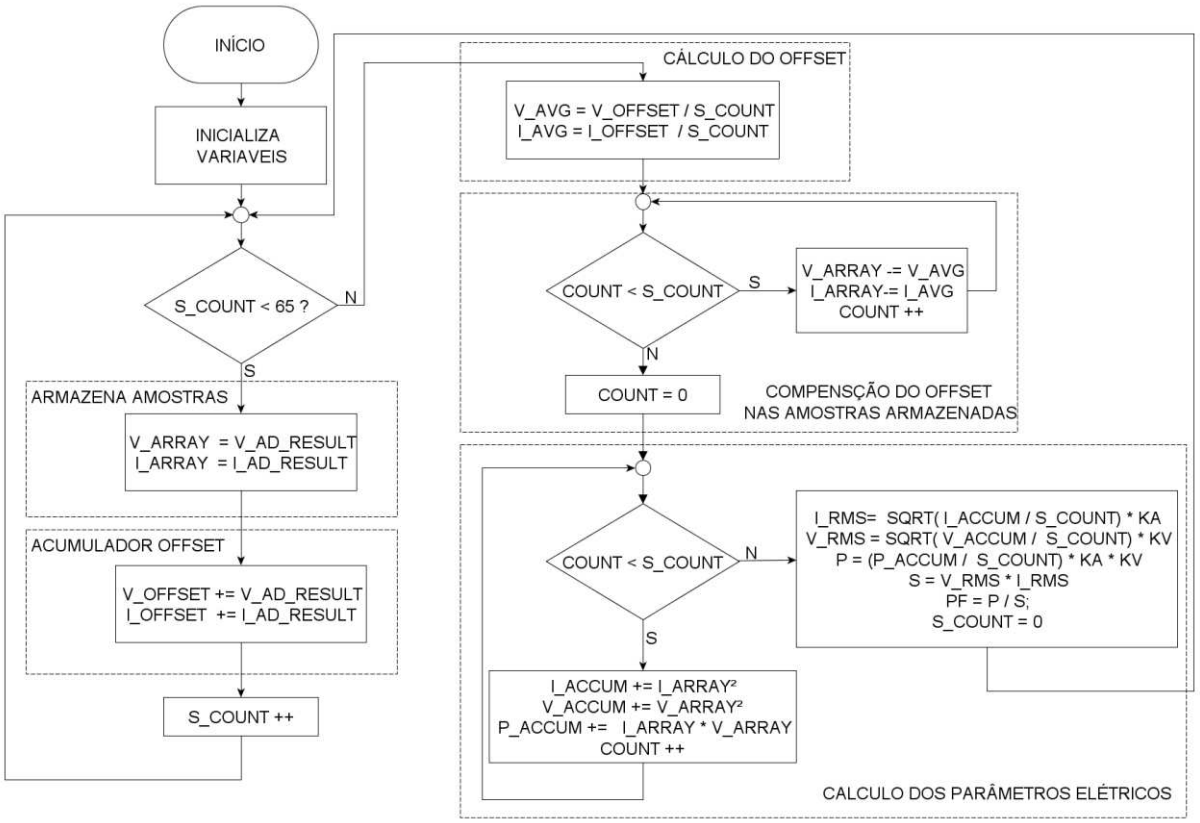


Figura 28. Fluxograma do módulo medidor de energia elétrica.

3.6.2 Algoritmo de Seleção de Frequência da Portadora PLC

Visto que as características do canal de transmissão da rede PLC são diferentes em cada instalação elétrica, é necessário ajustar frequência do sinal de portadora para o correto funcionamento do modem em diferentes canais de comunicação. O algoritmo proposto nesta seção tem o objetivo de aumentar a disponibilidade do modem PLC a partir de um algoritmo simples de seleção de frequência de portadora. O algoritmo consiste em selecionar uma dentre 23 frequências disponíveis (no intervalo de 500 kHz e 1 MHz) para o sinal de portadora do PLC.

Na Figura 29 é apresentado o fluxograma do algoritmo proposto. Esta rotina é executada na inicialização do sistema. O módulo PWM do microcontrolador é configurado para gerar o sinal de portadora do PLC com *duty cycle* de 50 %. Inicialmente, a frequência da portadora é configurada em 500 kHz e incrementada a cada iteração do algoritmo. O

valor incrementado da frequência não é constante. Inicia-se com aproximadamente 10 kHz e termina com aproximadamente 50 kHz. Essa variação é devido a fonte de *clock* responsável pela geração do PWM não conseguir manter a resolução constante para toda a faixa de frequência utilizada.

A cada iteração do algoritmo, é enviado para o modem PLC um pacote de um byte com os seguintes bits "01010101". Verifica-se o eco da transmissão; se o valor recebido for idêntico ao enviado a frequência corrente é uma candidata à frequência da portadora, e neste caso, essa frequência é armazenada em um acumulador (FREQ_ACUM) e o valor de um contador (FREQ_COUNT) é incrementado. Ao final do algoritmo, quando a frequência do sinal PWM chega ao valor máximo de 1 MHz, a frequência da portadora é configurada com o valor médio das frequências nas quais não houve erros de recepção.

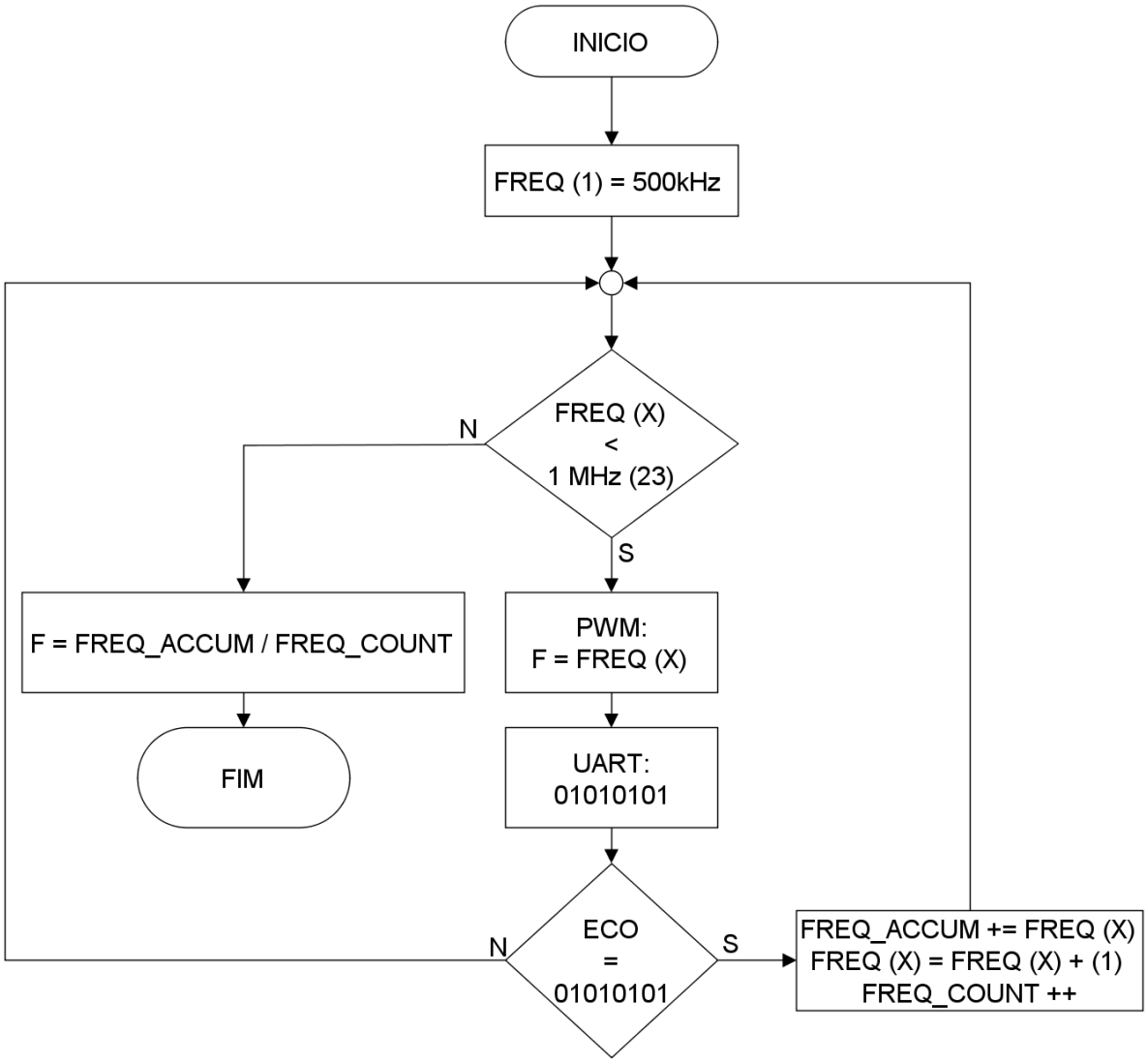


Figura 29. Fluxograma do algoritmo de seleção de frequência de portadora.

3.6.3 Algoritmo Para Ajuste de Ganho do Circuito Receptor

O algoritmo para ajuste de ganho do receptor PLC surgiu devido à necessidade de instalar os módulos medidores em redes elétricas que possam gerar grande atenuação ao sinal modulado e que, desta forma, inviabilizasse a comunicação. Foi implementado um algoritmo simples para controle de uma chave analógica responsável por selecionar um dos quatro ganhos possíveis do estado de amplificação. A seleção dos ganhos é realizada baseado no nível de ruído do canal de transmissão. O algoritmo para ajuste da frequência da portadora (descrito na seção anterior) e o algoritmo para ajuste de ganho do receptor PLC possuem a finalidade de aumentar a disponibilidade do sistema em instalações elétricas diversas.

A Figura 30 apresenta o fluxograma do algoritmo de ajuste de ganho. Assim como a rotina descrita anteriormente, essa também é executada na inicialização do sistema. A ideia básica desse algoritmo é a seguinte: deve-se usar o maior ganho possível, desde que não sejam identificados “falsos zeros” devido ao ruído. Dessa forma, em redes pouco ruidosas, devem ser usados ganhos elevados, permitindo a comunicação a longas distâncias, ao passo que em redes ruidosas devem ser usados ganhos pequenos, para que não sejam demodulados “falsos zeros”.

O algoritmo funciona da seguinte forma: incia-se com o ganho mínimo e incrementa-o até o valor máximo. Em cada um dos ganhos, verifica-se se o circuito demodulador reconhece algum bit 0 durante um período pré-estabelecido (1 s). Se for identificado algum bit 0, o ganho está muito alto, e o ruído está sendo identificado como bit 0. Então, utiliza-se o ganho imediatamente anterior.

Três portas digitais do microcontrolador são utilizadas por este algoritimo: duas portas configuradas como saídas, que realizam o controle da chave analógica; e a terceira porta configurada como entrada, para monitorar o nível lógico na saída do demodulador. A seleção dos bits de controle resulta nos ganhos apresentado na Tabela 1 com referência às identificações de resistências da Figura 23.

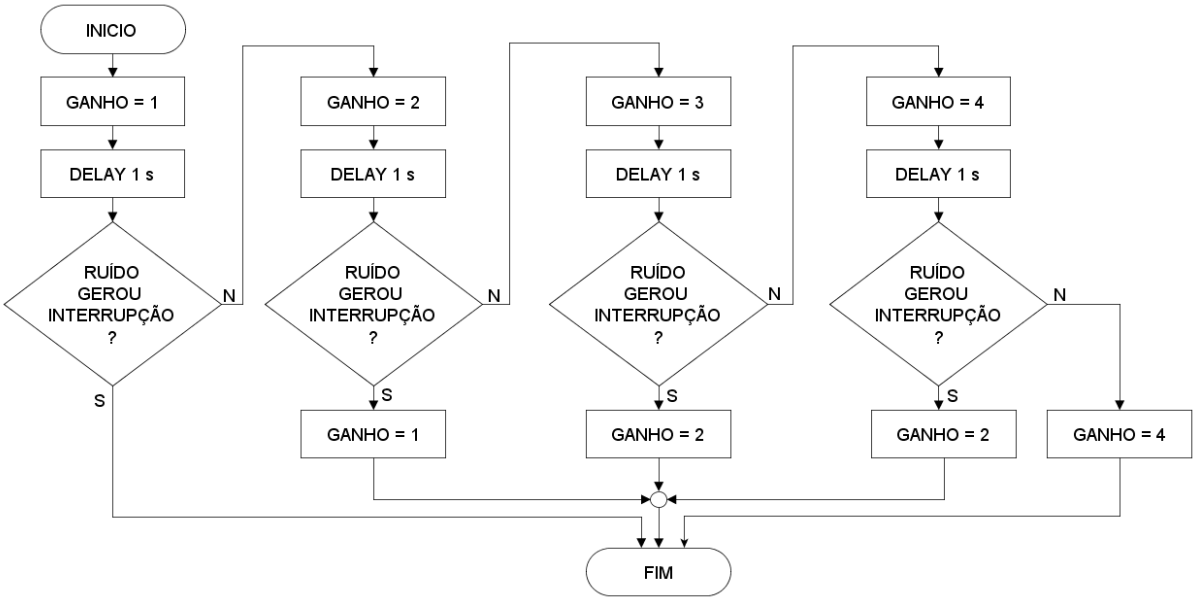


Figura 30. Fluxograma do algoritmo de ajuste de ganho.

Tabela 1. Valores de ganho no estágio de amplificação

BIT CONTROLE_2	BIT CONTROLE_1	RESISTÊNCIA EQUIVALENTE (Ω)	ASSOCIAÇÃO	GANHO (DB)
0	0	470	R29//R28	8
0	1	235	R29//R28//R37	12
1	0	113	R29//R28//R38	16
1	1	110	R29//R28//R37//R38	18

3.6.4 Protocolo de Comunicação

A troca de informação entre o módulo mestre e os módulos medidores é relizada através dos modems PLC desses dispositivos. Os dados digitais a serem transmitidos são enviados pelas unidades de processamento de forma serial (protocolo UART) para os modems PLC, onde são modulados e inseridos na rede elétrica. A informação inserida na rede elétrica é recebida por todos os modems conectados à rede, que neste caso operam como receptores. Os sinais recebidos são demodulados e enviados de forma serial para a unidade de processamento. A unidade de processamento é responsável por verificar a informação recebida e prover a resposta do respectivo medidor solicitado pelo módulo mestre.

O módulo mestre é responsável por controlar o fluxo de dados na rede. Ele envia a requisição de dados para os módulos medidores, os quais respondem com os

parâmetros elétricos solicitados. O módulo mestre também é responsável por enviar aos módulos medidores os comandos para acionar ou desacionar as cargas a eles conectados. Dessa forma, na rede desenvolvida, os módulos medidores se comportam como escravos.

Cada um dos módulos escravos é identificado por um endereço único *'i'* cujo valor é definido no *firmware* do módulo escravo durante a fabricação. Dessa forma, o módulo mestre deve definir no pacote enviado o endereço do receptor. Apesar de todos os medidores ouvirem as requisições, apenas o medidor com o endereço informado responderá.

Os módulos escravos são capazes de processar três tipos de comandos: *set, reset* e *get*. Todos esses comandos possuem 4 bytes, são iniciados com o caractere *'#'* e finalizam com o caractere *';'*. A estrutura desses comandos é apresentada abaixo:

- *get*: *'#' 'i' 'g' ';'*
- *set*: *'#' 'i' 's' ';'*
- *reset*: *'#' 'i' 'r' ';'*

As mensagens do tipo *set* e *reset* não geram respostas. O modulo medidor responde às mensagens *get* com os parâmetros elétricos solicitados. O formato da resposta do módulo medidor é diferente do formato dos comandos enviados pelo módulo mestre. A resposta que o módulo medidor envia para o módulo mestre é iniciada pelo caractere *'*'* seguido dos valores de tensão e corrente eficazes, potência ativa e aparente, energia ativa consumida e fator de potência, e então, finalizada pelo caractere *'!'*. Cada grandeza elétrica é composta de valores do tipo *float* de 4 bytes, resultando em um pacote de 26 bytes.

A Figura 31 apresenta a máquina de estados implementada no módulo medidor com o objetivo de tratar byte a byte os caracteres recebidos pela UART. Inicialmente a máquina de estado é colocada em estado OCIOSO, no qual aguarda a chegada do byte inicial *'#'.* Após o recebimento do byte inicial, a máquina de estado vai para o estado ID, no qual aguarda o byte seguinte, referente à identificação *'i'* do módulo medidor. Após a validação da identificação do medidor, a máquina vai para o estado COMANDO, e aguarda a leitura do próximo byte referente à ação do módulo medidor, que pode ser: *'s'* para o comando *set*, *'r'* para o comando *reset* e *'g'* para o comando *get*. Por fim, após o recebimento do caractere de fim de comando *';'*, a máquina é colocada em

estado OCIOSO e a rotina para tratamento do comando é chamada. Qualquer byte recebido diferente dos bytes esperados pela máquina em cada um de seus estados faz com que a máquina retorne para o estado OCIOSO.

