UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO

MEDIDOR ELETRÔNICO DE CONSUMO DE ENERGIA ELÉTRICA

Autor: _____

Paulo Gentil Gibson Fernandes

Orientador: _____

Carlos José Ribas D'Ávila

Examinador: _____

Fernando Antônio Pinto Barúqui

Examinador: _____

Paulo Magalhães Duarte Sobrinho

DEL
Março/2006

Aos meus pais,
pelo carinho e
dedicação.

Agradecimentos

Percebi que o trabalho de pesquisa é um grande exercício intelectual. Durante o desenvolvimento de um projeto existem ocasiões em que tudo parece muito complexo e, muitas vezes, é difícil encontrar uma saída para contornar os problemas. Com isso, ficamos desanimados e desmotivados em continuar persistindo na eterna busca por soluções engenhosas. Entretanto, quando existe uma pessoa que possui a capacidade de visualizar o problema e ditar diretrizes para que nos possibilite encontrar a lógica por traz do quebra-cabeça, ficamos imensamente satisfeitos e realizados. Assim, gostaria de agradecer imensamente ao Casé, meu orientador, por me proporcionar os subsídios necessários a encontrar soluções criativas para os problemas encontrados durantes a realização de nossos trabalhos. Pude notar que não podemos nos deixar abalar em momentos de dificuldades, pois só assim temos a capacidade de crescer.

Gostaria de agradecer ainda ao meu colega de trabalho Luciano, por estar sempre receptivo e disposto, ajudando-me com minhas dúvidas de funcionamento dos variados circuitos que necessitamos destrinchar em projetos de engenharia.

Agradeço também ao meu amigo Leonardo, que me ensinou a importância do trabalho em equipe quando foi necessária nossa união para vencer os desafios do PIC.

A toda equipe do laboratório LASPI por serem ótimos colegas tornando nosso ambiente de trabalho sempre agradável e prazeroso.

Agradeço a imensa dedicação e carinho de meu pai, por se empenhar proporcionar as melhores condições para minha formação humana. Ao meu irmão gêmeo, e melhor amigo, por estar sempre presente nos momentos difíceis e alegres da minha vida. A minha mãe, sempre me apoiando e me orientando nas minhas decisões. Aos meus irmãos e toda a família por proporcionarem um núcleo de união, harmonia e amizade.

Finalmente, agradeço a Natureza por me dar à oportunidade de estar aqui vivendo esta experiência humana.

Resumo

Nos dias de hoje, inúmeros meios para transferência de dados são utilizados: cabos coaxiais, fibra ótica, link de microondas e radiofreqüência (RF) são alguns exemplos. Entretanto, as concessionárias distribuidoras de energia elétrica ainda utilizam funcionários para o procedimento de leitura dos seus medidores eletromecânicos de consumo responsáveis por tarifar a energia entregue aos clientes.

Este trabalho apresenta uma solução para implementar um sistema de medição remota automática de consumo, reunindo os conceitos de microcontroladores, eletrônica digital, eletrônica de potência, RF e filtros, aprendidos no curso de engenharia eletrônica da UFRJ. Este sistema, chamado de AMR (Automatic Measurement Reading), utiliza medidores eletrônicos que se comunicam com uma estação concentradora (PC) através da própria rede elétrica (PLC). O computador utiliza uma placa para converter os sinais que trafegam pela linha PLC em um padrão que lhe seja compreendido. Dessa forma, comandos podem ser enviados por um software concentrador aos diversos medidores conectados ao mesmo barramento elétrico.

Assim, este desenvolvimento implementa os subsídios de hardware e firmware necessários para se criar uma rede integrada de gestão e controle do fornecimento de energia elétrica.

Palavras-chave

- AMR
- Medição de consumo
- Microcontrolador
- Firmware
- Comunicação
- Protocolo

Sumário

Agradecimentos ***iii***

Resumo ***iv***

Palavras-chave ***v***

Sumário ***vi***

Lista de Figuras ***viii***

Lista de Tabelas ***ix***

Lista de Tabelas ***ix***

Lista de Equações ***x***

Lista de abreviaturas ***xi***

Introdução ***1***

Capítulo 1: A Medição Remota de Consumo ***3***

1.1 Introdução 3

1.2 A topologia do sistema de medição remota de consumo (AMR) 4

1.3 As diferentes tecnologias para a transmissão de dados 5

Capítulo 2: A proposta do Medidor Eletrônico de Consumo ***7***

2.1 Diagrama de Blocos do Sistema 8

2.2 Programa Teste *Concentrador* 9

2.3 Medidor de energia elétrica 10

2.4 Conversor RS-232 PLC 11

Capítulo 3: O projeto do sistema ***13***

3.1 O circuito medidor 13

3.1.1 Projeto do circuito 18

3.2 Microcontrolador 21

3.2.1 Funcionamento do microcontrolador 22

3.2.2 O circuito de corte, religação e sensoriamento 23

3.2.3 O totalizador com memória não volátil 25

3.2.4 O protocolo de comunicação 25

3.2.5 A comunicação serial 27

3.3 Programas 27

3.3.1 Simulador e compilador 27

3.3.2 Firmware do Medidor 28

3.3.3 Firmware do Conversor RS-232 PLC 30

3.3.4 Software do Concentrador 33

3.4 Comunicação PLC 34

3.4.1 Transmissor FSK 34

3.4.2 Amplificador de Potência 36

3.4.3 O acoplador para rede elétrica 37

3.4.4 O receptor FM 39

Capítulo 4: Testes e resultados ***43***

4.1 Erro na Medição do Consumo 43

4.2 Desempenho da Comunicação PLC 44

Capítulo 5: Conclusões e trabalhos futuros ***46***

Apêndice 1 – Cálculos de projeto para o Medidor ***49***

Apêndice 2 – Descrição dos procedimentos de teste ***52***
Calibração do medidor 52
Método de calibração 52
Projeto 54
Procedimento de teste 55
Teste da taxa de erro de bit (BER) na rede PLC 56
Fimware transmissor 56
Firmware receptor 56
Resultados da BER 58
Placa Medidor de Energia Elétrica 59
Placa Conversor RS-232 PLC 60

Anexo 2 – Esquemático das placas desenvolvidas ***61***
Placa Medidor de Energia Elétrica – esquema 1: Medidor de ADE7757 61
Placa Medidor de Energia Elétrica – esquema 2: Microcontrolador PIC 62
Placa Medidor de Energia Elétrica – esquema 3: Estágio de Potência e fonte 63
Placa Medidor de Energia Elétrica – esquema 4: Demodulador 64
Placa Conversor RS-232 PLC – esquema 1: Micorcontrolador e interface RS-232 65
Placa Conversor RS-232 PLC – esquema 2: Transmissor e fonte 66
Placa Conversor RS-232 PLC – esquema 3: Demodulador 67

Anexo 3 – Códigos-fonte dos programas desenvolvidos ***68***
Programa Medidor 68
Programa Modem 83
Programa Teste Concentrador 88

Lista de Figuras

Figura 1 – Conexão entre medidores, concentrador e o servidor da distribuidora ___ 4
Figura 2 – Conexão Conversor RS-232 PLC, Medidor, Concentrador e rede elétrica. 8
Figura 3 – Conexão do resistor shunt junto ao Medidor. _____ 8
Figura 4 – Aplicativo para testar as funcionalidades do medidor. _____ 9
Figura 5 – Diagrama de blocos do medidor de energia e suas funcionalidades. ____ 11
Figura 6 – Conexão Conversor RS-232 PLC ao Concentrador _____ 12
Figura 7 – Esquema elétrico do medidor monofásico. _____ 14
*Figura 8 – Diagrama de blocos do circuito integrado ADE7757.*_____ 15
Figura 9 – Rede de atenuação para realizar a amostragem da tensão de linha. ____ 15
Figura 10 – Esquema de conexão entre a rede resistiva e a linha para o canal V2. _ 16
Figura 11 – Esquema de conexão entre a linha e o resistor shunt para o canal V1. _ 16
Figura 12 – Isolamento entre as alimentações do medidor e do controlador. _____ 17
Figura 13 – Fonte de alimentação para o circuito ADE7757. _____ 18
Figura 14 – Variação do oscilador em função de RCLKIN. _____ 19
*Figura 15 – Diagrama de pinos do microcontrolador PIC16F88*_____ 21
Figura 16 – Circuito para esquemático de funcionamento do microcontrolador ____ 22
*Figura 17 – Conexão série do relé de corte e religação.*_____ 23
Figura 18 – Foto do relé de estado sólido _____ 23
Figura 19 – Circuito para sensoriamento da fase _____ 24
Figura 20 – PIC simulator IDE _____ 28
Figura 21 – Fluxograma da rotina principal do firmware Medidor _____ 29
Figura 22 – Fluxograma da Função RX _____ 30
Figura 23 – Conexão entre os sinais do PC e da rede o microcontrolador _____ 31
Figura 24 – Fluxograma do programa Conversor RS-232 PLC _____ 32
Figura 25 – Fluxograma do programa Teste Concentrador. _____ 33
*Figura 26 – Conexão da saída serial do PIC para modulação por firmware*_____ 35
Figura 27 – Fluxograma da rotina para modular o sinal serial _____ 36
Figura 28 – Amplificador potência push-pull para o circuito transmissor _____ 37
Figura 29 – Esquema do acoplador de sinal para rede elétrica. _____ 37
Figura 30 – Circuito para cortar a alimentação do estágio de potência. _____ 38
Figura 31– Diagrama de blocos do receptor PLC. _____ 39
Figura 32 – Filtro passivo sintonizado _____ 40
Figura 33 – Circuito amplificador sintonizado _____ 41
*Figura 34 – Circuito integrado de demodulação*_____ 41
Figura 35 – Diagrama de blocos do demodulador MC13135. _____ 42
*Figura 36 – Filtro ativo passa baixa.*_____ 42
*Figura 37 – Erro na medição do consumo de energia em função da corrente.*_____ 44
Figura 38 –Taxa de erro em função da potência recebida _____ 45
Figura 39 – Projeto da rede resistiva. _____ 50
Figura 40 – Circuito final da rede resistiva. _____ 50
*Figura 41 – Funcionamento do medidor ADE7757*_____ 52
*Figura 42- Circuito para simulação da tensão da rede*_____ 53
Figura 43 – circuito para simulação de corrente _____ 53
Figura 44 – Fluxograma do firmware para transmissão de bytes 0x55 _____ 56
Figura 45 – Fluxograma do firmware para calcular a BER _____ 57

Lista de Tabelas

Tabela 1 – Seleção de freqüências F1-4, considerando oscilador a 450kHz. _____ *19*
Tabela 2 – Ajuste da freqüência de saída para o microcontrolador. _____ *20*
Tabela 3 – Especificações do projeto. _____ *20*
Tabela 4 – Característica do PIC16F88 necessárias a este projeto _____ *22*
Tabela 5 – Caracteres utilizados no protocolo de comunicação _____ *25*
Tabela 6 – Seqüência de caracteres para operação de leitura _____ *26*
Tabela 7- Seqüência de caracteres para operação de outros comandos do Medidor _ *26*
Tabela 8 – características da comunicação serial _____ *27*
Tabela 9 – Comparação entre os transformadores 3:1 e 1:1. _____ *38*
Tabela 10 – Resultados das medidas do consumo de energia _____ *43*
Tabela 11 – Resultados da comunicação PLC em 1 milhão de bits enviados _____ *45*
Tabela 12 – Valores dos resistores utilizados para as correntes teóricas. _____ *55*
Tabela 13 – Comparação entre valores práticos e teóricos _____ *55*
Tabela 14 – Erro de bits em 1 milhão para diferentes potências. _____ *58*

Lista de Equações

Equação 1 – Relação entre as freqüências F1 e F2 e as tensões V1 e V2. _____ *18*
Equação 2 – Freqüência de ressonância do filtro LC _____ *40*
Equação 3 – Formula da multiplicação de dois senos. _____ *41*
Equação 4 – Equação final do medidor de energia _____ *51*

Lista de abreviaturas

AMR	– Sistema de medição automática de consumo
PLC	– Comunicação através da rede de distribuição de energia.
USART	– Receptor transmissor serial síncrono e assíncrono.
WAN	– *Wide Area Network,* rede privada de larga cobertura para transmissão de dados
FM	– Modulação em freqüência.
FSK	- *Frequency Shift Keying*, modulação digital em freqüência
RF	– Rádio Freqüência
RTS	– *Request to Send,* sinal que indica o início da transmissão
RX	– Entrada de recepção do sinal serial
TX	- Saída de transmissão do sinal serial
PIC	- Microcontrolador da *Microchip*, PIC16F88
AD	- Conversor analógico-digital
SNR	- Razão sinal ruído
V2 ou V2rms	- Tensão de entrada do medidor proporcional ao valor da rede
V1 ou V1rms	- Tensão proporcional ao valor da corrente de carga
VREF	- Tensão de referência de 2,5V para o ADE7757
ADE7757	- Circuito integrado de medição de energia
MC13135	- Circuito integrado para demodulação FM
F1-4	- Freqüência base do ADE7757 para medição de energia
RCLKIN	- Pino do ADE7757 para conectar o oscilador local
F1 e F2	- Freqüência de saída do ADE7757 proporcional ao consumo
FCF	- Freqüência de saída do pino CF do ADE7757 para enviar o consumo ao microcontrolador
EEPROM	- Memória não volátil interna ao microcontrolador PIC
TMR0	- Registrador interno do PIC que armazena o valor do *Timer 0*.
Timer 0	- Contador interno do PIC que realiza o totalizador eletrônico.
BER	- Taxa de erro de bit
RS-232	- Porta serial do computador
Assembly	- Linguagem de baixo nível utilizada para programar o PIC
TX_MOD	- Saída do sinal de comunicação modulado do PIC

TX_POT - Saída de potência do sinal modulado

RTS_PIC - Sinal do PIC para indicar o envio da informação para linha PLC

RTS_PC - Sinal do PC para indicar que está enviando dados para o PIC

RX_DEM -Saída do receptor PLC com o sinal digital serial

RX_MOD -Sinal modulado recebido pelo receptor

Introdução

A idéia de se desenvolver um medidor eletrônico de energia elétrica foi a evolução do Projeto P&D-2002, da ANEEL, proposto pela UFRJ em conjunto com a LIGHT, para medição remota de consumo. No projeto foi desenvolvida uma placa que se comunicava através da rede de distribuição, permitindo ser acoplada ao medidor eletromecânico e monitorar seu funcionamento. Dessa forma, o circuito seria capaz de enviar informações internas como o valor do totalizador e a presença das fases. O sistema ainda permitia o corte e a religação do fornecimento de forma remota.

Este equipamento está sendo bastante procurado nos dia de hoje por empresas distribuidoras de energia para substituir o sistema convencional de medição. No sistema atual a empresa necessita de pessoas para ir até o estabelecimento consumidor e registrar o consumo. Com a tecnologia disponível hoje é possível realizar um sistema completamente automático para medição do consumo.

Entretanto, o que ainda atrasa a migração para a medição remota é o custo de um medidor eletrônico quando comparado ao convencional eletromecânico. Medidores eletrônicos são caros porque utilizam componentes com grande precisão e possuem, ainda, baixa escala da produção.

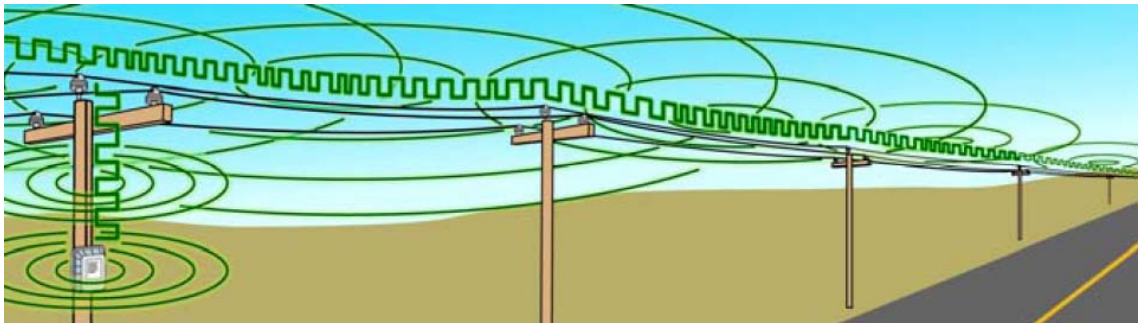
Com isso, no projeto em questão, tem-se como objetivo desenvolver um circuito completo para medir o consumo de energia e se comunicar com um concentrador local, agregando as mesmas funcionalidades do projeto anterior, mas sem necessitar de uma aparelhagem eletromecânica.

Este projeto utiliza um modulador FSK e um demodulador FM para uma comunicação banda estreita em 113kHz e um circuito integrado de baixo custo que realiza a tarifação do consumo de energia para clientes monofásicos com consumo até 70 amperes. Uma inovação existente neste projeto foi o desenvolvimento de uma rotina em *Assembly* para modular um sinal em FSK descartando a necessidade de utilização de um modulador. O receptor se baseia em um integrado que realiza uma demodulação FM por dupla conversão, um duplo receptor super-heteródino.

No primeiro capítulo deste trabalho será apresentado o sistema AMR. Em seguida, será discutido o projeto para medição remota automática de consumo com o desenvolvimento de um medidor eletrônico, uma placa conversora RS-232 / PLC e um software concentrador. No capítulo três um estudo profundo do funcionamento dos

circuitos e programas desenvolvidos é realizado. No capítulo quatro são abordados os procedimentos de testes, para checar e validar o funcionamento do projeto. Os resultados obtidos são apresentados. No capítulo cinco é realizada a conclusão do trabalho e são apresentadas propostas para trabalhos futuros. Ao final do trabalho existem fotos das placas, esquemas e códigos-fonte utilizados no projeto.

Capítulo 1: A Medição Remota de Consumo



(Ilustração)

1.1 Introdução

Hoje em dia, no Brasil, a maior parte das medições de consumo de energia elétrica envolve a leitura de um mostrador presente nos medidores eletromecânicos, o totalizador, exigindo o deslocamento de funcionários da concessionária para coletar esta informação.

Este método de leitura ainda não é automático e por este motivo envolve três etapas principais. Uma etapa para o deslocamento do funcionário ao estabelecimento consumidor para registrar o consumo. A outra etapa compreende a chegada ao local, leitura do totalizador eletromecânico e registro no papel. Em uma última etapa, transfere-se a informação do papel ao banco de dados, o que normalmente é realizado por um outro funcionário.

Esse método de realizar o serviço de leitura do consumo envolve um custo operacional, com pagamento de funcionários, transporte e material de trabalho. Além disso, como todo o trabalho é manual, ele torna-se lento, com possibilidade de erros no registro das informações.

Dentro deste contexto, a medição remota de consumo, conhecida internacionalmente como AMR (Automatic Measurement Reading), é, sem dúvida alguma, uma tecnologia promissora para as concessionárias de energia elétrica. Implantar um sistema automático para a leitura do consumo permite o conhecimento da curva de demanda, a identificação precisa da freqüência e duração das falhas no fornecimento (FEC e DEC) e o corte e religação remotos de clientes inadimplentes. Assim, a empresa pode melhorar o seu desempenho e aumentar o lucro final.

1.2 A topologia do sistema de medição remota de consumo (AMR)

Em uma arquitetura de medição remota de consumo os medidores de energia se comunicam eletronicamente a um concentrador local. Este concentrador, que pode ser implementado por um computador portátil, é responsável por armazenar as informações de consumo de todos os medidores conectados ao mesmo barramento.

Um sistema AMR pode estar baseado nos medidores eletromecânicos atualmente utilizados para tarifar o consumo de energia, sem que haja a necessidade da substituição. Nessa situação, é desenvolvido um circuito eletrônico para monitorar o medidor eletromecânico e armazenar a informação de consumo. Ao comando de requisição, este circuito embarcado no medidor eletromecânico envia os dados para o concentrador. Essa solução pode ser interessante na medida em que utiliza os medidores já adquiridos pela empresa, evitando um alto investimento na troca por novos medidores.

Uma outra opção é a substituição dos medidores eletromecânicos por eletrônicos. Dessa forma, o sistema de comunicação e monitoramento eletrônico do consumo de energia já estariam embutidos dentro do circuito do medidor.

Na topologia AMR, cada medidor deve possuir um endereço distinto para que possa ser identificado na linha. Por exemplo, em um condomínio, todos os medidores deverão estar conectados através de uma rede comum a um mesmo computador que deverá realizar pedidos regulares de leitura dos totalizadores e armazenar estes dados. A figura 1 mostra a conexão de um concentrador ao demais medidores.