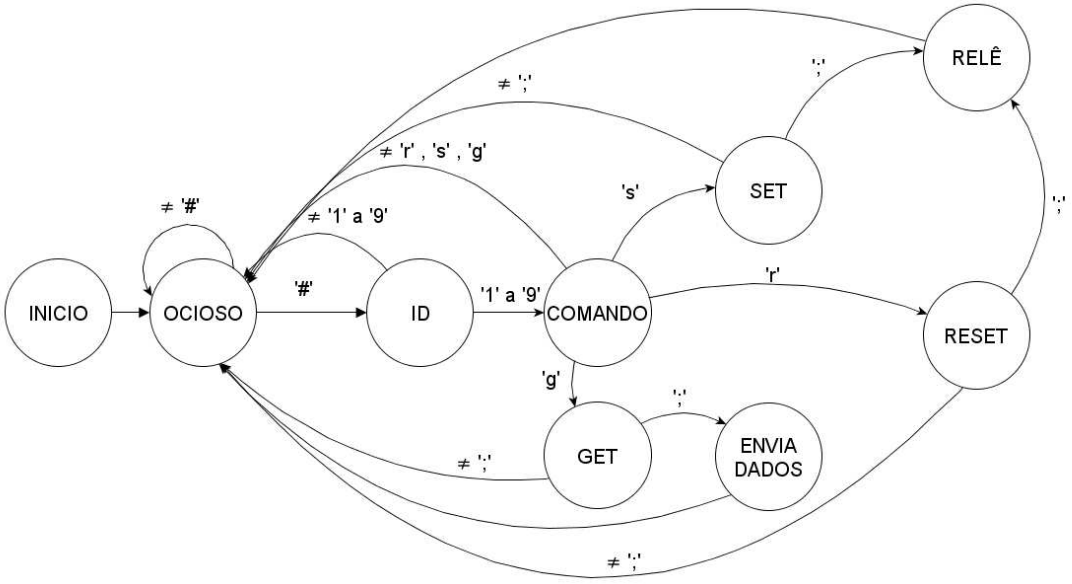


Figura 31. Máquina de estado do tratamento UART dos comandos enviados pelo

Capítulo 4

Sistema Concentrador de Dados

Neste capítulo, é apresentada toda a infraestrutura de software para armazenamento e disponibilização de dados do sistema de controle e moniramento de consumo de energia elétrica para validação do sistema de medição proposto.

4.1 Descrição do Sistema

O Sistema concentrador de dados é composto pelo módulo mestre *(Gateway)*, serviço de hospedagem web, MQTT broker e interface com usuário via web e smartphone. A Figura 32 apresenta o diagrama de blocos do Sistema concentrador de dados proposto.

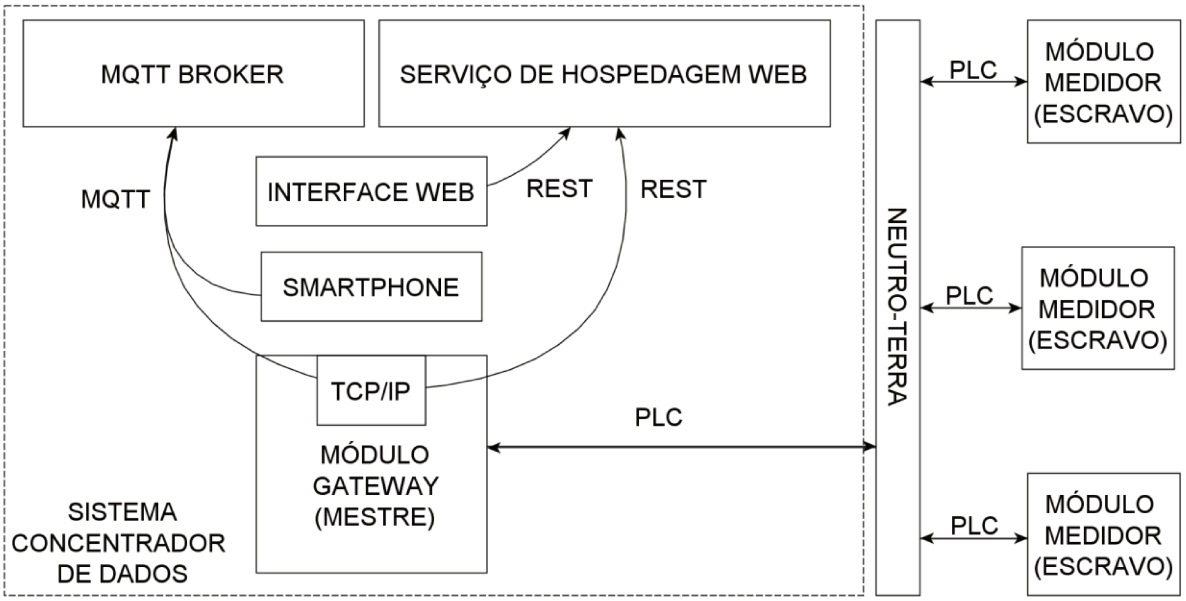


Figura 32. Sistema de concentração de dados.

Como apresentado anteriormente, o módulo mestre é responsável por solicitar informações aos módulos medidores conectados à rede elétrica. A resposta dos módulos medidores é processada pelo *Gateway*, o qual realiza a comunicação entre a rede local (PLC) e a internet. Os dados enviados pela internet são armazenados em um banco de dados disponibilizado pelo serviço de hospedagem web. O serviço de hospedagem web oferece uma interface web para apresentação dos dados dos medidores para os usuários. O acesso às informações das cargas monitoradas também é disponível aos usuários através de um aplicativo para smartphones.

4.2 Serviço de Hospedagem web

O uso de um sistema de hospedagem web comercial permite que os dados tenham alta disponibilidade e elimina a preocupação com o armazenamento dos mesmos. Serviços de hospedagem como o oferecido pelo *PythonAnywhere* disponibilizam um ambiente de desenvolvimento integrado (IDE) baseada em *Python*, além de sistema de gerenciamento de banco de dados *MySQL* com disponibilidade de 1 GB de armazenamento. A IDE permite desenvolver API (*Application Programming Interface)* em *Python* de modo que programas externos acessem as funcionalidades disponíveis pelo *PythonAnywhere*. Com este serviço, foi possível criar uma API responsável por executar as solicitações de armazenamento e consulta de dados enviadas pelo módulo mestre e disponibilizar as páginas html (*hypertext markup language*) criadas para interface com o usuário. O código desenvolvido está disponível no Apêndice A.

A API desenvolvida e executada pelo serviço *PythonAnywhere* não é responsável por realizar o controle dos módulos medidores, esta funcionalidade foi implementada utilizando o protocolo MQTT *(Message Queuing Telemetry Transport)*. O MQTT permite a integração com smartphones de forma mais simples do que através de API em *Phyton*. Além disto, este protocolo tem sido bastante difundido em soluções IoT que requerem a transmissão de mensagens pequenas (comandos) entre dispositivos.

4.2.1 Flask Framework

Frameworks web são utilizados para simplificar a criação de aplicações e serviços web. Fornecem modelos (*templates*) para criação de páginas web. O

microframework Flask é uma plataforma para desenvolvimento web descrita em *Python* cujo objetivo é oferecer o desenvolvimento de soluções web de maneira rápida e simples.

Com o objetivo de possibilitar a comunicação entre dispositivos (clientes) e o serviço web (servidor), o *framework* oferece a oportunidade de desenvolver a API REST de maneira simples, utilizando extensão compátivel com a linguagem *Python*.

4.2.2 REST API

A arquitetura *REST* é um modelo de arquitetura de software distribuído baseada no protocolo *http* que permite a interoperalidade entre sistemas computacionais conectados à internet. É comumente utilizada para implementar APIs para acesso de dispositivos a um serviço web. A comunicação entre dispositivos (clientes) e o serviço web (servidor) é realizada por URLs (Localizador Uniforme de Recursos), a qual especifica o ponto de entrada na API. A Figura 33 apresenta um exemplo de criação de um ponto de entrada da API na IDE do *PythonAnywhere* em linguagem Python e o tratamento da requisição enviada por um cliente.

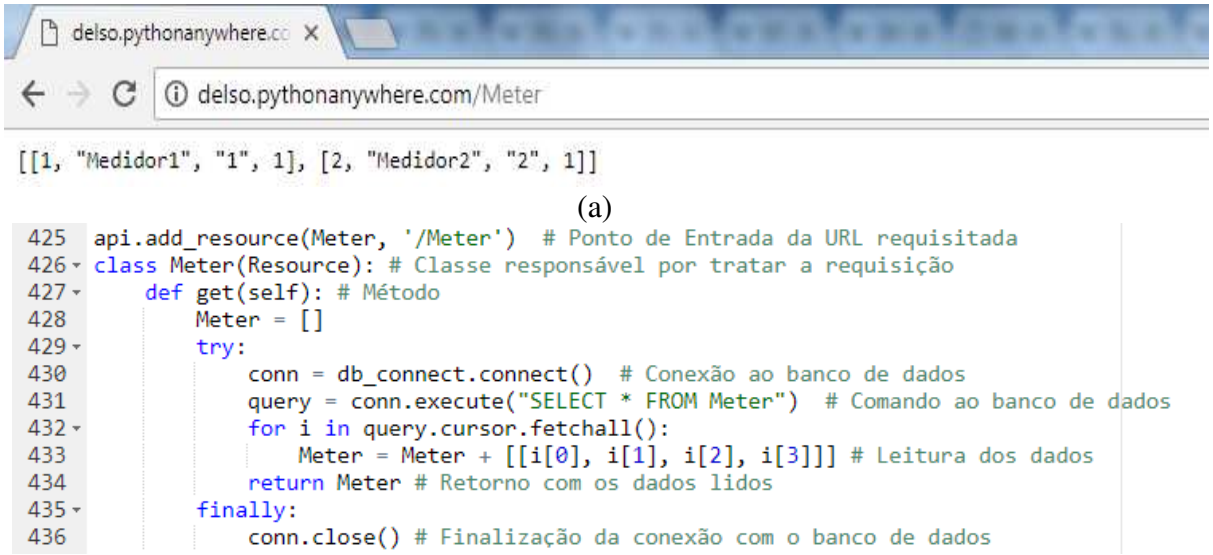


Figura 33. (a) Retorno da solicitação do cliente. (b) Implementação da REST API em Python

Na Figura 33a é possível visualizar uma solicitação enviada por um cliente para o servidor através da URL. A URL *delso.pythonanywhere.com/Meter* é enviada para o serviço de hospedagem, cujo dominio é *delso.pythonanywhere.com* e */Meter* é o ponto de entrada para a API, conforme mostrado na Figura 33b. Uma vez que */Meter* é um

caminho válido, a classe *Meter* é executada. O método dessa classe realiza conexão ao banco de dados e seleciona os dados da tabela *meter*. O retorno desse método é apresentado na Figura 33a. Portanto, uma vez implementada uma API como a apresentada, é possível criar caminhos para diversas solicitações do módulo mestre baseada em URLs para leitura e armazenamento de dados.

4.2.3 Interface WEB

As páginas web para interação com os usuários foram desenvolvidas em *html* e são disponibilizadas conforme as solicitações *http* recebidas pelo serviço de hospedagem web. Funções são implementadas em *Python* para tratar cada solicitação *http*, cujo retorno é a página solicitada. É possível nessas funções, realizar consulta a banco de dados de forma a disponibilizar a página com valores dinâmicos.

Foi implementado um conjunto de páginas web responsáveis por apresentar os parâmetros elétricos de cada medidor registrado no banco de dados de forma simples e amigável, agregando a funcionalidade de consulta de histórico de cada parâmetro monitorado e a apresentação dos dados graficamente. As páginas criadas estão disponíveis no Apêndice B.

4.2.4 MQTT Broker

O MQTT é um protocolo *machine-to-machine* (M2M) criado pela IBM em 1999. Ele é utilizado em aplicações que requerem pouca largura de banda e alta latência. Utiliza o padrão de mensagem *publish–subscribe*, no qual a troca de mensagens entre dispositivos é realizada por uma entidade central *(Broker)*, responsável por encaminhar mensagens dos transmissores (*publishers)* para os receptores (*subscribers)* através de tópicos *(topics),* conforme apresentado na Figura 34.

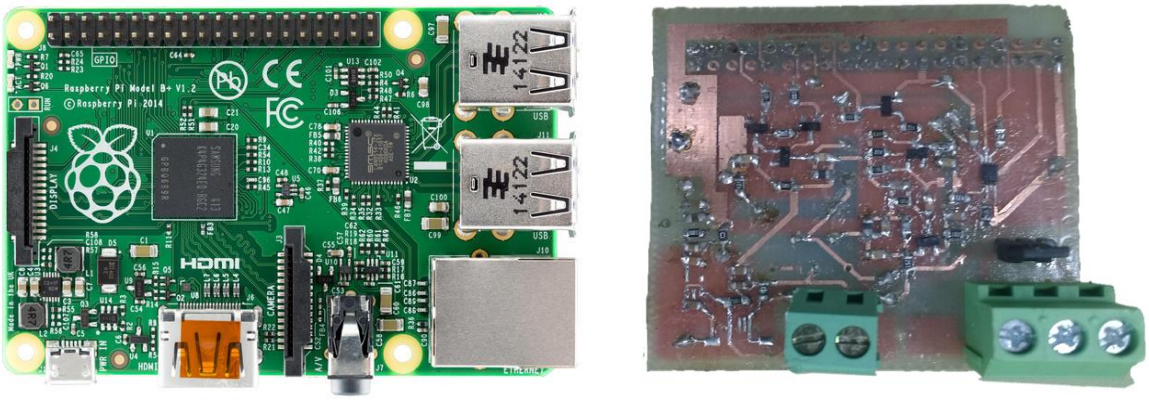
O padrão de mensagens *publish–subscribe* oferece o desacoplamento de *publishers* e *subscribers*; ambos não necessitam conhecer quais os tipos de dispositivos que formam a rede. Devido às suas caracteríticas, esse protocolo tem sido difundido em soluções M2M e IoT, onde largura de banda e consumo de energia são requisitos importantes [48].

Figura 34. Padrão *publish–subscribe* utilizado no protocolo MQTT [48].

Dessa forma, foi utilizado o protocolo MQTT para realizar o controle da carga via smartphone. A partir da interface criada utilizando o aplicativo *MQTT Dash* (disponível para *Android),* foi possível monitorar e controlar os módulos medidores presentes na rede PLC. A interface desenvolvida oferece dois modos de funcionamento: normal e stand-by. No modo normal, o usuário pode controlar as cargas, acionando ou desligando qualquer carga a qualquer instante. No modo stand-by, é possível desligar automaticamente as cargas que estão no estado stand-by. A identificação das cargas em stand-by é feita com base no consumo de corrente das mesmas. Considerou-se que cargas com consumo inferior a 50 mA (equivalente a uma carga de 6 W em redes 127 V_{RMS}) estão em stand-by. Dessa forma, no modo stand-by, se forem adquiridas 10 leituras abaixo do valor de 50 mA para alguma carga, ela será desligada automaticamente. Para reativar a carga o usuário deve colocar o sistema em modo normal e então realizar o acionamento pelo smartphone. A interface para controle via smartphone está disponível no Apêndice C.

4.3 Módulo Mestre (*Gateway*)

O módulo mestre possui duas funções: solicitar as leituras dos módulos medidores via PLC e atuar como interface entre a rede PLC e a internet. Este módulo é composto por uma Raspberry pi 1 modelo B+ e pelo modem PLC. Raspberry pi é um mini computador de baixo custo e com dimensões de um cartão de crédito criada no Reino Unido pela Raspberry Pi Fundation com o objetivo de inserir crianças, jovens e adultos no mundo da computação e no ensino de programação [49]. O modelo B+ foi lançado em 2014 e possui núcleo ARM single-core de 32 bits e 700 MHz, memória SDRAM de 256 MB, 4 portas USB, entrada para vídeo, saída para HDMI, interface ethernet, 40 pinos de I/O e consumo máximo de 1,75 W. O sistema operacional usualmente encontrado nesses minicomputadores é o Raspbian, baseado em linux, o qual é recomendado pelo fabricante. Entretanto, são suportadas outras distribuições baseadas em Linux, como, Arch Linux ARM e Fedora; e sistemas operacionais não baseados em linux como Windows 10 IoT. A Figura 35 apresenta a imagem da Raspberry utilizada e do modem PLC fabricado.



(a) (b)

Figura 35. Modulo mestre com interface PLC: (a) Raspberry PI 1 B+ e (b) Modem PLC.