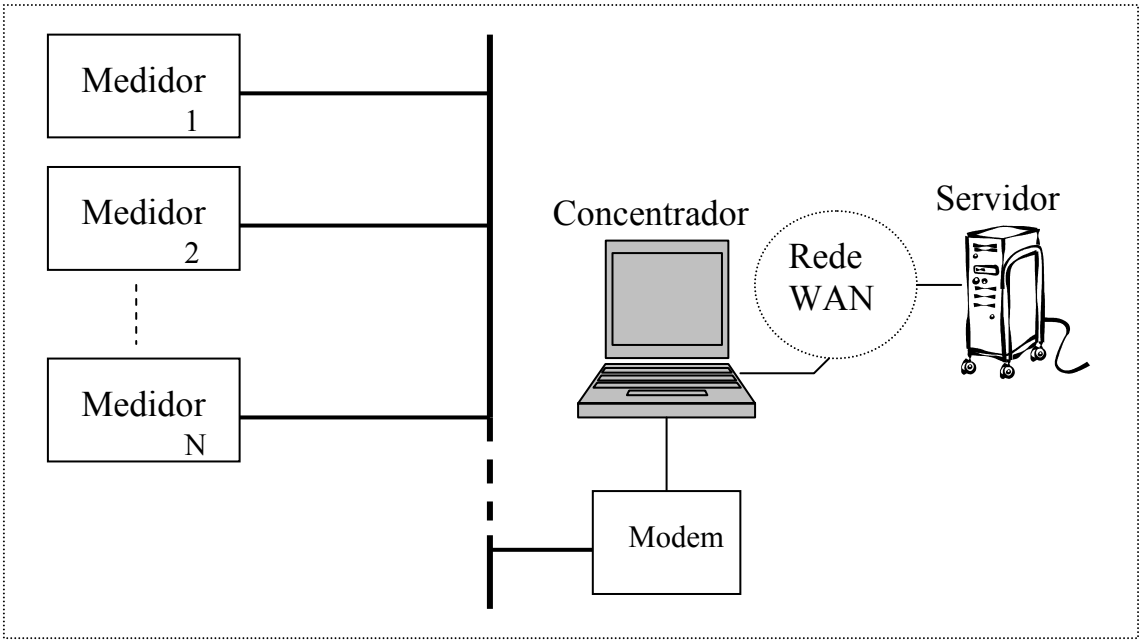


Figura 1 – Conexão entre medidores, concentrador e o servidor da distribuidora

Em seguida, as informações de consumo concentradas em um único ponto deverão ser enviadas a um servidor da companhia de distribuição através de uma rede interna privada WAN, ou mesmo uma rede pública como a Internet, usando, por exemplo, o protocolo TCP/IP. As informações dos clientes são armazenadas em um banco de dados e são utilizadas pelo sistema de cobrança da concessionária. Dessa forma, a concessionária atinge um ponto onde todo o processo de gerenciamento do consumo e cobrança estarão automatizados.

1.3 As diferentes tecnologias para a transmissão de dados

Os cabos coaxiais, fibras óticas, links de microondas, links de satélite e linha telefônica são alguns exemplos dos meios de transmissão de dados utilizados nos dias de hoje. Com isso, existem várias formas de se conceber a arquitetura de um sistema de medição remota de consumo para garantir uma comunicação entre o medidor e o servidor.

Para implantar o sistema, deve-se fazer uma distinção entre consumidores rurais e urbanos. As zonas rurais possuem uma baixa densidade populacional e as zonas urbanas possuem uma concentração maior de pessoas em uma mesma região. Dessa forma, pode ser mais interessante utilizar tecnologias diferentes para estes dois casos.

Um dos métodos mais baratos para se transmitir a informação de consumo em áreas rurais é a utilização de uma linha telefônica. Neste método, o medidor é substituído por um novo com modem de linha telefônica. O equipamento é programado para discar em um dado horário do dia e enviar as informações de consumo para um concentrador. A vantagem deste método é a utilização da infra-estrutura de comunicação telefônica existente não havendo a necessidade de se manter uma rede dedicada a este fim. A desvantagem é o pagamento do uso da linha.

Uma outra opção para clientes rurais é utilizar um sistema PLC, que será discutido ao longo do trabalho. Nesta solução a informação trafega pela própria rede de distribuição elétrica. Para o caso de clientes distantes, o sinal trafega em média tensão e, para clientes próximos, em baixa tensão, até alcançar um concentrador. Deve-se fazer uma distinção entre o PLC de baixa tensão e de média tensão, pois o último requer estruturas mais complexas, que devem suportar uma tensão de no mínimo 20kV, que por este motivo são mais caras.

Para o caso de zonas urbanas pode-se utilizar sistemas de comunicação sem fio ou um sistema de PLC para baixa tensão. Neste caso a distância dos medidores ao concentrador é menor, caso típico de um condomínio formado por edifícios ou um conjunto de casas em uma mesma rua. A informação trafega pela rede de baixa tensão (PLC) ou pelo ar (sem fio) até um computador utilizando um modem próprio para captar estes dados. Em seguida, ao concentrar todas as informações dos consumidores, o PC envia a informação através da rede pública (Internet) ou privada (WAN) para o servidor.

A principal vantagem de se utilizar a rede elétrica como meio de transmissão é que não há a necessidade de um investimento em uma infra-estrutura que estabeleça um meio físico para transportar as informações.

Entretanto, ao se decidir pela utilização da rede elétrica como meio de comunicação, deve-se analisar previamente a situação das instalações, pois, em alguns casos, circuitos em condições precárias ou topologias desfavoráveis podem atenuar o sinal, degradando o desempenho. A ocupação do espectro de rádio-freqüência, com sérias restrições de banda, também é um fator favorável na escolha da rede de distribuição elétrica para a transmissão de dados com alta taxa de comunicação.

Capítulo 2: A proposta do Medidor Eletrônico de Consumo

A proposta deste projeto é desenvolver um equipamento eletrônico de medição de consumo de energia com um totalizador eletrônico de 5 bytes, funcionalidades de corte/religação de energia, sensoriamento de fase e comunicação de dados pela rede elétrica. Ao se utilizar a rede elétrica com este fim, evita-se os gastos com a infra-estrutura de uma rede entre os medidores e um concentrador local. Para implantar a medição inteligente devemos eleger um local onde será instalado um computador, responsável por concentrar os dados de consumo dos medidores, devendo estar conectado a rede, e consequentemente, aos medidores.

Além de se desenvolver medidores com a funcionalidade de comunicação PLC foi necessário projetar um circuito para ser utilizado junto ao *Concentrador*. Esta placa, chamada de *Conversor RS-232 PLC*, realiza a interface entre um PC convencional e a rede elétrica. Ela recebe os pedidos provenientes do PC, através da porta RS-232, converte para um sinal PLC e faz o acoplamento com a linha de baixa tensão. O medidor correspondente recebe a informação pela rede elétrica e responde ao *Concentrador*. O *Conversor* então é capaz de receber a resposta, converter o sinal PLC e devolver ao computador pela linha serial RS-232.

Assim, o projeto compreende o desenvolvimento de dois módulos: o primeiro, chamado *Medidor*, tem o objetivo de ser instalado junto ao cliente para totalizar o consumo de energia, e o segundo, chamado *Conversor RS-232 PLC*, responsável por realizar a interface entre a saída serial do computador e a rede elétrica. Esses módulos são circuitos que envolvem comunicação serial, circuito para medição de energia, moduladores e demoduladores, entre outros, e são gerenciados por circuitos integrados denominados de microcontroladores.

O projeto também prevê o desenvolvimento dos firmwares a serem executados pelo microcontrolador PIC16F88 para o desempenho das funcionalidades do *Medidor* e do *Conversor*. Além disso, para testar o funcionamento das duas placas, foi desenvolvido um programa, em linguagem *C*, para ser executado em Windows. Este software é responsável por simular o funcionamento do *Concentrador* e permite testar a funcionalidades do medidor desenvolvido.

2.1 Diagrama de Blocos do Sistema

Para que fosse possível validar o funcionamento do *Medidor*, do *Conversor* e do software de teste do *Concentrador* foi necessário criar um ambiente para testar a comunicação. Esta estrutura foi montada no laboratório LASPI /UFRJ, utilizando a instalação elétrica local para conectar os equipamentos. Em uma tomada foi conectado o *Medidor* e, em um outro ponto mais distante, foi conectada a placa *Conversor* junto ao *Concentrador*, como mostra a figura 2.

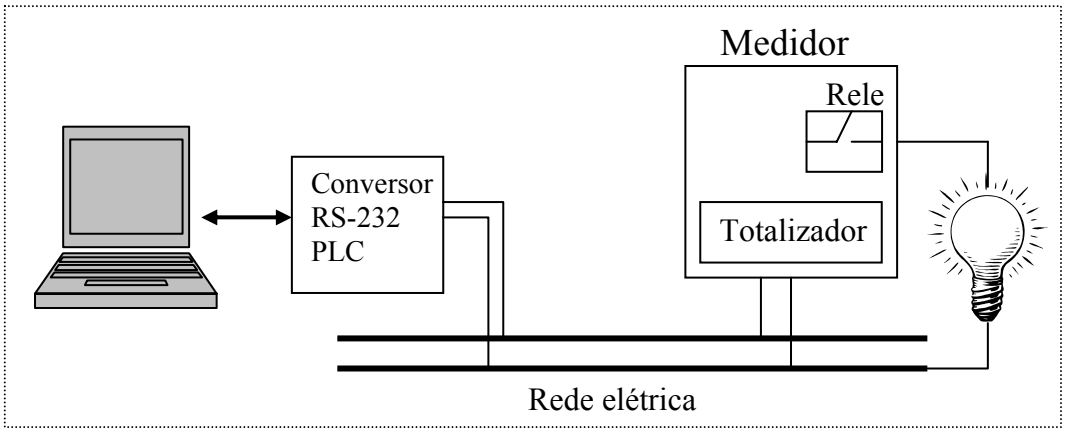


Figura 2 – Conexão Conversor RS-232 PLC, Medidor, Concentrador e rede elétrica.

No teste de validação do funcionamento do sistema é requisitado, repetidas vezes, ao circuito medidor que envie as informações do totalizador, checando no monitor a coerência do valor de consumo. Para gerar um consumo de energia e fazer com que o contador seja incrementado será necessário conectar uma carga ao medidor. A medição utiliza um resistor *shunt, Rs*, como mostrado na figura 3.