4.4 Sistema de Controle do Módulo Mestre

A implementação do sistema de controle do módulo mestre foi realizada em *Python*. O software embarcado na Raspberry pI implementa os mesmos algoritmos de ajuste de frequência e ganho embarcados nos módulos medidores (descritos nas seções 3.6.2 e 3.6.3). Também é responsável por definir os *topics* e as configurações para o protocolo MQTT; criar *threads* para a requisição de dados de cada módulo medidor presente na rede PLC cadastrado no banco de dados; tratar os dados recebidos pelos módulos medidores; e enviá-los para o serviço de hospedagem web. A entrada no serviço de hospedagem web é realizada por uma API REST, também implementada em *Python*. Essa API tem a função de direcionar as solicitações de leitura e armazenamendo no banco de dados oriundas do módulo mestre.

Os códigos foram separados em 4 arquivos: 3 arquivos de classes (*scripts*) e 1 arquivo principal e estão disponíveis no Apêndice D.

Capítulo 5

Resultados Experimentais e Discussão

Neste Capítulo são apresentados e discutidos os resultados dos ensaios realizados com todo o sistema proposto. Inicialmente, são apresentados os testes em laboratório do modem PLC proposto, utilizando um canal de comunicação emulado. Em seguida, o sistema foi instalado e avaliado em várias instalações elétricas reais. Também são apresentados testes de validação do módulo medidor de energia elétrica utilizando diferentes cargas.

5.1 Modem PLC

Visto que a performance do PLC é fortemente dependente das características da instalação elétrica, faz-se necessário criar um ambiente controlado para avaliar o modem PLC proposto. Dessa forma, foram considerados dois cenários de testes: rede elétrica emulada; e rede elétrica real. Os testes realizados com a rede elétrica emulada propostos por [35] são suficientes para avaliar a performance do modem desenvolvido. Os testes desenvolvidos em redes elétricas reais são importantes para avaliar a viabilidade e o desempenho do modem proposto sujeito às influências de uma rede elétrica real.

5.1.1 Canal de Comunicação Emulado

Os testes com o canal de comunicação emulado têm o objetivo de verificar a taxa de erro em diferentes *baud rates* (9600 b/s e 19200 b/s) com diferentes densidades espectrais de ruído (66,67$\mu V_{RMS}/\sqrt{Hz}$ à 400 $\mu V_{RMS}/\sqrt{Hz}$).

O canal de comunicação emulado foi construído com um cabo PVC, de 1,5 m de comprimento, composto por dois condutores de 2 mm de diâmetro e 2,5mm^2 de seção transversal, desconectados da rede elétrica. Ruídos brancos com diferentes intensidades foram inseridos no canal utilizando o gerador de sinais Agilent 33220a [50]. Visto que o modem PLC proposto é *narrowband* apenas ruídos na banda de interesse vão afetar o desempenho do sistema. A montagem do canal de comunicação emulado é apresentada na Figura 36.

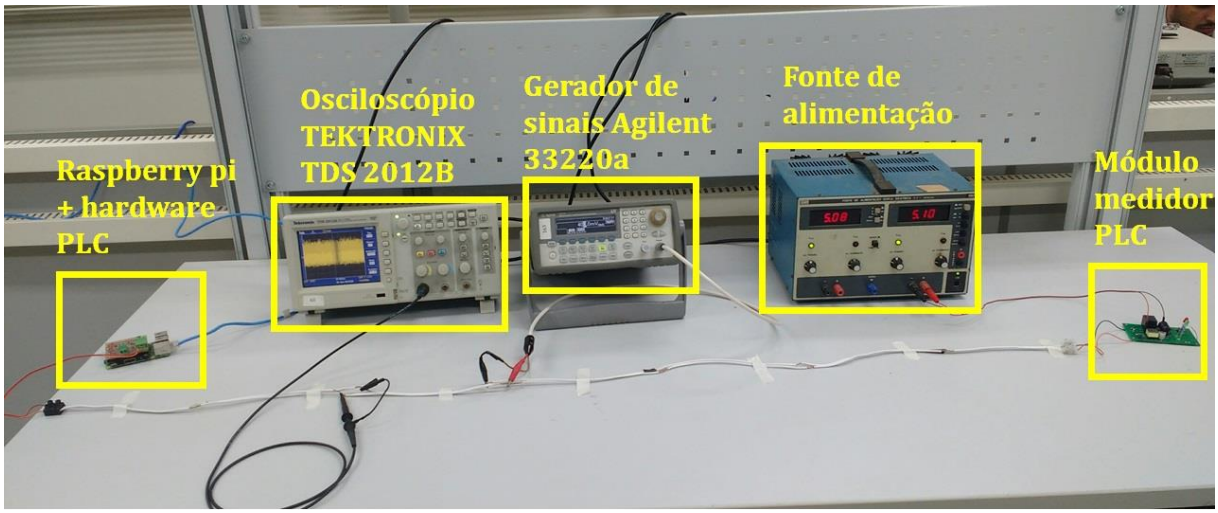


Figura 36. Montagem do canal de transmissão emulado para a análise do modem PLC proposto.

Foram instalados dois módulos PLC, o primeiro operando como mestre e o segundo como escravo. O módulo mestre consiste na raspberry pi com modem PLC, enquanto o módulo escravo é o módulo medidor PLC desenvolvido. A raspberry pi envia 1000 solicitações de dados para o módulo escravo. A cada solicitação, o módulo escravo responde com um pacote de conteúdo “01010101”. Esse dado é transmitido pelo canal de comunicação e recebido pela raspberry pi, a qual verifica a integridade do pacote. Caso o pacote recebido seja igual ao enviado então o pacote foi recebido com sucesso; caso contrário, é considerado como pacote perdido. O resultado do teste no cenário descrito é apresentado na Tabela 2.

Tabela 2. Resultado do teste do modem PLC na rede emulada.

Baud rate (b/s)	Densidade de ruído ($\mu V_{RMS}/\sqrt{Hz}$)	Frequência da portadora PLC (kHz)	Ganho do circuito de recepção (db)	Taxa de erro (%)	Taxa de erro com ganho maior que o ideal (%)			
					8db	12db	16db	18db
9600	66,67	680	18	0	-	-	-	-
	100	680	16	0	-			5
	116,67	680	12	0	-		5	95
	200	680	8	0	-	5	100	100
	266,67	680	8	0	-	63	100	100
	333,34	680	8	2,5	-	100	100	100
	366,67	680	8	9,4	-	100	100	100
	400	680	8	32	-	100	100	100
19200	66,67	680	18	0	-	-	-	-
	100	680	16	0	-	-	-	64
	116,67	680	12	0	-	-	30	95
	200	680	8	0	-	6,2	100	100
	266,67	680	8	0	-	90	100	100
	333,34	680	8	6,5	-	100	100	100
	366,67	680	8	22	-	100	100	100
	400	680	8	55,8	-	100	100	100

O ensaio foi realizado utilizando oito valores de ruído branco: 66,67 $\mu V_{RMS}/\sqrt{Hz}$, 100 $\mu V_{RMS}/\sqrt{Hz}$, 116,67 $\mu V_{RMS}/\sqrt{Hz}$, 200 $\mu V_{RMS}/\sqrt{Hz}$, 266,67 $\mu V_{RMS}/\sqrt{Hz}$, 333,34 $\mu V_{RMS}/\sqrt{Hz}$, 366,67 $\mu V_{RMS}/\sqrt{Hz}$ e 400 $\mu V_{RMS}/\sqrt{Hz}$. Em todos os ensaios, a frequência da portadora selecionada através do algoritmo desenvolvido foi de 680 kHz, a qual corresponde à frequência central encontrada pelo algoritmo proposto dentre o intervalo de frequências de 500 kHz a 1 MHz. Este comportamento é esperado pois na rede emulada proposta não há atenuações *(notches)*, portanto todas as frequências do intervalo poderão ser utilizadas.

O ganho do estágio de recepção foi definido pelo algoritmo apresentado na Seção 3.6.3. Apesar do algoritmo selecionar diferentes valores de ganho quando a densidade espectral de ruído na rede é inferior 116,67 $\mu V_{RMS}/\sqrt{Hz}$, percebe-se que a utilização de ganhos inferiores não afeta o recebimento de pacotes em redes em que a atenuação é pequena (como é o caso da rede emulada). Entretanto, ganhos superiores aos selecionados pelo algoritimo podem gerar erros que variam de 5 % a 100 %.

A comunicação apresenta erros superiores a 10 % para densidades de ruído superiores a 366,67 $\mu V_{RMS}/\sqrt{Hz}$ (considerando *baud rate* de 9600 b/s). O mesmo comportamento é observado para ruídos acima de 333,34 $\mu V_{RMS}/\sqrt{Hz}$ para *baud rates* de 19200 b/s. Dessa forma, sempre que possível, é mais adequado trabalhar com taxas de transmissão menores, pois os efeitos dos ruídos geram menor quantidade de erros nos bits transmitidos.

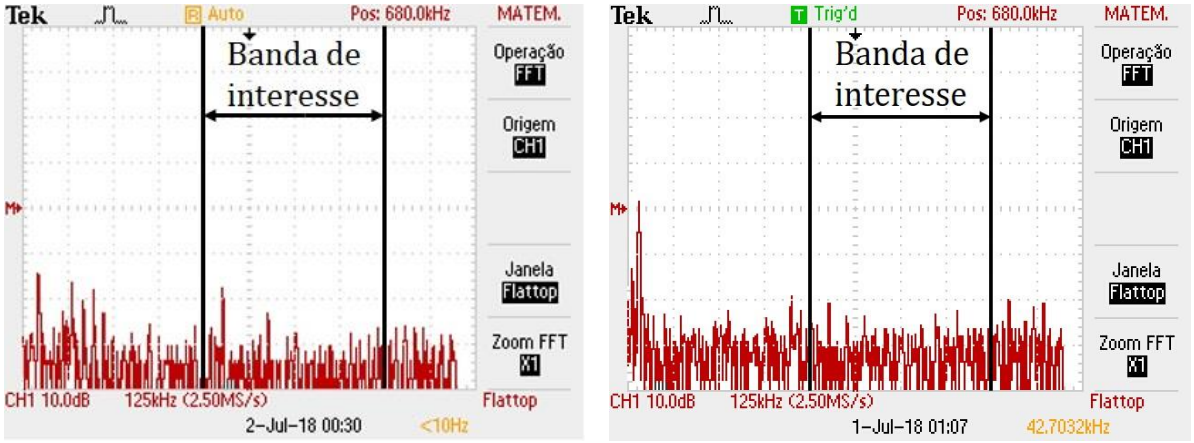
5.1.2 Testes em Campo

Os testes realizados em instalações elétricas reais são similares aos realizados com a rede elétrica emulada. Entretanto, ao invés de utilizar um gerador de sinais para inserir o ruído branco no canal de comunicação, os modems PLC propostos foram sujeitos aos ruídos inerentes a redes elétricas reais. Para avaliar o sistema proposto em diferentes ambientes, foram instalados três modems PLC (um operando como mestre e dois como escravos) em três instalações elétricas distintas: duas residências e um laboratório na UNICAMP.

A diversidade de equipamentos conectados à rede elétrica em cada uma das instalações selecionadas é conveniente para a verificação da robustez e da disponibilidade do sistema de comunicação em diferentes condições. Televisores, microondas, computadores, geladeira, radios AM e demais equipamentos eletrônicos são possíveis fontes de ruído para o sistema proposto. Para realização deste teste foi pressuposto que a instalação elétrica da localidade possua sistema de aterramento conforme as normas vigentes [51] e configuração de aterramento do tipo TN-S, na qual os condutores de terra e neutro são distintos e interligados somente na caixa de distribuição da residência. Além disso, foi utilizada a norma NBR 14136 [47] referente à padronização de tomadas elétricas residenciais como referência de dimensão e posição dos conectores de fase, neutro e terra do módulo medidor com a rede elétrica.

Os dados foram obtidos durante um período de 24 h para cada uma das *baud rates* (9600 b/s e 19200 b/s) em cada uma das instalações. A frequência da portadora e o ganho do estágio de recepção foram selecionados pelos algoritmos propostos no ínicio de cada teste. O módulo mestre foi configurado para solicitar dados para os módulos escravos a cada 5 segundos. A cada solicitação recebida, o módulo escravo envia um pacote de 1000 bytes com conteúdo “01010101”.

O primeiro ambiente de teste foi uma residência localizada na cidade de Campinas–SP, onde a rede elétrica possui tensão nominal de 127 V_{RMS}. Os módulos foram instalados em três tomadas distintas, com distância máxima de aproximadamente 15 metros uma da outra. Antes do início do teste, foi observado o espectro do sinal presente entre terra e neutro da instalação utilizando o osciloscópio TEKTRONIX TDS 2012B [52]. Neste osciloscópio a escala vertical apresenta valores em dB, com relação à 1 V_{RMS}. Na Figura 37 é apresentado o conteúdo espectral do canal de comunicação no início dos testes com os *baud rates* de 9600 b/s e 19200 b/s. O resultado desse experimento é apresentado na Tabela 3.



(a) (b)

Figura 37. Espectro da rede elétrica da residência localizada em Campinas – SP. (a) início do teste com *baud rate* de 9600 b/s. (b) início do teste com *baud rate* de 19200 b/s.

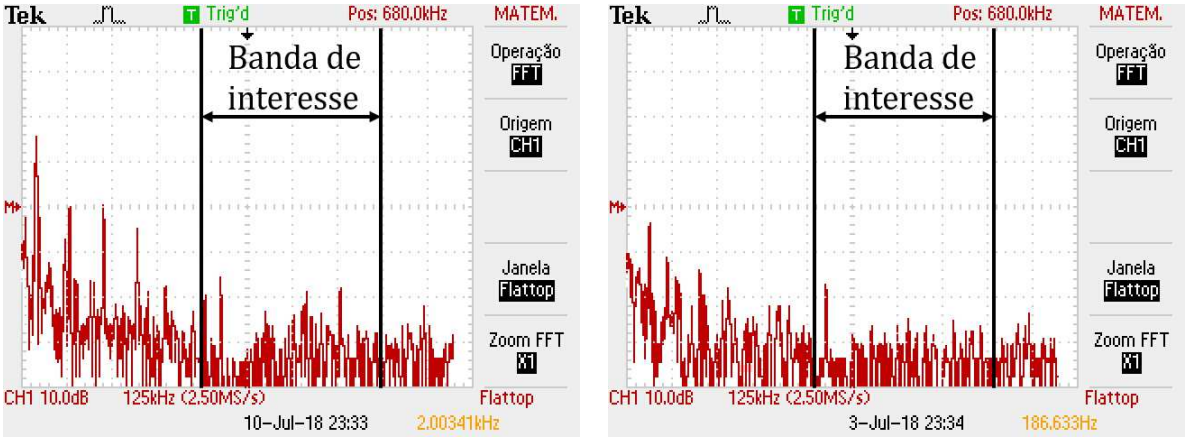
A frequência da portadora encontrada nos três módulos representa a frequência central da banda utilizada. A diferença apresentada entre o Módulo 1 e o Módulo 2 pode ser justificada pela variação de até 5% do DCO de cada microcontrolador, o qual é a fonte de *clock* para o módulo PWM (utilizado para gerar o sinal da portadora).

O algoritmo de seleção de frequência de portadora executado pelo módulo mestre oferece a oportunidade de inserir as frequências de forma direta (sem a necessidade de conhecer o *clock* utilizado pela raspberry pi) além disso as frequências utilizadas foram baseadas na frequência de 21 MHz (*clock* máximo dos módulos escravos). Portanto a diferença do valor de frequência encontrado entre o módulo mestre e os demais módulos também é justificada pela variação do valor de DCO utilizado. O erro máximo encontrado em um período de 24 h foi menor que 1 %.

Tabela 3. Resultado dos testes com o modem PLC na residência em Campinas-SP.

Baud rate (b/s)	Módulos utilizados	Frequência da portadora (kHz)	Ganho do receptor (db)	Distância entre módulos (m)	Taxa de erro (%)
9600	Módulo 1	680	18	15	0
	Módulo 2	694	18	5	0,004
	Mestre	684	18	-	
19200	Módulo 1	680	18	15	0,006
	Módulo 2	694	18	5	0
	Mestre	684	18	-	

O segundo ambiente de teste foi o Laboratório de Instrumentação, Sensores e Sistemas Embarcados – LISSE da Universidade Estadual de Campinas – UNICAMP também em Campinas-SP. Os módulos foram instalados em três tomadas distintas com distância máxima de aproximadamente 30 metros. Assim como no teste anterior, foi verificado o espectro do canal de transmissão antes do início do teste. Nas Figura 38a e Figura 38b são apresentados os conteúdos espectrais do ruído na rede elétrica observado no início do teste. Os resultados desse teste são apresentados na Tabela 4.



(a) (b)

Figura 38. Espectro da rede elétrica do LISSE. (a) início do teste com *baud rate* de 9600 b/s. (b) início do teste com *baud rate* de 19200 b/s.