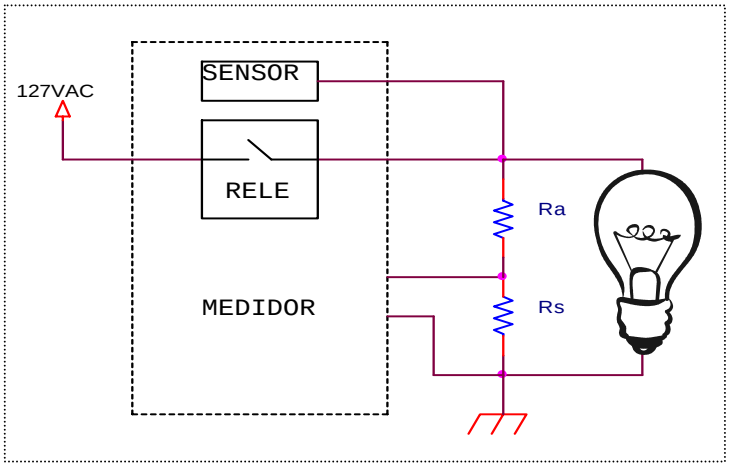


Figura 3 – Conexão do resistor *shunt* junto ao *Medidor*.

Para simular uma carga de valor elevado é utilizado o divisor resistivo formado por *Ra* e *Rs* (figura 3). Neste circuito, o consumo do cliente é escalado e está representado somente por *Ra*+*Rs*. Utilizando um resistor *Rs* proporcionalmente maior é possível verificar o funcionamento do circuito de medição para correntes mais elevadas. Para efeito dos testes, o consumo da lâmpada não será considerado, permanecendo apenas como uma indicação luminosa do fornecimento.

Além disso, para validar a funcionalidades de corte/religação remotos será utilizado um relé de estado sólido para cortar o fornecimento de energia. Para verificar a presença da fase, a entrada deste sensor é conectada à saída do relé. Através do programa de teste do *Concentrador* são enviados comandos ao *Medidor* para cortar e verificar a situação do fornecimento.

2.2 Programa Teste *Concentrador*

O programa de teste do *Concentrador* envia um conjunto de caracteres pela porta serial do PC, quando requisitado através do teclado, e em seguida apresenta no monitor a resposta ao comando. Este programa foi escrito com o objetivo de interagir com o medidor eletrônico segundo um protocolo pré-definido, que será apresentado neste trabalho. O software depende de um operador para selecionar a opção desejada e assim definir o momento em que o pedido será transmitido na rede elétrica. A figura 4 mostra o programa com suas opções de teste.

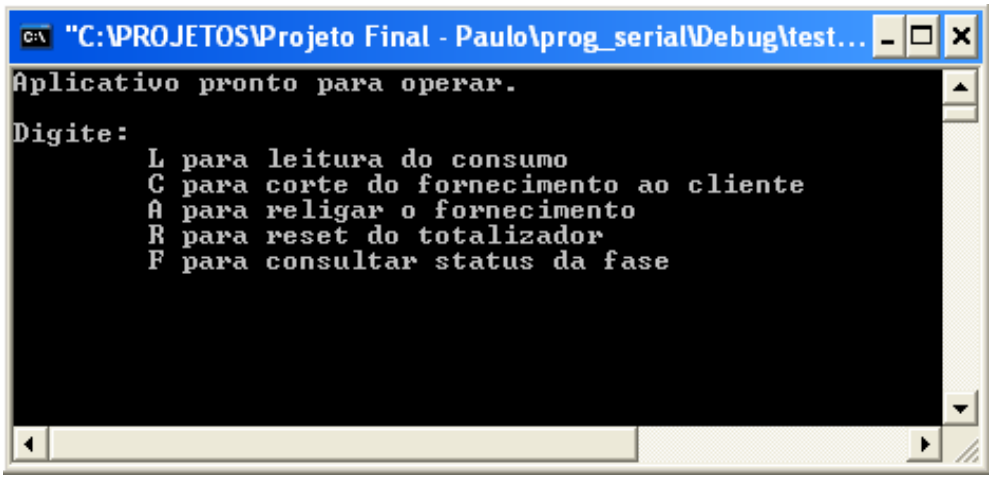


Figura 4 – Aplicativo para testar as funcionalidades do medidor.

Ao entrar com a opção, o programa envia o pedido, aguarda uma resposta do medidor e, em seguida, solicita uma confirmação desta resposta novamente ao medidor, esperando que este a valide. Desta forma, busca-se diminuir os erros nos comandos enviados pelo *Concentrador* ao *Medidor*, pois a rede elétrica é um meio físico suscetível a interferências eletromagnéticas que podem corromper os dados trafegados.

2.3 Medidor de energia elétrica

Como mencionado anteriormente, será desenvolvido um medidor de energia elétrica baseado em uma comunicação PLC. Este circuito utiliza o microcontrolador PIC16F88 para agregar as funcionalidades já mencionadas e possui três interfaces que estarão conectadas ao controlador interno: a interface de medição de energia elétrica, a de comunicação com a rede elétrica e a de corte do fornecimento, religação e presença de fase.

Para realizar a medição, será desenvolvido um circuito para aferir o consumo de energia e converter esta informação em um conjunto de pulsos. Estes pulsos serão enviados ao controlador que é capaz de totalizar e armazenar esta informação em uma memória não volátil. O controlador é responsável por seguir um protocolo através de uma comunicação serial pela rede elétrica. Este protocolo será definido neste trabalho e prevê todas as funcionalidades do sistema.

Para implementar a comunicação serial foram desenvolvidas rotinas em *Assembly* que utilizam um hardware interno ao microcontrolador, chamado de USART (Universal Sincrounous Assincrounous Receiver Transmitter). Este módulo é responsável por enviar de forma serial a informação, segundo uma configuração pré-estabelecida. Os sinais de TX e RX devem então ser conectados ao circuito do modem PLC. A figura 5 mostra o diagrama de blocos do *Medidor* e suas funcionalidades.

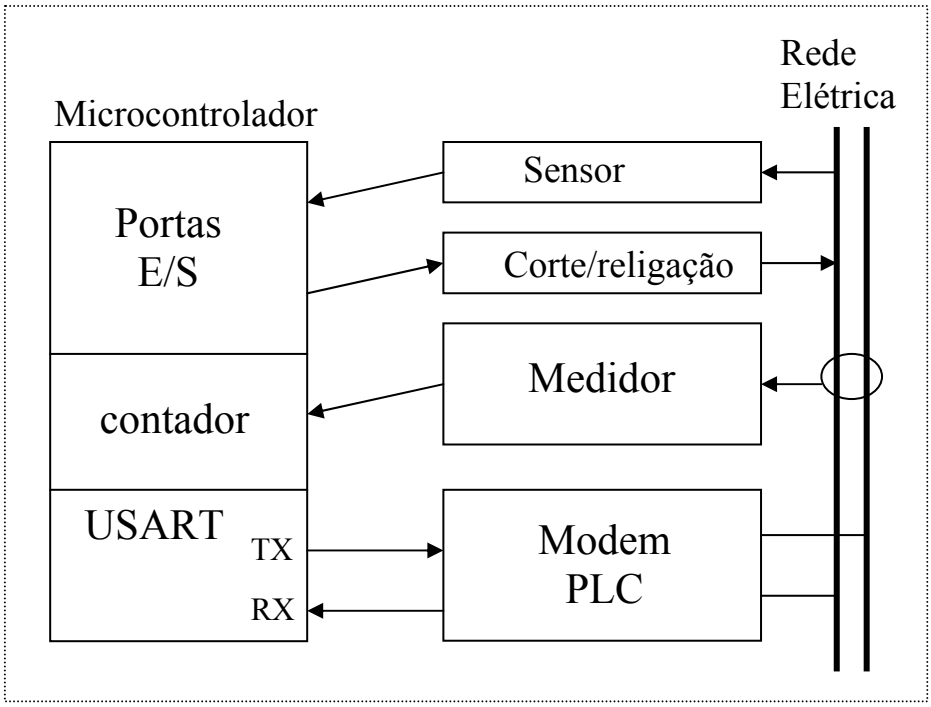


Figura 5 – Diagrama de blocos do medidor de energia e suas funcionalidades.

2.4 Conversor RS-232 PLC

Além do Medidor de energia será desenvolvida uma placa chamada de *Conversor RS-232 PLC,* que tem como objetivo realizar a interface entre o PC, executando o programa de teste do *Concentrador*, e a rede elétrica, onde estão conectados os demais medidores. Para se comunicar com essas duas frentes, este circuito é capaz de multiplexar os sinais do modem PLC e da porta RS-232. A figura 7 mostra a placa conversora conectada aos dois pontos.