Apesar da diferença de espectro entre as duas instalações, ambas não apresentaram atenuações na banda de interesse, e por isso, a frequência da portadora encontrada pelo algoritmo manteve-se a mesma. A diferença dos ganhos dos módulos escravos com relação ao módulo mestre pode ser justificada pela diferença dos microcontroladores utilizados; módulos escravos utilizam MSP430 enquanto a Raspberry Pi utiliza microcontrolador ARM. A sensibilidade do pino de I/O para a interrupção de borda de descida pode ser distinta nos diferentes núcleos utilizados, gerando ganhos diferentes. Outra justificativa é a variação temporal do ruído no canal de comunicação: os módulos podem ter visto níveis de ruídos diferentes no momento da execução do algoritmo de ajuste de ganho. Nesse teste foi atingida a distância máxima avaliada: 30 metros. Em ambas taxas de transmissão, o erro encontrado para o período observado foi menor que 1 %.

Tabela 4. Resultado dos testes com o Modem PLC no LISSE

Baud rate (b/s)	Módulos Utilizados	Frequência Portadora (kHz)	Ganho recepção (db)	Distância central e módulo (m)	Taxa de erro (%)
9600	Módulo 1	680	8	30	0,009
	Módulo 2	694	8	5	0,003
	Mestre	684	18	-	
19200	Módulo 1	680	8	30	0,036
	Módulo 2	694	8	5	0,003
	Mestre	684	18	-	

O terceiro ambiente de teste foi a residência localizada na cidade de Santo André – SP, cuja rede elétrica possui tensão nominal de 115 V_{RMS}. Os módulos foram instalados em três tomadas distintas com distância máxima de aproximadamente 20 metros. Neste ambiente de teste, o chaveamento de um inversor de frequência, o qual faz parte de um sistema de geração fotovoltaico, teve grande influência nos resultados. Assim como o anterior, foi verificada a densidade espectral de ruído do canal de transmissão

antes do início do teste. Nas Figura 39a e Figura 39b são apresentados o conteúdo espectral observado no início do teste em ambas taxas de transmissão, com o inversor de frequência em funcionamento. Para efeito de comparação, a Figura 39c apresenta o espectro do canal após o desligamento do inversor de frequência. Os resultados desses experimentos são apresentados na Tabela 5.

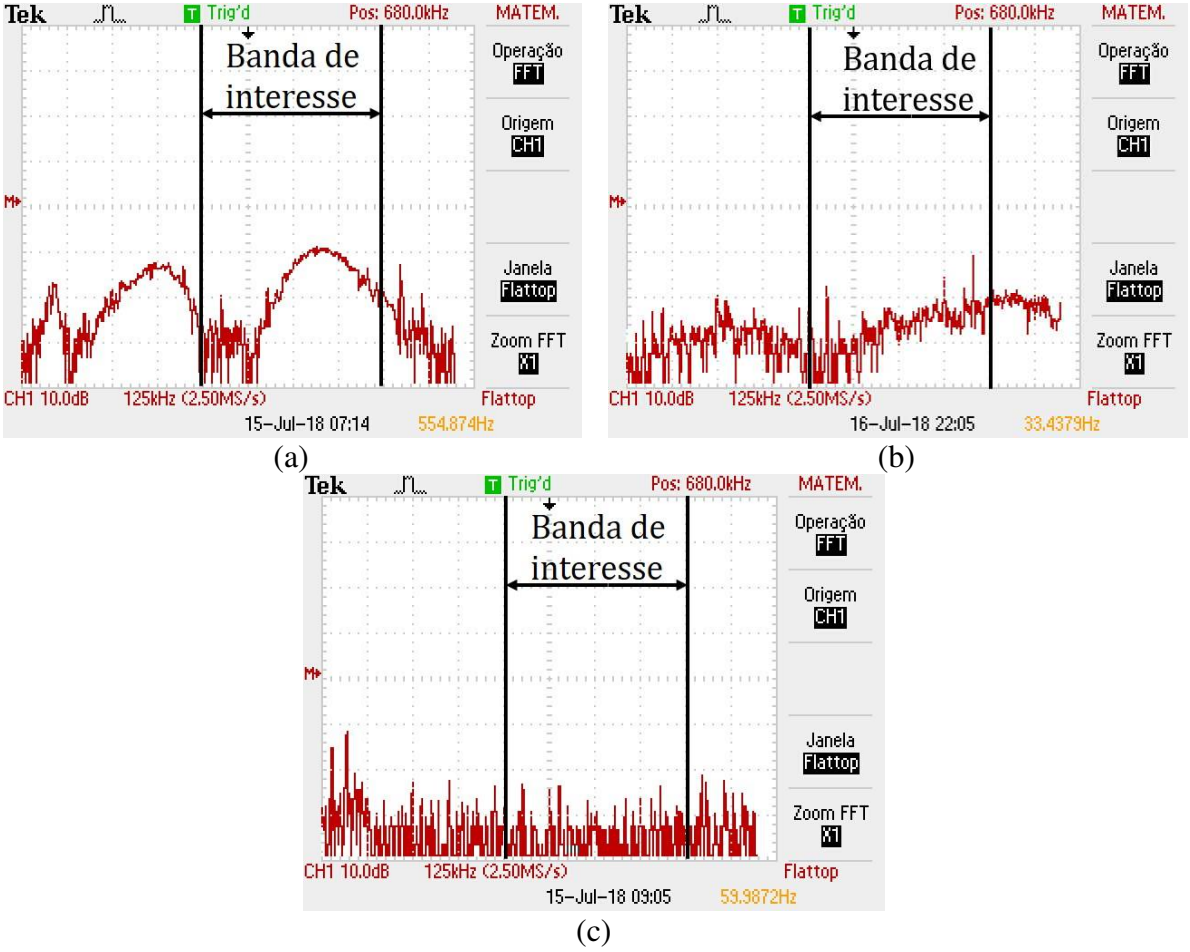


Figura 39. Espectro da rede elétrica da residência em Santo André-SP: (a) *baud rate* de 9600 b/s, (b) *baud rate* de 19200 b/s. (c) Inversor de frequência desligado.

O espectro apresentado nas Figura 39a e Figura 39b é justificada pelo chaveamento do inversor de frequência. Durante a noite, quando o inversor está desligado, o espectro tem a forma apresentada na Figura 39c. Este comportamento espectral foi responsável pelo aumento da perda de pacotes em relação aos outros cenários, uma vez que parte do conteúdo espectral do inversor de frequência interfere na banda de comunicação. Contudo, o alto nível de ruído na banda não impediu que um dos módulos escravos utilizasse o maior ganho enquanto o segundo módulo e o módulo central selecionaram o menor ganho. A diferença de ganho neste cenário pode estar

relacionada com a variação temporal do ruído do canal de comunicação: o ruído pode ter variado entre as formas apresentadas nas Figura 39a e Figura 39b em um breve intervalo de tempo. Dessa forma, os módulos podem ter observados ruídos diferentes durante a execução do algoritmo de ajuste de ganho, resultando em configurações de ganho diferentes. O nível de ruído também interferiu na seleção de frequência; enquanto que na *baud rate* de 9600 b/s, a frequência escolhida foi de 680 kHz para o módulo 1 e 694 kHz para o módulo 2; para 19200 b/s, a frequência da portadora escolhida pelo algoritmo foi de 667 kHz para o módulo 1 e 680 kHz para o módulo 2. No ambiente mais ruidoso, foi observado erro máximo de 12,54 % para *baud rate* de 9600 b/s e 19,76 % para *baud rate* de 19200 b/s.

Tabela 5. Resultado dos testes com o Modem PLC na residência em Santo André

Baud rate (b/s)	Módulos Utilizados	Frequência Portadora (kHz)	Ganho recepção (db)	Distância central e módulo (m)	Taxa de erro (%)
9600	Módulo 1	680	18	20	12,54
	Módulo 2	694	8	5	12,18
	Mestre	680	8	-	
19200	Módulo 1	667	8	20	19,76
	Módulo 2	680	8	5	18,23
	Mestre	663	8	-	

5.2 Medidor de Energia

Foi desenvolvido um programa em LabVIEW com o objetivo de calibrar e testar o sistema de medição das grandezas elétricas. A interface desse programa é apresentada na Figura 40.

As amostras referentes a um ciclo de rede e os parâmetros elétricos calculados pelo sistema de medição são enviados serialmente do modulo medidor PLC para o programa LabVIEW desenvolvido, utilizando *baud rate* de 9600 b/s. A interface do programa desenvolvido apresenta as formas de onda de tensão e corrente elétrica

(Counts x Samples) e os valores das grandezas elétricas processadas pelo módulo medidor: tensão e corrente eficazes, potência ativa e aparente, fator de potência e energia consumida.

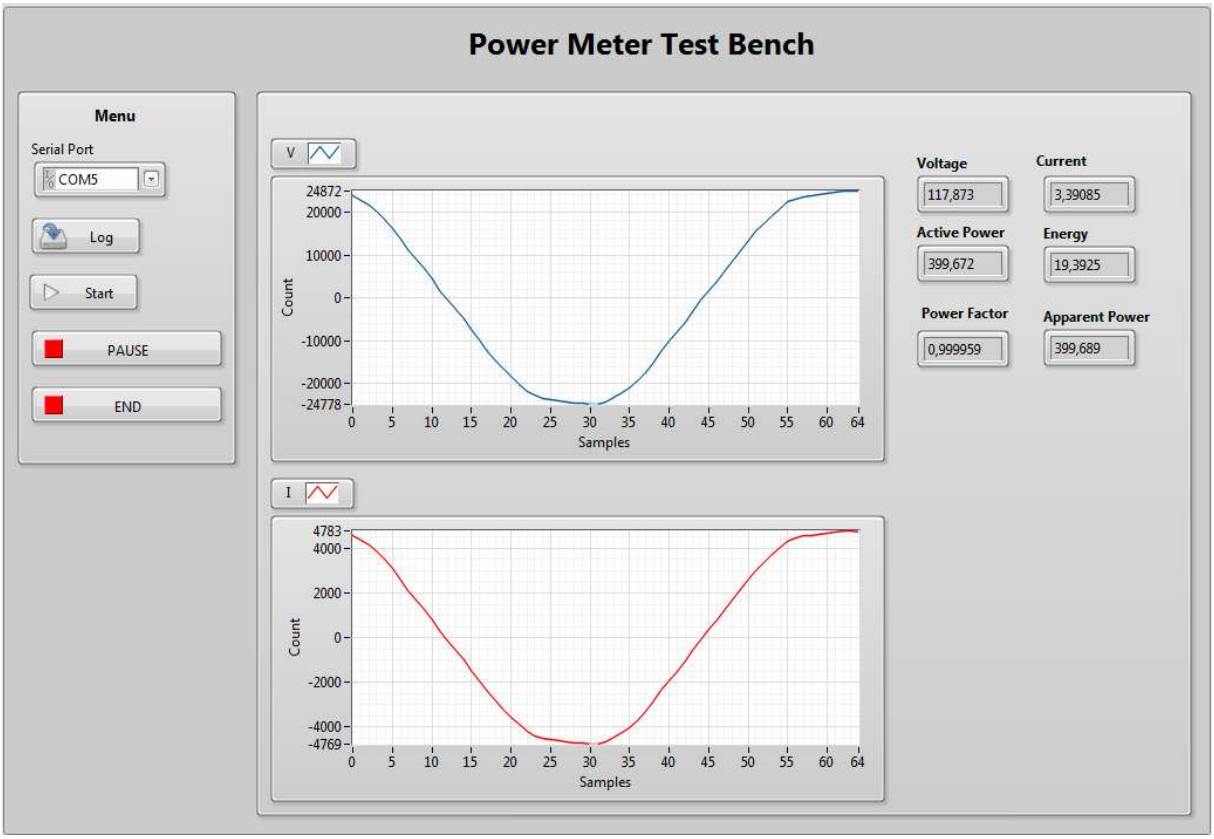


Figura 40. Interface do programa em LabVIEW para calibração e teste do sistema de medição de energia.

Inicialmente, foram realizados testes para verificar a faixa dinâmica dos sensores de tensão e corrente utilizados. Os testes consistem em verificar a pertubação de circuitos eletrônicos vizinhos (alimentação, PLC) nos circuitos de aquisição de tensão e corrente e relacionar o resultado com o fundo de escala do conversor A/D.

Para o teste com o sensor de tensão, foi retirado o resistor R11 (Figura 16) de modo que a entrada do conversor A/D ficasse aberta. Em seguida, foi realizada a comunicação entre o módulo medidor e o programa LabVIEW. As amostras de tensão obtidas foram enviadas para o programa LabVIEW e representam a medição de ruídos que as portas do conversor A/D estão sujeitas.

A mesma análise foi realizada com o sensor de corrente; nenhuma carga foi ligada no módulo medidor e a comunicação entre o módulo medidor e o programa LabVIEW foi realizada, dessa vez enviando as amostras de corrente.

A faixa dinâmica (DR) é calculada conforme Equação (15):

$$DR = 20\ log\ \frac{V_{FS}}{V_{RUÍDO}}, \tag{15}$$

onde V_{FS} é a tensão de fundo de escala do conversor A/D, $V_{RUÍDO}$ é a tensão do ruído a qual as portas do conversor A/D estão sujeitas.

Desse modo, o sensor de tensão apresenta $V_{RUÍDO} \cong 45\ \mu V_{PICO}$ e $V_{FS} = 500\ mV_{PICO}$, resultando em uma faixa dinâmica de aproximadamente 80 dB. O sensor de corrente apresenta $V_{RUÍDO} \cong 15\ \mu V_{PICO}$ e $V_{FS} = 100\ mV_{PICO}$ ($I_{MIN} \cong 3$ mA e $I_{MÁX} = 20$ A), resultando em uma faixa dinâmica de aproximadamente 76 dB.

A calibração e testes do medidor foram realizados no LISSE. A calibração consiste no ajuste dos ganhos de tensão (K_V) e de corrente (K_A) do módulo medidor. Não é necessário calibrar o *offset*, pois, como discutido na Seção 3.6.1 do Capítulo 3, a compensação de *offset* é feita continuamente durante a operação do sistema.

O multímetro Keithley 197 [53] foi utilizado como referência para medição de tensão e corrente durante a calibraçao. A carga de referência utilizada foi uma resistência de 400 W. O erro após a calibração é apresentado na Tabela 6.

Devido à impossibilidade de medir a potência ativa, foi utilizada como referência a medida realizada no trabalho [12] para cargas não resistivas. As potências ativas das cargas resistivas foram estimadas considerando o fator de potência igual a 1.

Tabela 6. Resultados da calibração com a carga de 400 W

Grandeza Elétrica	Valor de Referência	Valor medido	Desvio Padrão	Erro (%)
Tensão Eficaz (V_{RMS})	117,1	117,44	0,12	0,29
Corrente Eficaz (A_{RMS})	3,45	3,45	0,003	0,0
Potência Ativa (W)	404	405,1	0,77	0,27

Os testes para avaliação da precisão e acurácia do módulo medidor foram realizados com três cargas distintas: motor de 110 W (carga indutiva), lâmpada

fluorescente de 20 W (carga não linear) e lâmpada incandescente de 40 W (carga linear). Os resultados das medidas de tensão, corrente e potência ativa realizadas estão apresentados na Tabela 7, Tabela 8 e Tabela 9

Tabela 7. Resultados das medições do motor

Grandeza Elétrica	Valor de Referência	Valor medido	Desvio Padrão	Erro (%)
Tensão Eficaz (V_{RMS})	122,6	122,52	0,09	0,07
Corrente Eficaz (A_{RMS})	1,93	1,95	0,01	1,07
Potência Ativa (W)	111,67	114,81	0,48	2,81

Os resultados do ensaio realizado com o motor de 110 W apresentados na Tabela 7 apresentam erros superiores a 1 % para a leitura de corrente e potência ativa. Uma possível causa de erro na medição é que a corrente de aproximadamente 2 A_{RMS} representa aproximadamente 14 % do fundo de escala do sensor de corrente. Valores baixos de corrente tendem a diminuir a acurácia do módulo medidor desenvolvido uma vez que não foi implementado uma rotina para alterar o ganho (PGA) do conversor A/D dinamicamente.

Tabela 8. Resultados das medições da lâmpada fluorescente

Grandeza Elétrica	Valor de Referência	Valor medido	Desvio Padrão	Erro (%)
Tensão Eficaz (V_{RMS})	123,3	122,85	0,1	0,36
Corrente Eficaz (A_{RMS})	0,265	0,256	0,001	3,23
Potência Ativa (W)	19,37	18,79	0,10	3,04

Os resultados do ensaio realizado com a lâmpada fluorescente de 20 W apresentados na Tabela 8 apresentam erros superiores a 3% para a leitura de corrente e potência ativa. O erro apresentado pode ser justificado pela não implementação da rotina

de PGA dinâmico. Além disso, o algoritmo de medição desenvolvido utiliza taxa de amostragem (≅ 3096 S/s) que não permite obter um número inteiro de amostras da rede elétrica, contribuindo para o erro da medida.

Tabela 9. Resultados das medições da lâmpada incandescente

Grandeza Elétrica	Valor de Referência	Valor medido	Desvio Padrão	Erro (%)
Tensão Eficaz (V_{RMS})	123,35	122,24	0,11	0,09
Corrente Eficaz (A_{RMS})	0,311	0.317	0,00	1,93
Potência Ativa (W)	38,40	38.87	0,08	1,22

Os resultados do ensaio realizado com a lâmpada incandescente de 40 W apresentados na Tabela 9 apresentam erros superiores a 1 % para a leitura de corrente e potência ativa. O erro na medição desses parâmetros pode estar relacionado ao fato de estar medindo correntes com valores muito baixos em relação ao fundo de escala. Uma corrente de aproximadamente 300 mA representa 2% do fundo de escala do módulo medidor.

Capítulo 6

Conclusões e Trabalhos Futuros

Neste capítulo são sumarizados os principais resultados do sistema de medição proposto, ressaltando suas vantagens e limitações. Também são discutidas e apresentadas propostas de trabalhos futuros que podem contribuir com a melhoria do sistema desenvolvido.

6.1 Conclusões

Nesse trabalho, foi desenvolvido um sistema intrusivo para controle e monitoramento do consumo de energia elétrica. Visando a construção de um sistema simples, eficaz, de fácil instalação e com baixo custo, optou-se pela implementação do sistema de comunicação entre os módulos medidores através da rede elétrica (*Power Line Communication* - PLC). O sistema desenvolvido é composto pelos: Módulos Medidores PLC e Sistema concentrador de dados.

Os Módulos Medidores PLC são responsáveis por realizar a aquisição dos parâmetros elétricos da carga monitorada, controlar seu acionamento e enviar os parâmetros para o módulo concentrador através de comunicação PLC. Este medidor é capaz de monitorar tensão e corrente eficazes e calcular potência ativa, potência aparente, fator de potência e consumo de energia de cargas com consumo de até aproximadamente 1750 W em redes monofásicas de 127 V_{RMS}. Os módulos medidores foram construídos utilizando microcontroladores de baixo custo e consumo. Os sensores de tensão (divisor resistivo) e corrente *(shunt)* foram escolhidos com o objetivo de manter o baixo custo da solução. O acionamento da carga é realizado através de relé. O módulo mestre,

desenvolvido em plataforma Raspberry pi, é responsável por solicitar os dados dos medidores, bem como enviá-los a um serviço web para acesso aos dados pela internet.

Testes foram realizados para verificar a precisão e acurácia dos parâmetros elétricos medidos. Para cargas lineares, foram encontrados erros inferiores a 0,3 % para a leitura de tensão, para a leitura de corrente foram encontrados erros menores que 2 % e para o cálculo de potência ativa foram apresentados erros menores que 3 %. Os erros encontrados para a leitura de corrente e potência ativa podem ser explicados pela baixa potência das cargas monitoradas; o valor de corrente da lâmpada incadescente e do motor representam valores baixos do fundo de escala do sensor (2% e 14%, respectivamente). A carga não linear monitorada apresentou erro de 0,36 % para o valor de tensão, 3,23 % para o valor de corrente e 3,04 % para a potência ativa. A má utilização do fundo de escala e a obtenção de um número não inteiro de amostras são as fontes de erros sugeridas para as leituras do módulo medidor.

Somado às fontes de erros anteriores, deve-se destacar que o resistor *shunt* utilizado não é adequado para a aplicação. O sensor de corrente utilizado não possibilita a utilização de todo o fundo de escala do conversor A/D, dessa forma é necessário utilizar o PGA do conversor A/D para amplificar o sinal de tensão do *shunt,* uma vez que o mesmo gera pequenas tensões para as correntes monitoradas (100 mV_{PICO} no fundo de escala). O uso de PGA insere mais uma fonte de ruído na medição de corrente, que pode ter influenciado na medida. Um *shunt* adequado a essa aplicação é aquele que, assim como o sensor de tensão, possibilita a utilização de todo o fundo de escala disponível pelo conversor A/D e monitora correntes de até 20 A_{RMS}.

Uma alternativa para contornar os erros oriundo a má utilização do fundo de escala é a implementação de uma rotina de seleção de PGA dinâmico, o PGA terá um valor diferente de acordo com a carga monitorada. Dessa forma, cargas com baixo consumo oferecerão valores de tensão próximos ao fundo de escala do conversor A/D.

Como parte principal desse trabalho, foi proposto e construído um modem PLC de baixo custo para aplicações que requerem baixo fluxo de dados, como monitoramento e controle de recursos em ambientes residenciais. O principal diferencial do modem PLC proposto é a utilização dos condutores de neutro e terra como meio de comunicação, eliminando a necessidade de circuitos de isolação complexos, e, consequentemente, diminuindo o custo do sistema. O modem PLC proposto é capaz de transmitir e receber

informações utilizando modulação OOK com frequência de portadora variável de 500 kHz a 1 MHz e ganhos no estágio de recepção de 8 dB a 18 dB. O uso de protocolos e esquemas de modulação simples - não exigindo processamento digital do sinal tampouco filtros complexos e a implementação do circuito eletrônico com componentes discretos (transistores, resistores, capacitores e indutores), contribuíram para o desenvolvimento de um sistema simples e de baixo custo.

Foram realizados testes em laboratório e em campo com o objetivo de avaliar o desempenho do modem PLC desenvolvido. Os testes realizados no laboratório, com rede emulada, mostraram que o modem desenvolvido é capaz de transmitir dados a 9600 b/s com erro inferior a 10 %, quando a densidade espectral de ruído da rede é de 366,67 $\mu V_{RMS}/\sqrt{Hz}$; e com erro superior a 30 % quando a densidade espectral de ruído é de 400 $\mu V_{RMS}/\sqrt{Hz}$. O sistema também é capaz de operar a uma taxa de 19200 b/s, apresentando nesse caso erro superior a 20 % para densidade espectral de ruído de 366,67 $\mu V_{RMS}/\sqrt{Hz}$.

Testes foram realizados em campo, em três diferentes instalações, com o objetivo de verificar a robustez do sistema em condições operacionais. Em duas das três instalações as taxas de erro nas comunicações foram baixas, inferiores a 1% para ambos os módulos observados utilizando as taxas de transmissão de 9600 b/s e 19200 b/s. Foi observado que a distância entre os módulos medidores e o módulo mestre exerceu influência no resultado da segunda instalação: para a taxa de transmissão de 9600 b/s o módulo mais distante obteve taxa de erro três vezes maior em relação ao módulo mais próximo; para a taxa de transmissão de 19200 b/s o módulo mais distante obteve taxa de erro doze vezes maior em relação ao módulo mais próximo. Em uma das instalações, observou-se uma taxa de erro de 12% para a taxa de transmissão de 9600 b/s e erros superiores a 18 % para a taxa de transmissão de 19200 b/s. Esse alto índice de erros se deve ao fato da presença de um inversor de frequência nessa instalação. Não foi possível avaliar a influência da parcela de erro oriunda da distância dos módulos.

6.2 Trabalhos Futuros

Como trabalho futuro pretende-se aperfeiçoar os algoritmos de seleção de frequência, ajuste de ganho do circuito receptor e medição de energia elétrica. Além disso, a utilização de uma frequência estável como fonte da portadora PLC, o desenvolvimento

de um Sistema concentrador de dados utilizando conceitos de Engenharia de software, o projeto adequado do sensor de corrente e o estudo da viabilidade de novas implementações com o modem PLC proposto são objetos de trabalhos futuros.

O aperfeiçoamento dos algoritimos consiste em criar uma rotina de manutenção dos módulos medidores. Uma mensagem *broadcast* é transmitida periodicamente para os módulos medidores, solicitando a recalibração da frequência da portadora, do ganho do circuito de recepção e a reinicialização do valor de energia elétrica consumida. A recalibração períodica dos parâmetros de frequência e ganho podem garantir menores taxas de erro na comunicação, uma vez que os parâmetros recalculados poderiam minimizar a influência da variação temporal do ruído presente na rede elétrica. A reinicialização do valor de energia consumida permite maior confiabilidade no valor de energia calculado em comparação com o método atual, em que o valor é reinicializado somente com o desenergizamento do módulo medidor.

Além disso, o algoritmo de medição de energia elétrica pode ser melhorado com utilização de um cristal oscilador cujo valor de frequência é múltiplo da frequência da rede elétrica, permitindo obter um número inteiro de amostras. O desenvolvimento de uma rotina para escolher o PGA adequado para a carga monitorada pode melhorar a utilização do fundo de escala. Ademais, o processamento de mais ciclos da rede para o cálculo dos parâmetros elétricos pode contribuir para a redução de erro dos parâmetros elétricos calculados.

O Sistema concentrador de dados apresentado nesse trabalho teve o objetivo de apenas validar o módulo PLC proposto. O desenvolvimento de um Sistema concentrador de dados utilizando conceitos de Engenharia de software agregaria valor à solução proposta nesse trabalho.

O melhor projeto do sensor de corrente de modo a utilizar todo fundo de escala disponível pelo convesor A/D diminuiria a contribuição do ruído adicionado pelo PGA na medida de corrente.

A redução do hardware do módulo medidor PLC a dimensões que permita que o mesmo seja fixado em caixas de passagem de tomadas de uso comum oferecem a possibilidade de embuti-los na parede, garantindo economia de espaço. Além disto, o estudo de viabilidade de embarcar o modem PLC desenvolvido em novos

eletrodomésticos garantiria que todos eletrodomésticos em uma residência estariam conectados a uma rede local e seriam passiveis de controle e monitoramento.

Referências

[1] British Petroleum, "BP Statistical Review of World Energy 2017", *Br. Pet.*, nº 66, p. 1–52, 2017.

[2] I. E. AGENCY, "CO2 emissions from fuel combustion", 2016.

[3] "Electric power consumption (kWh per capita) | Data". [Online]. Available at: https://data.worldbank.org/indicator/EG.USE.ELEC.KH.PC?end=2014&locations=BR&start=1971&view=chart. [Acessado: 09-jul-2018].

[4] "Energia no Brasil e no mundo", p. 1–13, 2004.

[5] "LEI No 10.295, DE 17 DE OUTUBRO DE 2001." [Online]. Available at: https://www.planalto.gov.br/ccivil_03/leis/leis_2001/l10295.htm. [Acessado: 09-jul-2018].

[6] EPE, "Projeção da demanda de energia elétrica: 2017 - 2016". p. 95, 2017.

[7] A. K. Meier, "A worldwide review of standby power use in homes", *Lawrence Berkeley Natl. Lab.*, p. 1–5, 2001.

[8] "Standby and off-mode - European Commission". [Online]. Available at: https://ec.europa.eu/energy/en/topics/energy-efficiency/energy-efficient-products/standby. [Acessado: 07-dez-2018].

[9] D. Minoli, K. Sohraby, e B. Occhiogrosso, "IoT Considerations, Requirements, and Architectures for Smart Buildings – Energy Optimization and Next Generation Building Management Systems", *IEEE Internet Things J.*, vol. 4, nº 1, p. 269–283, 2017.

[10] M. Rizzi, P. Ferrari, A. Flammini, e E. Sisinni, "Evaluation of the IoT LoRaWAN Solution for Distributed Measurement Applications", *IEEE Trans. Instrum. Meas.*, vol. 66, nº 12, p. 3340–3349, 2017.

[11] G. Bedi, G. K. Venayagamoorthy, R. Singh, R. R. Brooks, e K.-C. Wang, "Review of Internet of Things (IoT) in Electric Power and Energy Systems", *IEEE Internet Things J.*, vol. 5, nº 2, p. 847–870, 2018.

[12] R. M. Bacurau, "Medidor de energia inteligente para discriminação de consumo por aparelho através de assinatura de cargas", Dissertação (Mestrado em Engenharia Elétrica), Universidade de Campinas, 2014.

[13] D. York *et al.*, "Frontiers of Energy Efficiency: Next Generation Programs Reach for High Energy Savings", 2013.

[14] J. Carroll, S. Lyons, e E. Denny, "Reducing household electricity demand through smart metering: The role of improved information about energy saving", *Energy Econ.*, vol. 45, p. 234–243, 2014.

[15] A. E. Efficiency, "Energy Savings Through Consumer Feedback Programs", nº February. p. 1–17, 2014.

[16] Y. Thongkhao e W. Pora, "A low-cost Wi-Fi smart plug with on-off and Energy Metering functions", *2016 13th Int. Conf. Electr. Eng. Comput. Telecommun. Inf.*

Technol. ECTI-CON 2016, 2016.

[17] A. Ridi, C. Gisler, e J. Hennebert, “A survey on intrusive load monitoring for appliance recognition”, *Proc. - Int. Conf. Pattern Recognit.*, p. 3702–3707, 2014.

[18] Zuli, “Zuli Smartplug”. [Online]. Available at: https://zuli.io/. [Acessado: 09-jul-2018].

[19] Apple, “iDevices Switch | Connected Plug”. [Online]. Available at: https://store.idevicesinc.com/idevices-switch/. [Acessado: 09-jul-2018].

[20] D-Link, “Wi-Fi Smart Plug with thermal protection and energy management | D-Link”. [Online]. Available at: http://us.dlink.com/products/connected-home/wi-fi-smart-plug/. [Acessado: 09-jul-2018].

[21] M. Altmann, P. Schlegl, e K. Volbert, “A low-power wireless system for energy consumption analysis at mains sockets”, *Eurasip J. Embed. Syst.*, vol. 2017, nº 1, 2017.

[22] E. Chobot, D. Newby, R. Chandler, N. Abu-Mulaweh, C. Chen, e C. Pomalaza-Ráez, “DESIGN AND IMPLEMENTATION OF AWIRELESS SENSOR AND ACTUATORNETWORK FOR ENERGY MEASUREMENT AND CONTROL ATHOME”, vol. 3, nº 5, p. 93–109, 2012.

[23] H. Morsali e S. Shekarabi, “Smart plugs for building energy management systems”, *Smart Grids (ICSG)*, vol. 1, nº 1, p. 2–6, 2012.

[24] C.-H. Lien, H.-C. Chen, Y.-W. Bai, e M.-B. Lin, “Power Monitoring and Control for Electric Home Appliances Based on Power Line Communication”, *2008 IEEE Instrum. Meas. Technol. Conf.*, p. 2179–2184, 2008.

[25] C. Cano, A. Pittolo, D. Malone, L. Lampe, A. M. Tonello, e A. Dabak, “State-of-the-art in Power Line Communications: from the Applications to the Medium”, vol. 34, nº 7, p. 1935–1952, 2016.

[26] T. R. Oliveira, A. A. M. Picorone, S. L. Netto, e M. V. Ribeiro, “Characterization of Brazilian in-home power line channels for data communication”, *Electr. Power Syst. Res.*, vol. 150, p. 188–197, 2017.

[27] S. Barker, D. Irwin, e P. Shenoy, “Pervasive Energy Monitoring and Control Through Low-Bandwidth Power Line Communication”, *IEEE Internet Things J.*, vol. 4, nº 5, p. 1349–1359, 2017.

[28] Powerline Alliance, “HomePlug AV White Paper”. 2005.

[29] ANEEL, “Resolução Normativa Nº 375, de 25 de agosto de 2009”. p. 4, 2009.

[30] ANATEL, “Resolução nº 527, de 8 de abril de 2009”. p. 2009, 2009.

[31] M. Katayama, T. Yamazato, e H. Okada, “A Mathematical Model of Noise in Narrowband Power Line Communication Systems”, *IEEE J. Sel. Areas Communictiones*, vol. 24, nº 7, p. 1267–1276, 2006.

[32] D. Chariag, D. Guezgouz, J. C. Le Bunetel, e Y. Raingeaud, “Modeling and simulation of temporal variation of channel and noise in indoor power-line network”, *IEEE Trans. Power Deliv.*, vol. 27, nº 4, p. 1800–1808, 2012.

[33] M. Zimmermann e Dostert Klaus, “Analysis and Modeling of Impulsive Noise in Broad-Band Powerline Communications”, *IEEE Trans. Electromagn. Compat.*, vol. 44, nº 1, p. 377–386, 2002.

[34] D. S. Kim, S. Y. Lee, K. Y. Wang, J. C. Choi, e D. J. Chung, “A power line

communication modem based on adaptively received signal detection for networked home appliances", *IEEE Trans. Consum. Electron.*, vol. 53, nº 3, p. 864–870, 2007.

[35] Y. T. Hwang e S. W. Chen, "Hardware efficient design for narrow band power line communication modem", *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, p. 508–512, 2011.