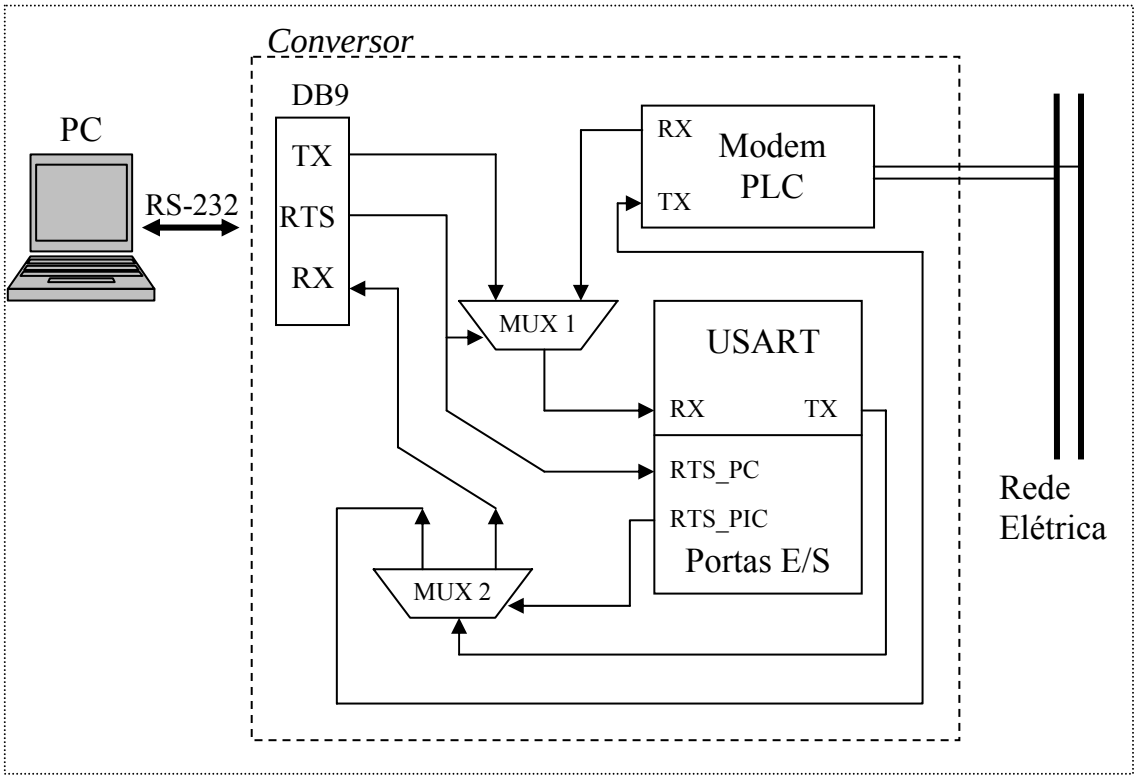


Figura 6 – Conexão Conversor RS-232 PLC ao Concentrador

Para transmitir uma informação para a rede elétrica, o programa do computador deve trocar o nível lógico do sinal RTS comutando o *mux 1* para que o microcontrolador receba no seu pino RX a informação proveniente do pino TX da porta serial do PC. O sinal de RTS do PC também é conectado a uma porta de entrada/ saída do PIC para sinalizar a requisição do PC. Através do sinal RTS_PC, o controlador comanda o *mux* 2 encaminhando a informação recebida para a rede elétrica.

Capítulo 3: O projeto do sistema

Esta seção tem o objetivo de descrever o funcionamento e projeto do sistema automático de medição de energia desenvolvido. Na primeira parte será calculado o circuito para tarifação de consumo e sua interface para o sistema controlador. Na segunda parte, o microcontrolador será projetado. Este circuito é responsável por gerenciar o protocolo de comunicação serial, agregar funcionalidades ao medidor e modular o sinal de transmissão. Em seguida, serão desenvolvidos os programas. Na quarta parte, os circuitos para comunicação PLC serão projetados.

3.1 O circuito medidor

O circuito medidor utiliza o chip ADE7757 [1] da *Analog Devices*, que foi escolhido por ser um medidor de watt-hora para sistemas monofásicos de baixo custo e por permitir uma precisão comparável aos medidores eletromecânicos atuais. Além disso, o integrado possui dois conversores AD (analógico-digital) para a conversão de voltagem e corrente e uma saída de pulso para realizar a interface com o microcontrolador. O fabricante ainda disponibilizou uma descrição detalhada de uma aplicação para medidores de baixo custo [2] (AN-679 *application note*) em que especifica detalhadamente como se realiza o projeto e quais os cuidados devem ser tomados. Quando esta documentação é estudada em conjunto com o manual do dispositivo [3] (ADE7757 *datasheet*) são encontradas todas as informações necessárias para se realizar o projeto do medidor. O esquema do medidor é mostrado na figura 7.

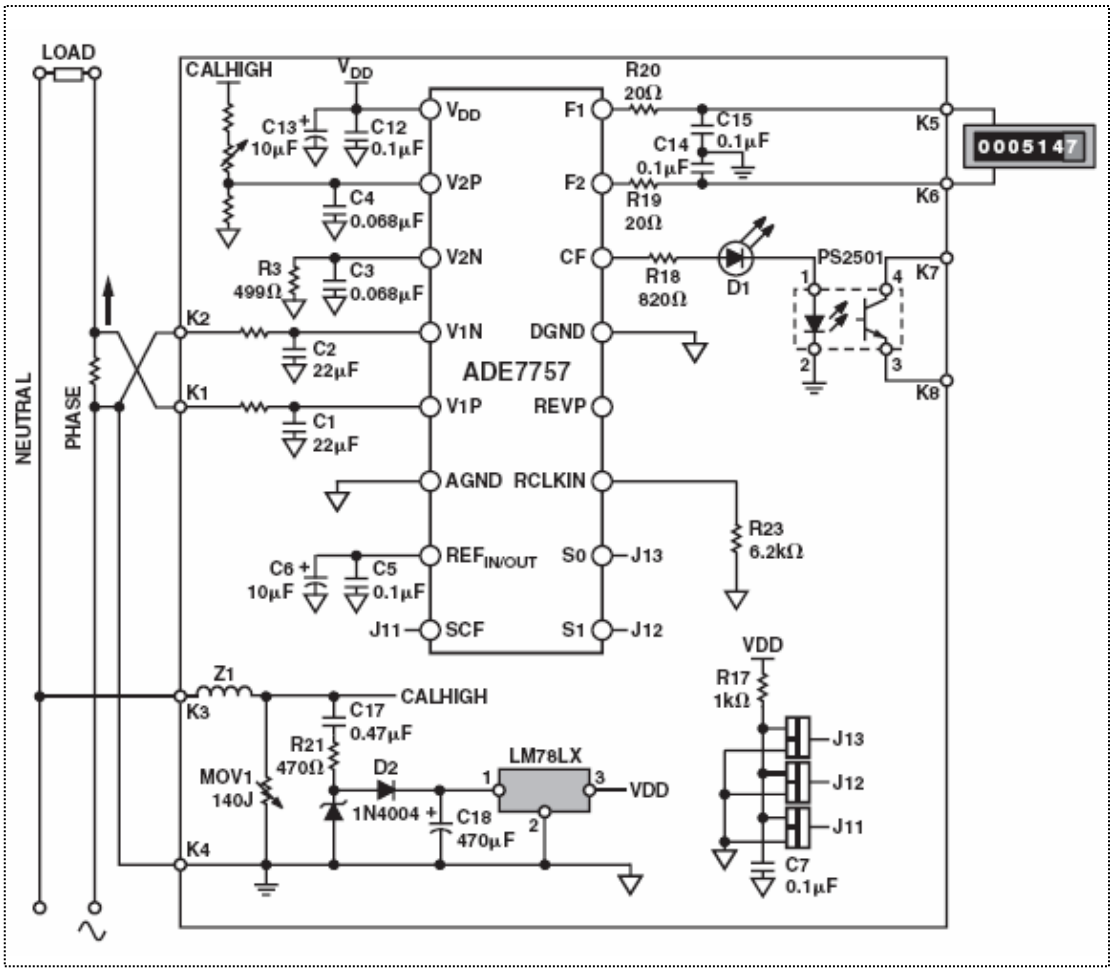


Figura 7 – Esquema elétrico do medidor monofásico.

O integrado possui internamente dois amplificadores diferenciais conectados a dois conversores AD de 16 bits, que estão destinados a medir os valores de tensão e corrente. Em seguida, os dados são encaminhados para um bloco de processamento digital de sinal em que são realizadas a filtragem, a correção de fase e a multiplicação dos sinais. A informação digital de consumo é então convertida em freqüência para ser enviada ao microcontrolador. A figura 8 mostra o diagrama de blocos do circuito integrado para medição de watt-hora.

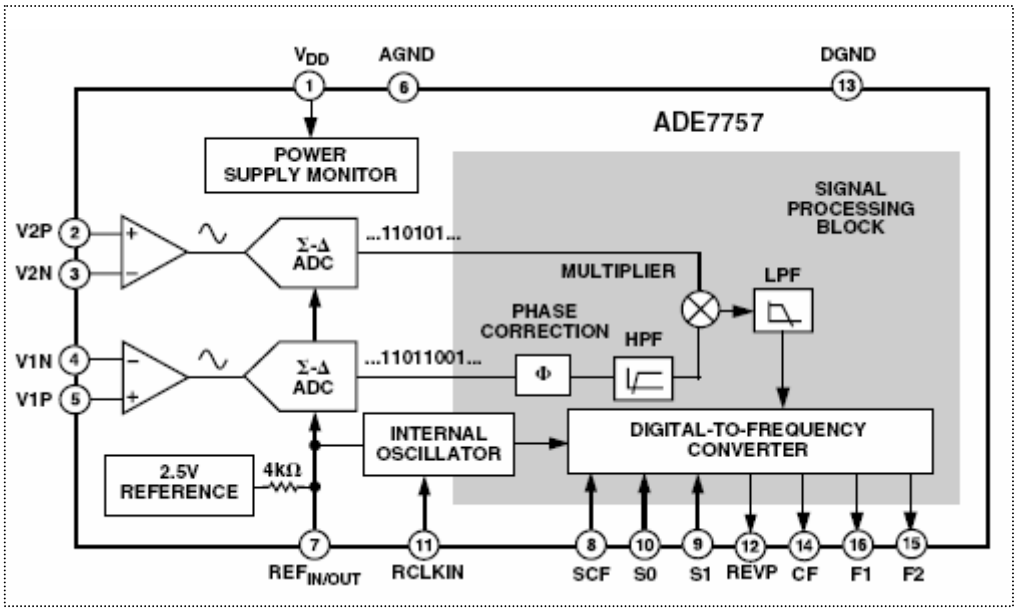


Figura 8 – Diagrama de blocos do circuito integrado ADE7757.

Para realizar a amostragem da tensão da linha é utilizada uma rede resistiva com o objetivo de atenuar essa tensão para uma ordem de dezenas de milivolts e entrar com esta informação no amplificador diferencial. A figura 9 mostra o esquema da rede resistiva.

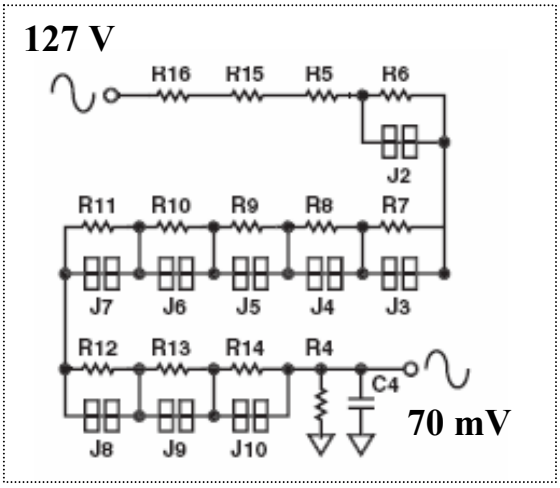


Figura 9 – Rede de atenuação para realizar a amostragem da tensão de linha.