[36] Y. F. Chen e T. D. Chiueh, "Baseband transceiver design of a 128-Kbps power-line modem for household applications", *IEEE Trans. Power Deliv.*, vol. 17, nº 2, p. 338–344, 2002.

[37] T. Instruments, "MIXED SIGNAL MICROCONTROLLER 1 MSP430AFE2x3 MSP430AFE2x2 MSP430AFE2x1", nº March. 2011.

[38] T. Instruments, "TS5A2066 Dual-Channel 10- Ω SPST Analog Switch". 2017.

[39] H. Electronics, "PCB Relay — JQC-3F (T73)". p. 73.

[40] T. Instruments, "UCC28910, UCC28911 High-Voltage Flyback Switcher with Primary-Side Regulation and Output Current Control". 2016.

[41] T. Instruments, "CAP-FREE NMOS 400 mA LOW-DROPOUT REGULATORS WITH REVERSE CURRENT PROTECTION", nº February. 2009.

[42] D. Ji, "Non-Isolated High-Side Buck Converter with UCC28910", nº June. p. 1–13, 2016.

[43] R. Erickson, "The Flyback Converter - Lecture notes ECEN4517", nº c, 2012.

[44] R. Patel e G. Fritz, "Switching power supply design review 60 watt flyback regulator", *Unitrode Corporation. Disponível em:< http://www. unitrode. com/> acesso em*, vol. 5. 2009.

[45] M. C. Sarofim, "Fly-Back Converter", *Science*. p. 1–39.

[46] V. Dale, "Vishay Dale Power Metal Strip ® Resistors , High Power (7 W) Low Value , Surface Mount WSHM2818", 2010.

[47] ABNT, "ABNT NBR 14136:2002 - Plugues e tomadas para uso domestico e análogo até 20 A/250 V em corrente alternada — Padronização", p. 20, 2002.

[48] "MQTT". [Online]. Available at: http://mqtt.org/. [Acessado: 09-jul-2018].

[49] "Raspberry Pi Foundation - About Us". [Online]. Available at: https://www.raspberrypi.org/about/. [Acessado: 20-mar-2018].

[50] Keysight Techonogies, "Keysight 33220A. 20 MHz function/arbitrary waveform generator", *Data Sheet*, 2015.

[51] ABNT, "Instalações elétricas de baixa tensão", 2004.

[52] W. Drive, "Test Equipment Solutions Datasheet", vol. 44, nº 0, p. 6–10.

[53] K. Instruments, "Instruction Manual Model 197 Autoranging Microvolt DMM", nº 197.

Apêndice A – Software desenvolvido no serviço de hospedagem web

```
from flask import Flask, render_template
from flask.ext.sqlalchemy import SQLAlchemy
from sqlalchemy import create_engine
from time import gmtime, strftime, strptime
from datetime import datetime
import MySQLdb
from flask_restful import Resource, Api
import pytz

app = Flask(__name__)
api = Api(app)

db_connect = create_engine('mysql://delso:adelson123@delso.mysql.pythonanywhere-services.com:3306/delso$RemoteMetermysql')


        ##Start of creation of dynamic web pages ##
## Web Homepage


@app.route('/')
def index():
  return render_template('HomepageV4.html')
## Page of Meter parameters
#    # Insert Data code 1 = voltage (V)
#    # Data code 2 = current (A)
#    # Data code 3 = power factor
#    # Data code 4 = tot wh (wh)
#    # Data code 5 = Pot (W)
```

```
#    # Data code 6 = ReacPower (var)
@app.route('/Medidor<int:MedidorId>')
def Medidor_index(MedidorId):
   MeteringData = []
   LastMeteringId = []
   MeterId = MedidorId
   try:
      conn = db_connect.connect()
      query = conn.execute("Select Max(Id) from Metering where MeterId = %i" %
(MedidorId))
      for i in query.cursor.fetchall():
         LastMeteringId = i[0]
      if LastMeteringId == None:
         return render_template('ErrorpageV4.html')
      query = conn.execute("Select TimeStamp from Metering where Id = %i" %
(LastMeteringId))
      for i in query.cursor.fetchall():
         LastTimeStamp = i[0]
      _formatedTime = FormatTime(LastTimeStamp)
      query = conn.execute("select Value from MeteringData where MeteringId = %i " %
LastMeteringId)
      for i in query.cursor.fetchall():
         MeteringData = MeteringData + [i[0]]
      if MeterId == 4:
         return render_template('MedidorGeralV4.html', Voltage1 = MeteringData[0],
Current1 = MeteringData[1],
                        ActiveEnergy = (MeteringData[3] + MeteringData[8] +
MeteringData[13]),
                        ReativeEnergy = (MeteringData[4] + MeteringData[9] +
MeteringData[14]),
                        Voltage2=MeteringData[5], Current2=MeteringData[6],
                        Voltage3=MeteringData[10], Current3=MeteringData[11],
                        Time = _formatedTime, MeterId = MeterId )
      else:
         return render_template('MedidoresTemplateV4.html', Voltage=MeteringData[0],
Current=MeteringData[1],
                           Factor=MeteringData[2],          ActiveEnergy=MeteringData[3],
```

```
ReativeEnergy=MeteringData[4],
ActivePower=MeteringData[5],ReactivePower=MeteringData[6],Time=_formatedTime,
MeterId=MeterId)
   finally:
      conn.close()
## data history consult page
@app.route('/Medidor<int:MedidorId>/Historico<int:Info>')
def Info(MedidorId,Info):
   _info = Info
   return render_template('HistoricoV4.html',_infoname = InfoName(_info))


         ## Start 3-Phase ##
@app.route('/Medidor<int:MedidorId>/Historico<int:InfoGeral>Fase<int:Fase>')
def GeralInfo(MedidorId,InfoGeral,Fase):
   _infogeral = InfoGeral
   return render_template('HistoricoV4.html',_infoname = InfoName(_infogeral))
@app.route('/Medidor<int:MedidorId>/Historico<int:InfoGeral>/inicio<string:dataStar
t>fim<string:dataEnd>')
def GeralConsulta(MedidorId, InfoGeral, dataStart,dataEnd):
   _info = InfoGeral
   MeteringId = []
   values = []
   period = []
   try:
      conn = db_connect.connect()
      query = conn.execute("SELECT Id FROM Metering WHERE MeterId = '%i' AND
(TimeStamp BETWEEN '%s' AND '%s')" %
                  (MedidorId, dataStart, dataEnd))
      for i in query.cursor.fetchall():
         MeteringId = MeteringId + [i[0]]
      for z in MeteringId:
         query = conn.execute("SELECT Value FROM MeteringData WHERE MeteringId =
'%i' AND Data = '%i' "
                     "AND MeteringMeterId = '%i'" % (z, InfoGeral, MedidorId))
         for i in query.cursor.fetchall():
            values = values + [i[0]]
```

```
        query = conn.execute("SELECT TIMESTAMP FROM Metering WHERE Id = '%i'" %
(z))
        for y in query.cursor.fetchall():
            _formatedTime = FormatTime(y[0])
            period = period + [_formatedTime]
    return render_template('ConsultaV4.html', values=values, period=period,
_infoname=InfoName(_info))
  finally:
    conn.close()
.....
.....
.....
.....
.....## End 3-Phase ##


## result page of history consult
@app.route('/Medidor<int:MedidorId>/Historico<int:Info>/inicio<string:dataStart>fim
<string:dataEnd>')
def Consulta(MedidorId, Info, dataStart,dataEnd):
  _info = Info
  MeteringId = []
  values = []
  period = []
  try:
    conn = db_connect.connect()
    query = conn.execute("SELECT Id FROM Metering WHERE MeterId = '%i' AND
(TimeStamp BETWEEN '%s' AND '%s')" % (MedidorId, dataStart, dataEnd))
    for i in query.cursor.fetchall():
      MeteringId = MeteringId + [i[0]]
    _tamanhoDados = len(MeteringId)
    for z in MeteringId:
      query = conn.execute("SELECT Value FROM MeteringData WHERE MeteringId =
'%i' AND Data = '%i' "
                           "AND MeteringMeterId = '%i'" % (z, Info, MedidorId))
      for i in query.cursor.fetchall():
        values = values + [i[0]]
```

```
        query = conn.execute("SELECT TIMESTAMP FROM Metering WHERE Id = '%i'" %
(z))
        for y in query.cursor.fetchall():
            _formatedTime = FormatTime(y[0])
            period = period + [_formatedTime]
    return render_template('ConsultaV4.html', values=values, period=period,
_infoname=InfoName(_info), _length = _tamanhoDados)
  finally:
    conn.close()


def InfoName(info):
  _info = info
  _infoname = ''
  if _info == 1:
    _infoname = 'Tensão'
  if _info == 2:
    _infoname = 'Corrente'
  if _info == 3:
    _infoname = 'Fator de Potência'
  if _info == 4:
    _infoname = 'Energia Ativa (Consumo)'
  if _info == 5:
    _infoname = 'Energia Reativa'
  if _info == 11:
    _infoname = 'Tensão Fase 1'
  if _info == 21:
    _infoname = 'Corrente Fase 1'
  #if _info == 31:
  #   _infoname = 'Fator de Potência'
  if _info == 41:
    _infoname = 'Energia Ativa'
  if _info == 51:
    _infoname = 'Energia Reativa'
  if _info == 12:
```

```
        _infoname = 'Tensão Fase 2'
    if _info == 22:
        _infoname = 'Corrente Fase 2'
    #if _info == 32:
    #    _infoname = 'Fator de Potência'
    #if _info == 42:
    #    _infoname = 'Energia Ativa (Consumo) Fase 2'
    #if _info == 52:
    #    _infoname = 'Reativa Fase 2'
    if _info == 13:
        _infoname = 'Tensão Fase 3'
    if _info == 23:
        _infoname = 'Corrente Fase 3'
    #if _info == 33:
    #    _infoname = 'Fator de Potência'
    #if _info == 43:
    #    _infoname = 'Energia Ativa (Consumo) Fase 3'
    #if _info == 53:
    #    _infoname = 'Reativa Fase 3'
    return _infoname

def FormatTime(dbTimeFormat):
    local_tz = pytz.timezone('America/Sao_Paulo')
    local_dt = dbTimeFormat.replace(tzinfo=pytz.utc).astimezone(local_tz)
    _formatedTime = local_dt.strftime('%d-%m-%Y %H:%M:%S')
    return _formatedTime

                ##End of the creation of dynamic web pages ##

.....
.....
.....
.....
.....## Start Rest API ##
```

```
class MeterName(Resource):
    def get(self):
        MeterName = []
        try:
            conn = db_connect.connect()
            query = conn.execute("SELECT Name, Address FROM Meter")
            for i in query.cursor.fetchall():
                MeterName = MeterName + [i[0], i[1]]
            return MeterName
        finally:
            conn.close()

## API entrance to Meter table
class Meter(Resource):
    def get(self):
        Meter = []
        try:
            conn = db_connect.connect()
            query = conn.execute("SELECT * FROM Meter")
            for i in query.cursor.fetchall():
                Meter = Meter + [[i[0], i[1], i[2], i[3]]]
            return Meter
        finally:
            conn.close()

## API entrance to metering table
class Metering(Resource):
    def get(self):
        Metering = []
        try:
            conn = db_connect.connect()
            query = conn.execute("select Id, MeterId, TimeStamp, MeterFlag  from Metering;")
            for i in query.cursor.fetchall():
                Metering = Metering + [[i[0], i[1], i[2].strftime('%d-%m-%Y %H:%M:%S'), i[3]]]
            return Metering
```

```
        finally:
            conn.close()


## API entrance to MeteringData table
class MeteringData(Resource):
    def get(self, LastId):
        MeteringData = []
        try:
            conn = db_connect.connect()
            query = conn.execute("select MeteringId, Data, Value from MeteringData where
MeteringId = %i " % LastId)
            for i in query.cursor.fetchall():
                MeteringData = MeteringData + [[i[0], i[1], i[2]]]
            return MeteringData
        finally:
            conn.close()


## API entrance to change Meter table, column Userflag
class ChangeUserFlag(Resource):
    def get(self, Address):
        try:
            conn = db_connect.connect()
            query = conn.execute("UPDATE Meter SET UserFlag = 1 WHERE Address = %s" %
(Address))
            return 'User Flag = 1 para o Endereço'
        finally:
            conn.close()


## DEBUG purpose
class ChangeUserFlagTeste(Resource):
    def get(self, Address):
        try:
            conn = db_connect.connect()
            query = conn.execute("UPDATE Meter SET UserFlag = 0 WHERE Address = %s" %
(Address))
            return 'User Flag = 0 para o Endereço'
```

```
        finally:
            conn.close()


## API entrance to change Meter table, column MeterFlag
class ChangeMeterFlag1(Resource):
    def get(self, Address):
        try:
            conn = db_connect.connect()
            query = conn.execute("UPDATE Meter SET MeterFlag = 1 WHERE Address = %s" %
(Address))
            return 'Meter Flag = 1 para o Endereço'
        finally:
            conn.close()


class ChangeMeterFlag0(Resource):
    def get(self, Address):
        try:
            conn = db_connect.connect()
            query = conn.execute("UPDATE Meter SET MeterFlag = 0 WHERE Address = %s" %
(Address))
            return 'Meter Flag = 0 para o Endereço'
        finally:
            conn.close()


## API entrance to change Meter table, column Mode
class NormalMode(Resource):
    def get(self, MeterId):
        try:
            conn = db_connect.connect()
            query = conn.execute("UPDATE Meter SET Mode = 1 WHERE Address = %i" %
MeterId)
            return 'Mode = 1 para o Endereço'
        finally:
            conn.close()
```

```
class StandByMode(Resource):
   def get(self, MeterId):
      try:
         conn = db_connect.connect()
         query = conn.execute("UPDATE Meter SET Mode = 2 WHERE Address = %i" %
MeterId)
         return 'Mode = 2 para o Endereço'
      finally:
         conn.close()

## API entrance to return Meter parameters
class Mode(Resource):
   def get(self, MeterId):
      Meter = []
      try:
         conn = db_connect.connect()
         query = conn.execute("Select Id, Mode from Meter WHERE Id = %i" % MeterId)
         for i in query.cursor.fetchall():
            Meter = Meter + [i[0], i[1]]
         return Meter
      finally:
         conn.close()

## API entrance to return LastMeteringId from Metering table
class LastMeteringId(Resource):
   def get(self, MeterId):
      LastMeteringId = []
      try:
         conn = db_connect.connect()
         query = conn.execute("Select MeterId, Max(Id) from Metering where MeterId = %i"
% MeterId)
         for i in query.cursor.fetchall():
            LastMeteringId = LastMeteringId + [[i[0], i[1]]]
         return LastMeteringId
      finally:
```

```
        conn.close()

## API entrance to automatic turn off meter in standby mode
class LastestMetering(Resource):
   def get(self, MeterId, Iminimum):
      DataId = []
      relayIntermedValue = []
      relayValue = [0] *10
      iCount = 0
      try:
         conn = db_connect.connect()
         query = conn.execute("""SELECT Id from Metering WHERE MeterId = "%i" ORDER
BY TimeStamp DESC Limit 5;""" % (MeterId))
         for i in query.cursor.fetchall():
            DataId = DataId + [i]
         for finddata in DataId:
            query = conn.execute("""SELECT Value from MeteringData WHERE MeteringId =
"%i" AND Data = 2;""" %(finddata))
            for j in query.cursor.fetchall():
               relayIntermedValue = relayIntermedValue + [j]
         relayValue[MeterId] = relayIntermedValue

         if(len(relayValue[MeterId]) == 5):
            for iValue in relayValue[MeterId]:
               if ( iValue[0] < Iminimum):
                  iCount = iCount + 1
               if iCount == 5:
                  return 'D'
            return "CORRENTE > QUE VALOR MINIMO"
         return "NAO HÁ AMOSTRAS SUFICIENTES"

      finally:
         conn.close()

## API entrance to return LastMeteringId from Metering table
```

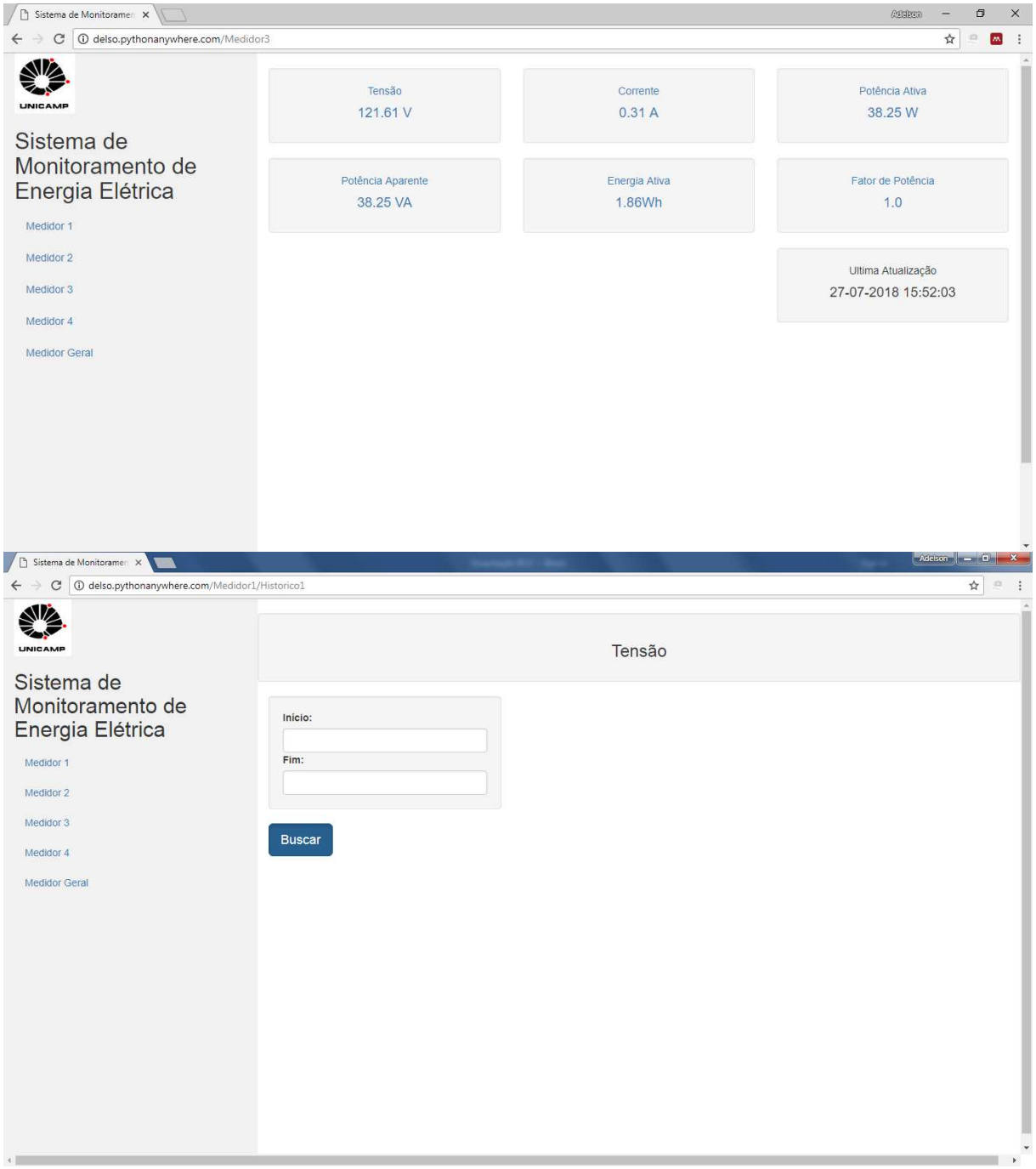
```
class LastMeteringIdRasp(Resource):
  def get(self):
    LastMeteringId = []
    try:
      conn = db_connect.connect()
      query = conn.execute("Select Max(Id) from Metering")
      for i in query.cursor.fetchall():
        LastMeteringId = i[0]
      return LastMeteringId
    finally:
      conn.close()


## API entrance to insert data on database
class InserirDados(Resource):
  def get(self, Id, MeteringMeterId, Voltage, VoltageValue, Current, CurrentValue, PowerFactor, PowerFactorValue, ActEnergy, ActEnergyValue, ReactiveEnergy, ReactiveEnergyValue, ActPower, ActPowerValue, ReactPower, ReactPowerValue):
    sql =''
    try:
      conn = db_connect.connect()
      sql = "INSERT INTO MeteringData (MeteringId, Data, Value, MeteringMeterId) VALUES "
      sql = sql + "('%i','%i','%f','%i'), " % (Id, Voltage, VoltageValue, MeteringMeterId)
      sql = sql + "('%i','%i','%f','%i'), " % (Id, Current, CurrentValue, MeteringMeterId)
      sql = sql + "('%i','%i','%f','%i'), " % (Id, PowerFactor, PowerFactorValue, MeteringMeterId)
      sql = sql + "('%i','%i','%f','%i'), " % (Id, ActEnergy, ActEnergyValue, MeteringMeterId)
      sql = sql + "('%i','%i','%f','%i')," % (Id, ReactiveEnergy, ReactiveEnergyValue, MeteringMeterId)
      sql = sql + "('%i','%i','%f','%i')," % (Id, ActPower, ActPowerValue, MeteringMeterId)
      sql = sql + "('%i','%i','%f','%i');" % (Id, ReactPower, ReactPowerValue, MeteringMeterId)
      conn.execute(sql)
      #conn.commit()
      return 'Inserido Novos Dados'
```

```
    finally:
      conn.close()

## API entrance to insert new metering id
class InserirNovaMedicao(Resource):
  def get(self):
    try:
      conn = db_connect.connect()
      query = conn.execute("INSERT INTO Metering (MeterId, MeterFlag) VALUES (2,1)")
      return 'Inserido Nova Medição'
    finally:
      conn.close()

## API entrance to get meter flag (on or off)
class SelecionarFlags(Resource):
  def get(self, MeterId):
    Flags = []
    try:
      conn = db_connect.connect()
      #query = conn.execute("SELECT MeterFlag, UserFlag, Mode FROM Meter WHERE Id = (%i)" % MeterId)
      query = conn.execute("SELECT MeterFlag FROM Meter WHERE Id = (%i)" % MeterId)
      for i in query.cursor.fetchall():
        Flags = i[0]
        #Flags = Flags + [[i[0], i[1], i[2]]]
      return Flags
    finally:
      conn.close()

## API entrance to insert a metering flag
class InsereMeterFlagNovaMedicao(Resource):
  def get(self, MeterId, Flag):
    try:
      conn = db_connect.connect()
```

```
        query = conn.execute("INSERT INTO Metering (MeterId, MeterFlag) VALUES (%i, %i)" % (MeterId, Flag))
        return "Atualizado Flag no medidor %i para nova medicão" %(MeterId)
      finally:
        conn.close()


## API entrance to update a metering flag
class AtualizaMeterFlag(Resource):
   def get(self, Flag, MeterId):
      _Flag = Flag
      try:
        conn = db_connect.connect()
        if _Flag == 0:
          query = conn.execute("UPDATE Meter SET MeterFlag = 0 WHERE Id = %i" % (MeterId))
          MeterFlag = 0;
        else:
          query = conn.execute("UPDATE Meter SET MeterFlag = 1 WHERE Id = %i" % (MeterId))
          MeterFlag = 1;


        return "Status do medidor (MeterFlag) = %i " % (MeterFlag)
      finally:
        conn.close()



## Routing
# api.add_resource(MeterName, '/Meter/Name') # Route_1 - Return Meter name and address
api.add_resource(Meter, '/Meter')
api.add_resource(Metering, '/Metering')
api.add_resource(MeteringData, '/MeteringData/<int:LastId>')
api.add_resource(ChangeUserFlag, '/UserFlag1/<string:Address>')
api.add_resource(ChangeUserFlagTeste, '/UserFlag0/<string:Address>')
api.add_resource(ChangeMeterFlag1, '/MeterFlag1/<string:Address>')
api.add_resource(ChangeMeterFlag0, '/MeterFlag0/<string:Address>')
```

```
api.add_resource(NormalMode, '/Normal/<int:MeterId>')

api.add_resource(StandByMode, '/Standby/<int:MeterId>')

api.add_resource(Mode, '/Mode/<int:MeterId>')

api.add_resource(LastMeteringId, '/LastId/<int:MeterId>')

api.add_resource(LastestMetering,
'/LastestMetering/Medidor<int:MeterId>/<float:Iminimum>')

api.add_resource(LastMeteringIdRasp, '/LastMeteringIdRasp')

api.add_resource(InserirDados,
'/InserirDados/ID<int:Id>_METERINGMETERID<int:MeteringMeterId>__DATA<int:Volt
age>_VALUE<float:VoltageValue>__DATA<int:Current>_VALUE<float:CurrentValue>__D
ATA<int:PowerFactor>_VALUE<float:PowerFactorValue>__DATA<int:ActEnergy>_VALU
E<float:ActEnergyValue>__DATA<int:ReactiveEnergy>_VALUE<float:ReactiveEnergyVal
ue>__DATA<int:ActPower>_VALUE<float:ActPowerValue>__DATA<int:ReactPower>_VA
LUE<float:ReactPowerValue>')  # Route_1 ##Resposta do site OK

api.add_resource(InserirNovaMedicao, '/InserirNovaMedicao')

api.add_resource(SelecionarFlags, '/SelecionarFlags<int:MeterId>')

api.add_resource(InsereMeterFlagNovaMedicao,
'/InsereMeterFlagNovaMedicao<int:MeterId>Flag<int:Flag>')

api.add_resource(AtualizaMeterFlag,
'/AtualizaMeterFlag=<int:Flag>_Meter=<int:MeterId>')


        ## End Rest API ##


if __name__ == '__main__':
   app.run()
```

Apêndice B – Páginas web




Sistema de Monitoramen
delso.pythonanywhere.com/Medidor1/Historico1/inicio2018-06-05fim2018-06-07
Sistema de Monitoramento de Energia Elétrica
Medidor 1
Medidor 2
Medidor 3
Medidor 4
Medidor Geral
Tensão
Tensão
121.6
121.4
121.2
121.0
120.8
120.6
120.4
120.2
05-06-2018 10:19:06
05-06-2018 10:52:37
05-06-2018 11:04:06
05-06-2018 11:08:00
05-06-2018 11:11:32
05-06-2018 11:15:16
05-06-2018 11:19:03
05-06-2018 11:22:35
05-06-2018 11:26:19
05-06-2018 11:30:00
05-06-2018 11:33:27
05-06-2018 11:37:04
05-06-2018 11:40:43
05-06-2018 11:47:45