A atenuação da tensão é obtida através de um circuito divisor resistivo. A associação série dos resistores define uma corrente entre a fase e o neutro, que ao passar pela resistência R4 gera uma tensão V_{R4} proporcional àquela presente na linha. Esta rede pode ser calculada para permitir uma calibração na tensão de saída. Isto ocorre porque o valor de resistência equivalente associado a R4 pode diminuir quando conectamos os *jumpers* da rede resistiva, colocando seus resistores em curto.

Ao se calcular os nove resistores começando por H6 e terminando por R14 (figura 9), cada um com valor igual à metade de seu predecessor, a tensão VR4 poderá ser calibrada com uma precisão definida pelo menor resistor da rede. Ao se visualizar os nove resistores como um sistema binário ligado/desligado, a tensão VR4 do caso real poderá ser ajustada, entre um valor máximo em 000000000, representando todos os resistores em curto, e um valor mínimo em 111111111, quando todos estiverem associados.

Com isso, a tensão VR4 é inserida na entrada V2P do amplificador diferencial do canal V2. A entrada V2N deve ser conectada ao terra do circuito. A figura 10 mostra como deve ser a conexão entre a rede resistiva e a entrada do canal V2. Deve-se notar que a rede resistiva está representada pelos resistores RA, RB e RF. É importante mencionar que, segundo o manual do fabricante, este canal suporta uma tensão de entrada em V2P máxima de 165mV de pico, positiva ou negativa, e um nível DC comum máximo de 25mV, que devem ser considerados durante o projeto.

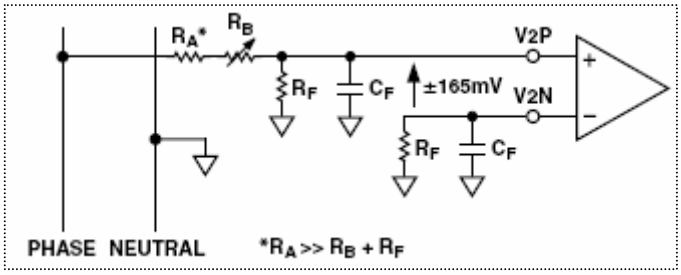


Figura 10 – Esquema de conexão entre a rede resistiva e a linha para o canal V2.

Para medir a corrente fornecida um resistor shunt é utilizado em série com linha, permitindo a conversão da corrente em uma tensão proporcional. A tensão máxima AC na entrada do canal V1, definida pelo manual do fabricante é de ± 30mV de pico, com um nível DC máximo de ± 6,25mV. A figura 11 mostra o esquema de conexão do resistor shunt e a linha.

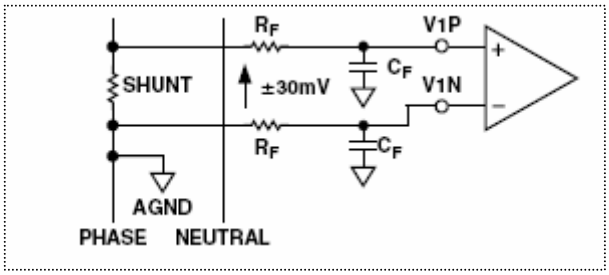


Figura 11 – Esquema de conexão entre a linha e o resistor shunt para o canal V1.

O resistor shunt é dimensionado para maximizar a faixa dinâmica no canal V1. Entretanto, o valor do resistor não pode ser alto e a potência dissipada elevada, pois esta será uma perda intrínseca ao método de medição do equipamento. Além disso, a alta dissipação de potência pode dificultar o projeto na medida em que, ao aquecer, há variação no valor da resistência. Por este motivo deve ser escolhido um shunt produzido com uma liga de baixo coeficiente de temperatura como, por exemplo, a *Manganina.*

Esse método de medição é interessante por ser puramente resistivo e permitir um baixo custo de implementação, evitando a utilização de componentes de custo mais elevado, como transformadores de tensão e corrente. Além disso, o sistema agrega uma capacidade de calibração através da rede resistiva. Um outro ponto é que resistores permitem uma melhor repetibilidade no momento da fabricação, fato importante quando se necessita de uma produção em larga escala.

Entretanto, o sistema resistivo está referenciado a um ponto da comum da rede, fase ou neutro, tornando todo o circuito eletricamente conectado a linha de baixa tensão.

Embora o desenvolvimento de um circuito para medir o consumo de energia envolva a proposta principal deste trabalho, o coração do projeto é sustentado por uma inteligência local que controla todas as funcionalidades do medidor.

Com isso, surge a necessidade de se utilizar um acoplador ótico[1] para enviar para a inteligência local o conjunto de pulsos referente ao consumo, uma vez que o microcontrolador encontra-se isolado da linha através de uma alimentação por um transformador. A figura 12 mostra esta situação.

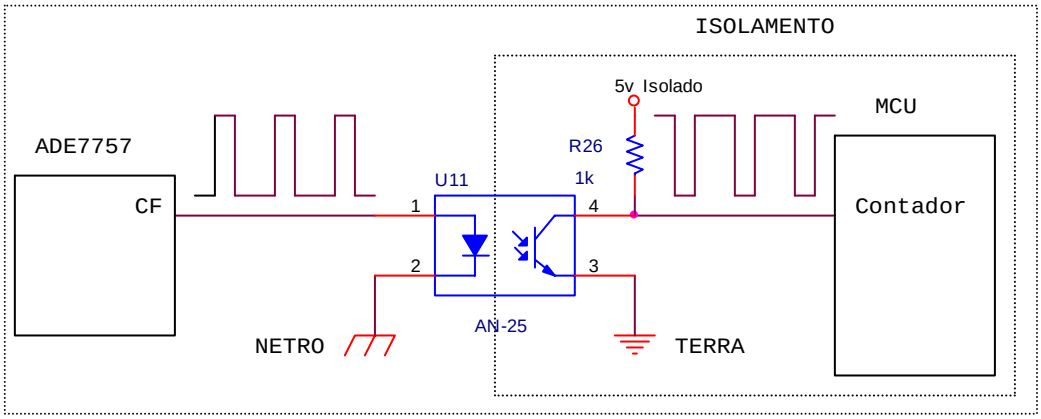


Figura 12 – Isolamento entre as alimentações do medidor e do controlador.

[1] Explicação do funcionamento do optoacoplador na seção 3.2.2 (figura 19).

O circuito do ADE7757 é alimentado pela própria rede elétrica.. Como o consumo é muito baixo, o fabricante sugere a utilização de uma fonte baseada em um divisor capacitivo, como mostra a figura 13.

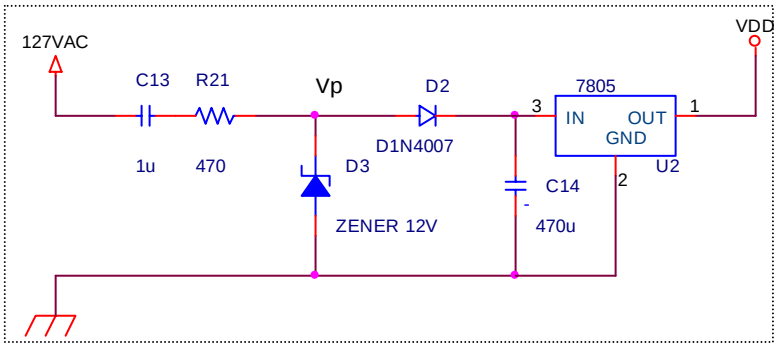


Figura 13 – Fonte de alimentação para o circuito ADE7757.

A associação de C13 e R21 causam um a redução na voltagem da rede, limitada em 12V no ponto Vp, pelo diodo D3. O regulador 7805 converte esta tensão em 5V para alimentar o ADE7757.

3.1.1 Projeto do circuito

O ADE7757 produz uma freqüência base de saída para um totalizador eletromecânico, que é proporcional ao produto das duas tensões referentes ao consumo de energia, V1 e V2. A amostragem do valor AC da linha e da corrente fornecida são estas tensões V1 e V2 e se relacionam segundo a equação 1.

$$Freq = \frac{515.84 \times V1_{rms} \times V2_{rms} \times F_{1-4}}{V_{ref}^{\ 2}}$$

Equação 1 – Relação entre as freqüências F1 e F2 e as tensões V1 e V2.

Nesta equação *Freq* é a freqüência de saída em hertz nos pinos F1 e F2. *V1rms* é a tensão diferencial em volt rms do canal V1 e *V2rms* é a tensão rms do canal V2. V_{REF} é a tensão de referência de 2,5V, no pino $V_{REF\ IN/OUT}$, fornecida pelo circuito do regulador interno. Para se obter uma maior acuidade e menor variação com a temperatura , pode-se utilizar um circuito externo para gerar esta referência. Entretanto, por simplicidade, neste projeto utilizamos o regulador interno. F_{1-4} é uma das 4 possíveis freqüências selecionadas pelas entradas lógicas S0 e S1 mostrada na tabela 1.

S1	S0	OSC Relation[1]	F_{1-4} at nominal OSC(Hz)[2]
0	0	$OSC/2^{19}$	0.86
0	1	$OSC/2^{18}$	1.72
1	0	$OSC/2^{17}$	3.44
1	1	$OSC/2^{16}$	6.86

Tabela 1 – Seleção de freqüências F_{1-4}, considerando oscilador a 450kHz.

Estas freqüências são geradas a partir de um oscilador interno do ADE7757 em 450 kHz, quando um resistor de valor 6,2 kΩ é utilizado na entrada RCLKIN. É importante lembrar que este resistor deve possuir baixa tolerância e baixa variação com a temperatura, para garantir estabilidade e linearidade de operação. Neste projeto foi utilizado um resistor de 6,2 kΩ e 0.1% de tolerância. A figura 14 mostra a curva de variação da freqüência do oscilador local em função do valor de RCLKIN.