Apêndice C – Interface smartphone

Apêndice D – Programa desenvolvido no módulo mestre

```
############ ARQUIVO PRINCIPAL ####################
import os
import sqlite3
import time
import signal
from TesteMeterV19.meterv19 import Meter
from TesteMeterV19.InitV19 import FrequencyScan
from TesteMeterV19.MQTTV3 import MQTT
import threading
import requests
import RPi.GPIO as GPIO
import pigpio
import sys, traceback

def main():
    print('\nStarting SMI - Core...\n')
    os.system('systemctl stop serial-getty@ttyAMA0.service')
    time.sleep(3)
    print('\nLoading Meters...\n')
    Cloud = MQTT()
    Scan = FrequencyScan()
    Frequency = AdjustParam()
    Meters = []
    Meters = DownloadMeters(Frequency)
    Flag = 1
    iThreshold = 0.1
```

```
  try:
    print('\nStarting Meters...\n')
    MeterIdx = []
    MeterIdx = StartMeters(Meters)
    while True:
      metering = []
      metering_lastid = 0

      for MeterIdx in Meters:
        MQTTCommand(MeterIdx, iThreshold, Flag)

##########################################################################
          if (MeterIdx[4].meteringAvailable() == True):
            url = "http://delso.pythonanywhere.com/SelecionarFlags%i" % (MeterIdx[0])
            Flag = UrlResponseVerification(url)
            metering = MeterIdx[4].meteringGet()

            for metering_row in metering:
              url                                                                    =
"http://delso.pythonanywhere.com/InsereMeterFlagNovaMedicao%iFlag%i" % (
              MeterIdx[0], Flag)
              teste1 = UrlResponseVerification(url)

            url = "http://delso.pythonanywhere.com/LastMeteringIdRasp"
            teste1 = UrlResponseVerification(url)
            metering_lastid = teste1

            #    # Insert Data code 1 = voltage (V)
            #    # Data code 2 = current (A)
            #    # Data code 3 = power factor
            #    # Data code 4 = tot wh (wh)
            #    # Data code 5 = Pot (W)
            #    # Data code 6 = AparentPower (va)

            url                                                                    =
```

```
"http://delso.pythonanywhere.com/InserirDados/ID%i_METERINGMETERID%i__DATA
1_VALUE%.2f__DATA2_VALUE%.2f__DATA3_VALUE%.2f__DATA4_VALUE%.2f__DATA5_
VALUE%.2f__DATA6_VALUE%.2f" \
                % (metering_lastid, MeterIdx[0], metering_row[1], metering_row[2],
metering_row[3],
                    metering_row[4], metering_row[5],metering_row[6])
            teste1 = UrlResponseVerification(url)
.....
.....
.....
.....
.....
            VoltageMQTT = ("/Medidor/%i/Tensao") %(MeterIdx[0])
            CorrenteMQTT = ("/Medidor/%i/Corrente") % (MeterIdx[0])
            FPMQTT = ("/Medidor/%i/FP") % (MeterIdx[0])
            EnergiaMQTT = ("/Medidor/%i/Energia") % (MeterIdx[0])
            ReativaMQTT = ("/Medidor/%i/Reativa") % (MeterIdx[0])

            Cloud.client.publish(VoltageMQTT, ("Tensão = %fV" %(metering_row[1])))
            Cloud.client.publish(CorrenteMQTT, ("Corrente = %fA" % (metering_row[2])))
            Cloud.client.publish(FPMQTT, ("Fator de potencia = %f" %
(metering_row[3])))
            Cloud.client.publish(EnergiaMQTT, ("Energia = %fWh" % (metering_row[4])))
            Cloud.client.publish(ReativaMQTT, ("Energia Reativa = %fVArh" %
(metering_row[5])))
            metering = []
             except KeyboardInterrupt:
                print('Stopping SMI...\n')
                Scan.pi.set_mode(15, pigpio.ALT0)  # GPIO 15 as RX
                traceback.print_exc(file=sys.stdout)
                for MeterIdx in Meters:
                   GPIO.cleanup()
                   print('Stopping [%s]' % (MeterIdx[1]))
                   MeterIdx[4].TurnOffPWM()
                   if MeterIdx[4].isAlive(): MeterIdx[4].kill_received = True
```

```
            pid = os.getpid()
            os.kill(pid, signal.SIGKILL)
        except Exception:
            print('Stopping SMI...\n')
            Scan.pi.set_mode(15, pigpio.ALT0)  # GPIO 15 as RX
            traceback.print_exc(file=sys.stdout)
            for MeterIdx in Meters:
                GPIO.cleanup()
                print('Stopping [%s]' % (MeterIdx[1]))
                MeterIdx[4].TurnOffPWM()
                if MeterIdx[4].isAlive(): MeterIdx[4].kill_received = True
                pid = os.getpid()
                os.kill(pid, signal.SIGKILL)
    def UrlResponseVerification(url):
        response = requests.get(url)
        while (response.status_code != 200):
            if (response.status_code == 500):
                time.sleep(0.1)
                response = requests.get(url)
        return response.json()
    def AdjustParam():
        Scan = FrequencyScan()
        Gain = Scan.GainAdjust()
        time.sleep(10)
        Frequency = Scan.FrequencyAdjust()
        return Frequency
```

```
        def DownloadMeters(Frequency):
            Meters = []
            url = "http://delso.pythonanywhere.com/Meter"
            retorno = UrlResponseVerification(url)
            for Row in retorno:
                Meters = Meters + [[Row[0], Row[1], Row[2], Row[3], Meter(Row[2],
Frequency)]]
            print('Meters %s\n' % Meters)
            return Meters
        def StartMeters(Meters):
            for MeterIdx in Meters:
                print('[%s] Init' % (MeterIdx[1]))
                MeterIdx[4].start()
                time.sleep(1)
            return MeterIdx
        def MQTTCommand(MeterId,iThreshold,MeterFlag):
            if MeterId[0] == 1:
                if MeterFlag == 0:
                    if MQTT.RelayCommandId_1 == 1:
                        if  MeterId[4].meteringON(MeterId[0]):
                            print('Carga do Medidor %i ligada' % (MeterId[0]))
                            url                                                    =
"http://delso.pythonanywhere.com/AtualizaMeterFlag=1_Meter=%i" % (MeterId[0])
                            teste1 = UrlResponseVerification(url)

                elif MeterFlag == 1:
```

```
                if MQTT.SBCommandId_1 == 1:
                    if MQTT.RelayCommandId_1 == 0:
                    # if FlagLigado == 0:
                        url = "http://delso.pythonanywhere.com/LastestMetering/Medidor%i/%f" % (MeterId[0], iThreshold)
                        teste1 = UrlResponseVerification(url)
                        if teste1 == "D":
                            if MeterId[4].meteringOFF(MeterId[0]):
                                print('Carga do Medidor %i Desligado StandBy' % MeterId[0])
                                url = "http://delso.pythonanywhere.com/AtualizaMeterFlag=0_Meter=%i" % (MeterId[0])
                                teste1 = UrlResponseVerification(url)
                            else:
                                print("Erro ao desligar a carga %i" %(MeterId[0]))
                else:
                    if MQTT.RelayCommandId_1 == 0:
                    # if FlagLigado == 0:
                        if MeterId[4].meteringOFF(MeterId[0]):  # Função de desligamento
                            print('Carga do Medidor %i desligado' % (MeterId[0]))
                            url = "http://delso.pythonanywhere.com/AtualizaMeterFlag=0_Meter=%i" % (MeterId[0])
                            teste1 = UrlResponseVerification(url)
                        else:
                            print("Erro ao desligar a carga %i" % (MeterId[0]))


            elif MeterId[0] == 2:
```

```
                if MeterFlag == 0:
                    if MQTT.RelayCommandId_2 == 1:
                        if  MeterId[4].meteringON(MeterId[0]):
                            print('Carga do Medidor %i ligada' % (MeterId[0]))
                            url = "http://delso.pythonanywhere.com/AtualizaMeterFlag=1_Meter=%i" % (MeterId[0])
                            teste1 = UrlResponseVerification(url)
                elif MeterFlag == 1:
                    if MQTT.SBCommandId_2 == 1:
                        if MQTT.RelayCommandId_2 == 0:
                        # if FlagLigado == 0:
                            url = "http://delso.pythonanywhere.com/LastestMetering/Medidor%i/%f" % (MeterId[0], iThreshold)
                            teste1 = UrlResponseVerification(url)
                            if teste1 == "D":
                                if MeterId[4].meteringOFF(MeterId[0]):
                                    print('Carga do Medidor %i Desligado StandBy' % MeterId[0])
                                    url = "http://delso.pythonanywhere.com/AtualizaMeterFlag=0_Meter=%i" % (MeterId[0])
                                    teste1 = UrlResponseVerification(url)
                                else:
                                    print("Erro ao desligar a carga %i" %(MeterId[0]))
                    else:
                        if MQTT.RelayCommandId_2 == 0:
                        # if FlagLigado == 0:
                            if MeterId[4].meteringOFF(MeterId[0]):  # Função de desligamento
```

```
                    print('Carga do Medidor %i desligado' % (MeterId[0]))
                    url =
"http://delso.pythonanywhere.com/AtualizaMeterFlag=0_Meter=%i" % (MeterId[0])
                    teste1 = UrlResponseVerification(url)
                else:
                    print("Erro ao desligar a carga %i" % (MeterId[0]))
            elif MeterId[0] == 3:
                if MeterFlag == 0:
                    if MQTT.RelayCommandId_3 == 1:
                        if  MeterId[4].meteringON(MeterId[0]):
                            print('Carga do Medidor %i ligada' % (MeterId[0]))
                            url =
"http://delso.pythonanywhere.com/AtualizaMeterFlag=1_Meter=%i" % (MeterId[0])
                            teste1 = UrlResponseVerification(url)
                elif MeterFlag == 1:
                    if MQTT.SBCommandId_3 == 1:
                        if MQTT.RelayCommandId_3 == 0:
                        # if FlagLigado == 0:
                            url =
"http://delso.pythonanywhere.com/LastestMetering/Medidor%i/%f" % (MeterId[0],
iThreshold)
                            teste1 = UrlResponseVerification(url)
                            if teste1 == "D":
                                if MeterId[4].meteringOFF(MeterId[0]):
                                    print('Carga do Medidor %i Desligado StandBy' % MeterId[0])
                                    url =
"http://delso.pythonanywhere.com/AtualizaMeterFlag=0_Meter=%i" % (MeterId[0])
```

```
                    teste1 = UrlResponseVerification(url)
                else:
                    print("Erro ao desligar a carga %i" %(MeterId[0]))
            else:
                if MQTT.RelayCommandId_3 == 0:
                # if FlagLigado == 0:
                    if MeterId[4].meteringOFF(MeterId[0]):  # Função de desligamento
                        print('Carga do Medidor %i desligado' % (MeterId[0]))
                        url = "http://delso.pythonanywhere.com/AtualizaMeterFlag=0_Meter=%i" % (MeterId[0])
                        teste1 = UrlResponseVerification(url)
                    else:
                        print("Erro ao desligar a carga %i" % (MeterId[0]))
        if __name__ == "__main__":
            main()



        ########### CLASSE DOS MEDIDORES ###########
        import time
```

```
import serial
import os
from threading import Thread, Lock
import threading
import struct
from time import gmtime, strftime
from TesteMeterV19.InitV19 import FrequencyScan
import binascii
import pigpio
import RPi.GPIO as GPIO
class Meter(Thread):
    _address = None
    _addressControl = 0
    _SerialConfig = FrequencyScan()
    _metering = []
    _meteringReady = False
    _soframe = b'*'
    _eoframe = '0x21'
    _lock = threading.Lock()
    _Frequency = 0
    _AdjustParams = 0
    def __init__(self, address=None, Frequency = 0):

        Thread.__init__(self)
        self._address = address
        self._Frequency = Frequency
```

```
    self._stop_event = threading.Event()
def stop(self):
    self._stop_event.set()
def run(self):
    Scan = FrequencyScan()
    print ('## Starting Meter at Address ' + str(self._address))
    SerialControl = self._SerialConfig._commport
    SerialControl.flushInput()
    SerialControl.flushOutput()
    self.pi = pigpio.pi()
    self.pi.set_mode(18, pigpio.OUTPUT)
    Frequencies = FrequencyScan._Frequency
    OKcount = 0
    Errorcount = 0
    total = 0
    while True:
        print('Medidor '+ str(self._address) + ' aguardando...' )
        with self._lock:
            SerialControl.flushInput()
            SerialControl.flushOutput()
            GPIO.output(5, GPIO.HIGH)
            #demora +- 50ms
            rxdata = []
            rxeco = []
            txdata = []
            voltage = 0.0
```

```
current = 0.0
wh = 0.0
varh = 0.0
w = 0
var = 0
pf = 0.0
print('## Reading Meter ' + str(self._address))
send = ("#" + str(self._address) + "g" + ";")
send_byte = send.encode('ascii')
self.pi.hardware_PWM(18, Frequencies[self._Frequency] , 500000)  #
1.4MHz, 50%
txdata = SerialControl.write(send_byte)
rxeco = SerialControl.read(4)  # eco
rxdata = SerialControl.read(1)
if len(rxdata):
   if rxdata == self._soframe:
      rxdata = rxdata + SerialControl.read(29)
      teste = len(rxdata)
      if len(rxdata) == 30:
         hexarxdata =  hex(rxdata[29])
         # End of Frame check
         if hexarxdata == self._eoframe: #Verifica o header de fim
            #Lê os paramêtros enviados
            timestamp = strftime("%Y-%m-%d %H:%M:%S", gmtime())
            voltage, = struct.unpack('>f', rxdata[1:5])
            current, = struct.unpack('>f', rxdata[5:9])
```

```
                                pf, = struct.unpack('>f', rxdata[9:13])
                                wh, = struct.unpack('>f', rxdata[13:17])
                                varh, = struct.unpack('>f', rxdata[17:21])
                                w, = struct.unpack('>f', rxdata[21:25])
                                var, = struct.unpack('>f', rxdata[25:29])
                                print("## Voltage %.3f Current %.3f pf %.3f Wh %.3f varh %.3f w %.3f var %.3f\n " % (voltage, current, pf, wh, varh, w, var))
                                self._metering = self._metering + [[timestamp, voltage, current, pf, wh, varh, w, var]]
                                self._meteringReady = True
                                print('## Transmissão Completa')
                            else:
                                print('## Invalid End of Frame Meter ' + str(self._address) + '\n\n')
                        else:
                            print('## Frame Size Error Meter ' + str(self._address) + '\n\n')
                    else:
                        print('## Invalid Start of Frame Meter ' + str(self._address) + '\n\n')
                else:
                    print('## Meter ' + str(self._address) + ' RX < TIMEOUT\n')
                self.pi.hardware_PWM(18, 0, 0)
                GPIO.output(5, GPIO.LOW)
                time.sleep(5)
            def AdjustParamCount(self,C):
                if C == 1:
```

```
            return self._AdjustParams
        elif C == 0:
            self._AdjustParams = 0
    def TurnOffPWM(self):
        self.pi.hardware_PWM(18, 0, 0)
    def meteringAvailable(self):
        return self._meteringReady
    def meteringGet(self):
        meteringNow = []
        meteringNow = self._metering
        self._meteringReady = False
        self._metering = []
        return meteringNow
    def meteringOFF(self, addressOFF):
        # self._lock.acquire()
        SerialControl = self._SerialConfig._commport
        Frequencies = FrequencyScan._Frequency
        self._addressControl = addressOFF
        rxdata = []
        txdata = []
        self.pi.hardware_PWM(18, Frequencies[self._Frequency], 500000)     #
1.4MHz, 50%
        send1 = ("#" + str(self._addressControl) + "r" + ";")
        txdata = SerialControl.write(send1.encode('ascii'))
        rxeco = SerialControl.read(4)
        rxdata = SerialControl.read(1)
```

```
        self.pi.hardware_PWM(18, 0, 0)
        return 1
    def meteringON(self, addressON):
        # self._lock.acquire()
        SerialControl = self._SerialConfig._commport
        Frequencies = FrequencyScan._Frequency
        self._addressControl = addressON
        rxdata = []
        txdata = []
        self.pi.hardware_PWM(18, Frequencies[self._Frequency], 500000)  # 1.4MHz, 50%
        send = ("#" + str(self._addressControl) + "s" + ";")
        txdata = SerialControl.write(send.encode('ascii'))
        rxeco = SerialControl.read(4)
        rxdata = SerialControl.read(1)
        self.pi.hardware_PWM(18, 0, 0)
        return 1
```

############ CLASSE DE INICIALIZAÇÃO ###########

ALGORITMO DE SELEÇÃO DE FREQUENCIA

```
############E AJUSTE DE GANHO ################
import os
import time
import pigpio
import serial
from operator import xor
import RPi.GPIO as GPIO
import sys, traceback
os.system('pigpiod')
class FrequencyScan():
   _commport = '/dev/ttyAMA0'
   _testframe = b'T'
   _Frequency = [505000, 517000, 530000, 543000, 558000, 573000, 589000,
            606000, 624000, 643000, 663000, 684000, 707000, 731000,
            757500, 785500, 815769, 848400, 883750, 992173, 964090,
            1010000, 1060500]
   _FrequencyAmount = 23
   _start = 1
   _FreqIdx = 0
   _PortFlag = 1
   pi = pigpio.pi()
   def __init__(self):
      self._commport = serial.Serial(self._commport, baudrate=9600,
timeout=2, xonxoff=False, rtscts=False,
                        dsrdtr=False)
      GPIO.setmode(GPIO.BCM)
```

```
        GPIO.setup(6, GPIO.OUT)
        GPIO.setup(26, GPIO.OUT)
        GPIO.setup(19, GPIO.OUT)
        GPIO.setup(5, GPIO.OUT)
        GPIO.output(6, GPIO.LOW)
        GPIO.output(26, GPIO.LOW)
        GPIO.output(19, GPIO.LOW)
        GPIO.output(5, GPIO.LOW)
    def FrequencyAdjust(self):
        self.pi.set_mode(18, pigpio.OUTPUT)
        kcounter = 0
        k = [0]*23
        s = [0]*23
        Frequency = 0
        _FreqIdx = self._FreqIdx
        _start = self._start
        print('Inicio da varredura de frequência')
        self._commport.flushInput()
        self._commport.flushOutput()
        for contador in range(self._FrequencyAmount):
            send = ("T")
            send_byte = send.encode('ascii')
            self.pi.hardware_PWM(18, self._Frequency[contador], 500000)     #
1.4MHz, 50%
            txdata = self._commport.write(send_byte)
            time.sleep(0.1)
```

```
            rxeco = self._commport.read(1)  # eco
            if (rxeco == self._testframe):
                print('Frequência %i OK' % (self._Frequency[contador]))
                k[contador] = 1
                kcounter = kcounter + 1
                s[contador] = contador
            else:
                print('Erro na Frequência %i' % (self._Frequency[contador]))
                k[contador] = 0
                s[contador] = 0
            _FreqIdx += s[contador]
        FrequencyIdx = int(_FreqIdx/kcounter)
        print('Média = %i' % (FrequencyIdx))
        print('Frequência selecionada = %iHz' % self._Frequency[FrequencyIdx])
        return FrequencyIdx
    def GainAdjust(self):
        c = 0
        GPIO.setup(15, GPIO.IN)
        GPIO.add_event_detect(15, GPIO.FALLING, callback=self.RXinterrupt)
        State = 1
        while (c != 1):
            if (State == 1):
                GPIO.output(6, GPIO.LOW)
                GPIO.output(26, GPIO.LOW)
                time.sleep(1)
                if (self._PortFlag == 1):
```

```
            State = 2
        else:
            c = 1
    elif (State == 2):
        GPIO.output(6, GPIO.HIGH)
        GPIO.output(26, GPIO.LOW)
        time.sleep(1)
        if (self._PortFlag == 1):
            State = 3
        else:
            GPIO.output(6, GPIO.LOW)
            GPIO.output(26, GPIO.LOW)
            State = 1
            c = 1
    elif (State == 3):
        GPIO.output(6, GPIO.LOW)
        GPIO.output(26, GPIO.HIGH)
        time.sleep(1)
        if (self._PortFlag == 1):
            State = 4
        else:
            GPIO.output(6, GPIO.HIGH)
            GPIO.output(26, GPIO.LOW)
            State = 2
            c = 1
    elif (State == 4):
```

```
            GPIO.output(6, GPIO.HIGH)
            time.sleep(1)
            if (self._PortFlag == 1):
               c = 1
            else:
               GPIO.output(6, GPIO.LOW)
               GPIO.output(26, GPIO.HIGH)
               State = 3
               c = 1
      print('Ganho ajustado = %i' % (State))
      GPIO.remove_event_detect(15)
      self.pi.set_mode(15, pigpio.ALT0)
      return (State)
   def RXinterrupt(self, channel):
      _PortFlag = self._PortFlag
      print('Interrupção em RX')
      _PortFlag = 0
      self._PortFlag = _PortFlag
```

```
####### CLASSE DE CONTROLE DO MEDIDOR ############
import paho.mqtt.client as mqtt, os, urllib.parse
```

```
import time
class MQTT():
    RelayCommandId_1 = 1
    RelayCommandId_2 = 1
    RelayCommandId_3 = 1
    RelayCommandId_4 = 1
    RelayCommandId_5 = 1
    SBCommandId_1 = -1
    SBCommandId_2 = -1
    SBCommandId_3 = -1
    SBCommandId_4 = -1
    SBCommandId_5 = -1
    def TrataComando(topic, command):
        _topic = topic
        _command = command
        if _topic == "/Medidor/1/Comando":
            if _command == "1":
                MQTT.RelayCommandId_1 = 1
            else:
                MQTT.RelayCommandId_1 = 0
        elif _topic == "/Medidor/1/Standby":
            if _command == "1":
                MQTT.SBCommandId_1 = 1
            else:
                MQTT.SBCommandId_1 = 0
        elif _topic == "/Medidor/2/Comando":
```

```
            if _command == "1":
                MQTT.RelayCommandId_2 = 1
            else:
                MQTT.RelayCommandId_2 = 0
        elif _topic == "/Medidor/2/Standby":
            if _command == "1":
                MQTT.SBCommandId_2 = 1
            else:
                MQTT.SBCommandId_2 = 0
        elif _topic == "/Medidor/3/Comando":
            if _command == "1":
                MQTT.RelayCommandId_3 = 1
            else:
                MQTT.RelayCommandId_3 = 0
        elif _topic == "/Medidor/3/Standby":
            if _command == "1":
                MQTT.SBCommandId_3 = 1
            else:
                MQTT.SBCommandId_3 = 0
    def on_connect(mosq, obj, rc):
        print("")
    def on_message(mosq, obj, msg):
        print ("on_message:: this means  I got a message from brokerfor this
topic")
        Teste = msg.topic
        Teste2 = str(int(msg.payload))
```

```
    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))
    MQTT.TrataComando(Teste, Teste2)
    print("")
def on_publish(mosq, obj, mid):
        pass
def on_subscribe(mosq, obj, mid, granted_qos):
        pass
def on_log(mosq, obj, level, string):
    print(string)
client = mqtt.Client()
client.on_message = on_message
client.on_connect = on_connect
client.on_publish = on_publish
client.on_subscribe = on_subscribe
client.on_log = on_log
client.username_pw_set("tdqtilqg", "DA-c7ZBF0YQA")
client.connect('m10.cloudmqtt.com', 15435, 60)
client.subscribe([("/Medidor/1/Comando", 0), ("/Medidor/1/Standby", 0),
        ("/Medidor/2/Comando", 0), ("/Medidor/2/Standby", 0),
        ("/Medidor/3/Comando", 0), ("/Medidor/3/Standby", 0)])
client.loop_start()
```