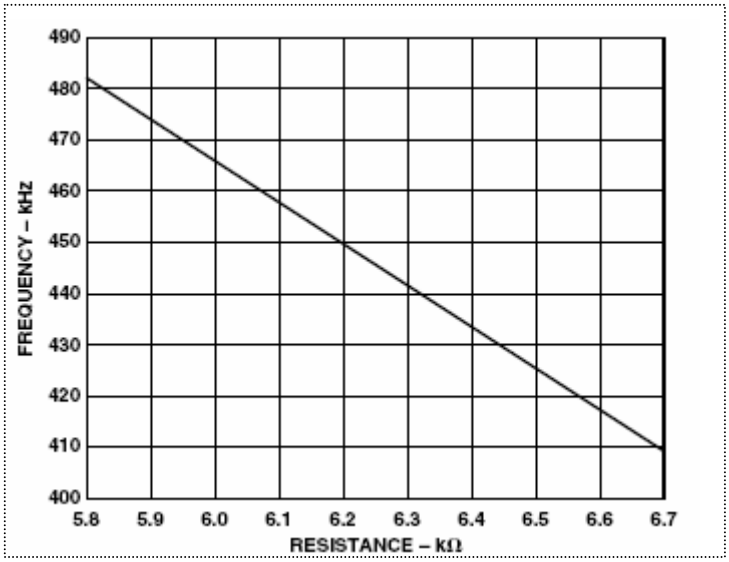


Figura 14 – Variação do oscilador em função de RCLKIN.

Os pinos F1 e F2 do ADE7757 são preparados para serem conectados a um totalizador eletromecânico que normalmente é o mostrador encontrado nos medidores analógicos. A cada 100 pulsos nestes pinos o totalizador deverá incrementar 1 kWh. Para efeito de simplicidade este mostrador analógico não será utilizado, pois o microcontrolador já possui a função de totalizar e armazenar o consumo. O integrado disponibiliza também uma saída no pino CF, que possui uma constante de valor mais elevado para ser conectado a controladores e pode ser ajustada de acordo com a tabela 2.

SCF	S1	S0	CF Max for AC Signals (Hz)*
1	0	0	128 × F1, F2 = 22.4
0	0	0	64 × F1, F2 = 11.2
1	0	1	64 × F1, F2 = 22.4
0	0	1	32 × F1, F2 = 11.2
1	1	0	32 × F1, F2 = 22.4
0	1	0	16 × F1, F2 = 11.2
1	1	1	16 × F1, F2 = 22.4
0	1	1	2048 × F1, F2 = 2.867 kHz

Tabela 2 – Ajuste da freqüência de saída para o microcontrolador.

A informação de consumo será verificada através de um pedido de leitura feito pelo *Concentrador* via PLC. O valor do totalizador de consumo será mostrado na tela do software de teste que estará sendo executado em um computador pessoal conectado a placa *Conversor RS-232 PLC*.

Para se realizar o projeto do medidor será considerado um sistema monofásico em uma rede de 127V, fornecendo uma corrente máxima de 70A. O valor do resistor shunt será calculado para que não seja ultrapassado o limite de 30mV na entrada do canal V1, como visto anteriormente. Com isso, o fabricante sugere que a tensão máxima na entrada deste canal seja próxima a 20mV, permitindo que haja uma margem de segurança. O valor de 300 μΩ foi escolhido por gerar uma tensão de 21mV em 70A. As especificações do projeto são mostradas na tabela 3.

Especificações	
Tensão da linha	127 V
Corrente Máxima	70 A
Constante do totalizador	100 imp/ kWh
Resistor Shunt	300 μΩ

Tabela 3 – Especificações do projeto.

O medidor ADE7757 garante uma acuidade melhor que 0.1% em uma faixa dinâmica de 1:400. Isto significa que o projeto deve ser feito utilizando-se de uma corrente base Ib, em que 70A corresponda o nível de corrente máximo para a faixa dinâmica. Os cálculos do projeto da rede resistiva encontram-se ao final deste trabalho no Apêndice 1. O esquemático completo do circuito de medição projetado encontra-se no Anexo 2, página 1, da placa *Medidor*.

3.2 Microcontrolador

A opção de se utilizar dispositivos programáveis, como microcontroladores ou microprocessadores, em um projeto, vem da necessidade de se controlar e gerenciar a funcionalidades do mesmo. A vantagem inerente aos microcontroladores sobre os microprocessadores é que aqueles são circuitos simples de serem utilizados e integram memória de programa, memória volátil e não-volátil, CPU, portas de entrada e saída, contadores e temporizadores e módulos para comunicação síncrona e assíncrona em um único chip. Dessa forma, para que a CPU funcione e execute a lógica de estados, não há necessidade de se adicionar qualquer estrutura externa, exceto um cristal oscilador, uma vez que todos os circuitos periféricos já estão contidos dentro da mesma pastilha de silício. Essa característica é muito importante porque agrega simplicidade e robustez ao projeto, pois todas as estruturas já se encontram em um único componente, diminuindo a probabilidade de problemas com funcionamento incorreto de algum circuito periférico.

Um outro ponto importante é que, ao se optar por trabalhar com essa solução, deve-se ter em mente que o valor intelectual do projeto encontra-se, em sua maior parte, no código fonte, em detrimento do projeto do circuito.

A figura 15 mostra a pinagem do microcontrolador PIC16F88, da *Microchip.* Pode-se notar o reduzido número de pinos.

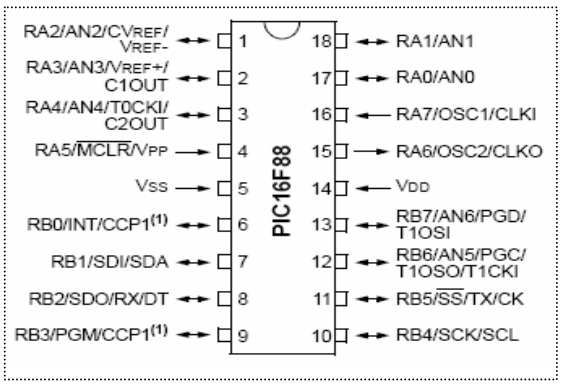


Figura 15 – Diagrama de pinos do microcontrolador PIC16F88

Essa inteligência local é utilizada para gerenciar o medidor de energia e seus periféricos, permitindo agregar funcionalidades como armazenar o totalizador de consumo em memória não volátil, se comunicar com uma unidade concentradora através de um protocolo predefinido, cortar ou religar o fornecimento de energia quando requisitado e detectar falhas na entrega da energia. Para satisfazer esse requisito, foi

utilizado o microcontrolador PIC16F88[3], da *Microchip Technology Inc*.[4], chamado de PIC. Esse controlador foi escolhido por possuir os periféricos necessários e por ser um dispositivo de fácil programação e utilização. As características do PIC escolhido que atendem as necessidades do projeto estão descritas na tabela 4.

Características
Oscilador de até 20MHz
USART para comunicação serial
EEPROM (256 bytes)
Memória FLASH de programa
Contadores/temporizadores

Tabela 4 – Característica do PIC16F88 necessárias a este projeto

3.2.1 Funcionamento do microcontrolador

O PIC é uma máquina seqüêncial que executa uma instrução de programa em um ciclo de *clock*. O tempo do relógio é definido a partir da freqüência de um cristal conectado aos pinos OSC1 e OSC2 do integrado. Um cristal de 20MHz foi escolhido para permitir o máximo de instruções por segundo. Esta condição é necessária, pois o próprio controlador será utilizado para modular o sinal de comunicação, como será visto mais a frente, na seção 3.4, o que permite atingir as freqüências desejadas na comunicação RF. O circuito do microcontrolador com o oscilador pode ser visualizado na figura 16.

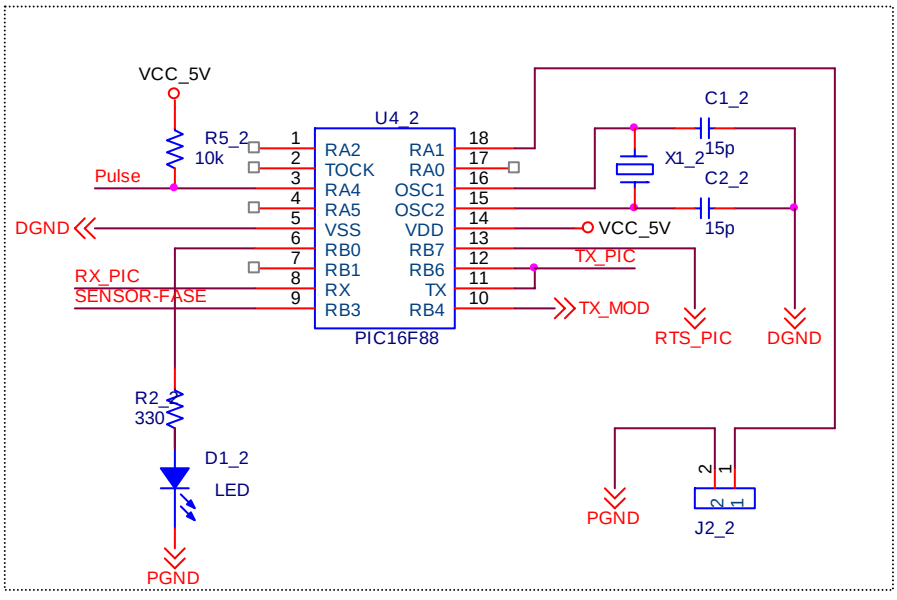


Figura 16 – Circuito para esquemático de funcionamento do microcontrolador.

3.2.2 O circuito de corte, religação e sensoriamento

Para cortar e religar o fornecimento de energia, bem como sentir a presença da fase, foram utilizadas as portas bidirecionais do PIC. Através de uma configuração dos registradores de funções especiais os pinos do microcontrolador foram selecionados como entrada ou saída de sinais digitais. O dispositivo possui duas portas de 8 bits com dois registradores relacionados a estas portas.

A porta RA1, pino 18, foi selecionada como saída para o controlar o relé, que fecha sua chave com 5V, ligando a energia, e a corta com 0V. Este relé deve ser conectado em série com a malha de fornecimento e neutro como mostrado na figura 17.

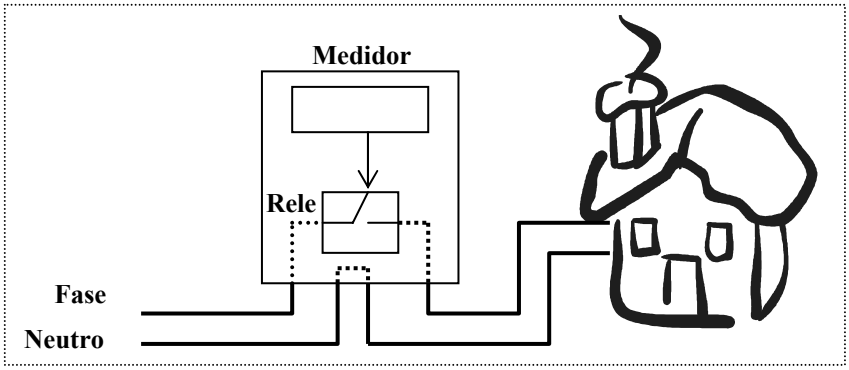


Figura 17 – Conexão série do relé de corte e religação.

A opção de um relé de estado sólido da Novus Produtos Eletrônicos LTDA [5] foi feita por este ser um elemento de simples utilização. O dispositivo já possui internamente um sistema lógico, isolado eletricamente da chave de comutação. Assim, não há necessidade de agregar componentes externos, podendo-se conectar a saída do PIC diretamente ao relé. O firmware grava o estado do fornecimento internamente em uma memória não-volátil chamada de *EEPROM,* disponível internamente no PIC. Com isso, caso a alimentação seja interrompida o circuito reestabelece o estado anterior ao acessar esta memória. A figura 18 permite visualizar a foto do relé de estado sólido.

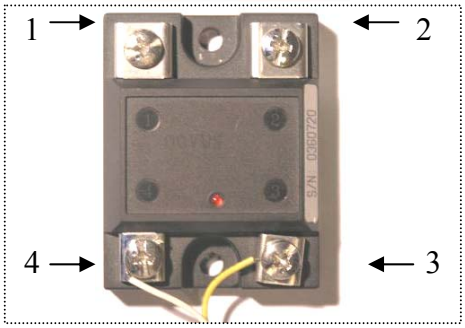


Figura 18 – Foto do relé de estado sólido

Para instalar o relé no medidor conectamos o pino 3 ao terra e o pino 4 a saída do microcontrolador, pino 18. Enquanto o PIC mantiver o pino 4 em 5V o relé estará fechado e permitirá a passagem de corrente entre os pinos 1 e 2. Quando o pino 4 estiver com 0V o dispositivo interrompe a passagem de corrente entre os pinos 1 e 2.

Para desenvolver um sensor de presença de fase foi utilizado um optoacoplador. Este elemento é composto por um por led e um fototransistor em um circuito integrado de quatro ou seis pinos. Apesar de ambos se encontrarem no mesmo chip, estes dois componentes estão áreas da pastilha que são isoladas eletricamente. O funcionamento desse circuito, mostrado na figura 20, é simples: quando o led está aceso, o fototransistor entra em funcionamento e sai da sua região de corte para a região de saturação, permitindo que circule uma corrente do pino 4 para o 3. Quando o led está apagado o pino 4 encontra-se em alta impedância.

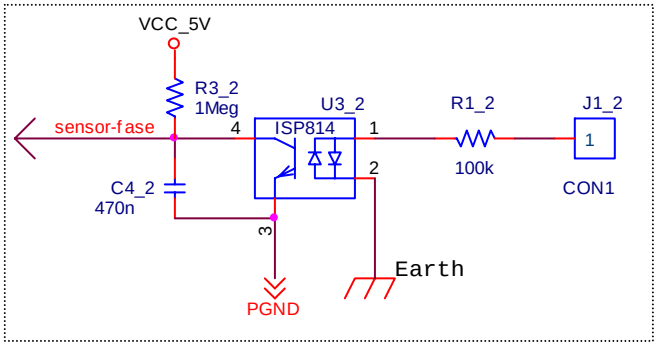


Figura 19 – Circuito para sensoriamento da fase

A referência aparece na figura 20 como *Earth*, pois já vem do neutro da rede elétrica conectada na placa do medidor. Para sentir a fase desejada basta ligá-la ao conector *J1_2*, em uma malha composta por um resistor de 100kΩ e dois leds internos ao circuito, até retornar ao neutro. Na medida em que há tensão de 127VAC da rede, em *J1_2* existe uma corrente ac próxima a 1mA, acendendo um led no semi-ciclo positivo e o outro no negativo. Quando um dos leds está aceso, a impedância vista entre os pinos *4* e *3* é muito pequena e a saída do circuito *sensor de fase* está em estado baixo, próxima a 0V. Quando não houver tensão da rede, o transistor estará cortado e a saída do circuito *sensor de fase* estará em estado alto, próxima a 5 volts.

3.2.3 O totalizador com memória não volátil

O totalizador de consumo é implementado por firmware e utiliza um contador interno, chamado de *timer 0 (TMR0)*. O PIC permite configurar este contador para ser incrementado quando ocorrer uma transição positiva de um sinal de uma fonte externa conectada ao pino 3 (*TOCK*).

O número de transições positivas é armazenada em um registrador da memória RAM com tamanho de 40 bits. Para que o valor do consumo não seja perdido, caso a alimentação do circuito seja interrompida, o firmware grava o conteúdo do registrador de cinco bytes na memória *EEPROM* interna, a cada novo incremento de *TMR0*.

3.2.4 O protocolo de comunicação

Para tornar possível trocar informações entre o microcontrolador e o concentrador, foi necessário criar um protocolo específico. O protocolo desenvolvido consiste no envio de mensagens formadas por caracteres ASCII.

Cada pacote deve conter caracteres de início de bloco, bytes de endereço, caractere correspondente ao tipo de requisição e bytes que carregam a informação, como o conteúdo do totalizador. A tabela 5 mostra os caracteres ASCII utilizados, seus significados e o valor decimal correspondente.

Caractere	ASCII	Sinificado	Valor
BEGIN	@	início de bloco	64
END1	1	endereço 1	49
END2	2	endereço 2	50
CMD1	L	operação leitura	112
CMD1, CMD2	C, T	operação corte	0
CMD1, CMD2	A, C	operação religação	38
CMD1, CMD2	R, E	operação reset	36
CMD1, CMD2	F, A	operação estado fase	35
ACK1	?	requisição de confirmação	63
ACK2	!	envio de confirmação	2
ACK3	O, K	execução de operação	0
B0, B1, B2, B3, B4	---	bytes B0 a B4 totalizador	0 a 255

Tabela 5 – Caracteres utilizados no protocolo de comunicação

Este protocolo prevê cinco tipos diferentes de operações que estão relacionadas às funcionalidades do medidor. Toda mensagem enviada pelo concentrador ao medidor deverá conter três caracteres "@" para indicar o início do pacote. Dessa forma, o controlador pode ser programado para validar o bloco ao encontrar estes três bytes de início. Em seguida, dois bytes de endereço, *END1* e *END2*, deverão ser enviados para identificar o medidor em questão. O próximo byte indicará o comando a ser executado na requisição: ao enviar um *L* deseja-se realizar um pedido de leitura do totalizador, ao enviar um *C* e *T*, deseja-se cortar o fornecimento e ao enviar um *A* e *C*, religá-lo. Para reiniciar o totalizador deve-se enviar um *R* e *E* e para consultar o estado da fase envia-se um *F* e *A*. As tabelas 6 e 7 mostram as seqüências de caracteres utilizados nas operações do medidor.

#	**Remetente**	**Mensagem**
M1	Concentrador	@ @ @ 1 2 L
M2	Medidor	@ @ @ 1 2 L B0 B1 B2 B3 B4?
M3	Concentrador	@ @ @ 1 2 L B0 B1 B2 B3 B4 !
M4	Medidor	@ @ @ 1 2 L O K

Tabela 6 – Seqüência de caracteres para operação de leitura

#	**Remetente**	**Mensagem**
M1	Concentrador	@ @ @ 1 2 CMD1 CMD2
M2	Medidor	@ @ @ 1 2 CMD1 CMD2 ?
M3	Concentrador	@ @ @ 1 2 CMD1 CMD2 !
M4	Medidor	@ @ @ 1 2 CMD1 CMD2 O K

Tabela 7- Seqüência de caracteres para operação de outros comandos do *Medidor*

O protocolo exige que cada requisição de comando seja seguida de uma confirmação, como pode ser visualizado nas tabelas 6 e 7. Ao receber um conjunto de caracteres que definem uma dada operação, o controlador deve compreender este pedido e em seguida envia uma mensagem, terminada pelo caractere "?", perguntando se é realmente o que se deseja fazer. Em caso afirmativo o PIC receberá uma segunda mensagem vindo do concentrador contendo o caractere "!", e, após executar a operação, enviará uma confirmação contendo "O" e "K".

3.2.5 A comunicação serial

Para comunicação entre o medidor e o concentrador foram desenvolvidas rotinas em *Assembly* que utilizam o hardware do microcontrolador dedicado a comunicação serial. Os registradores para funções especiais do PIC permitem configurar a taxa de bits, a opção de paridade, sincronismo e número de bits.

Optou-se por uma taxa de 1200 bits por segundo, por ser a menor que se pode conseguir quando se utiliza oscilador do clock em 20 MHz. A velocidade de comunicação é relevante quando se realiza uma modulação FSK, pois está diretamente relacionada à banda utilizada e quanto mais estreita, maior será sua razão sinal/ruído, SNR. Como neste caso a quantidade de informação por período de tempo não é crítica, esse requisito foi sacrificado em prol de uma menor taxa de erro dos bits transmitidos, BER. A tabela 8 lista as características da comunicação serial deste projeto.

Taxa de bits	1200 bits/s
Bit de paridade	Não
Modo de comunicação	Assíncrona
Time-out entre bytes	100ms
Time-out entre mensagens	500ms

Tabela 8 – características da comunicação serial

3.3 Programas

Este projeto é composto por dois firmwares e um software. Um firmware, chamado de *Medidor*, executa o programa no medidor eletrônico implementando o protocolo para comunicação remota pela rede e suas funcionalidades.

O outro firmware, chamado de *Modem*, executa o programa na placa *Conversor RS-232 PLC* e tem como objetivo receber a informação do *Concentrador* e transmití-la na linha, utilizando a técnica PLC. Por fim, o software, chamado de *Teste Concentrador*, tem o objetivo de interagir com o medidor.

3.3.1 Simulador e compilador

Ambos os firmware foram desenvolvidos em linguagem Assembly utilizando o *PIC Simulator IDE* [6]. Este aplicativo para Windows é um ambiente que integra um

compilador e editor Assembly e permite simular as várias funcionalidades do PIC, como: portas de entrada e saída, valor dos registradores, comunicação serial, gravações em *EEPROM*, timers, e rotinas de interrupção, entre outras.

A grande vantagem em se poder simular os programas está se conseguir se reduzir o tempo gasto com depuração. Não há necessidade de se executar o código no microcontrolador para saber se está bem estruturado. Por exemplo, com o simulador foi possivel enviar todos os caracteres do protocolo e ter certeza de que a sequencia lógica até a execução da operação estava correta. A figura 20 permite visualizar o ambiente integrado.

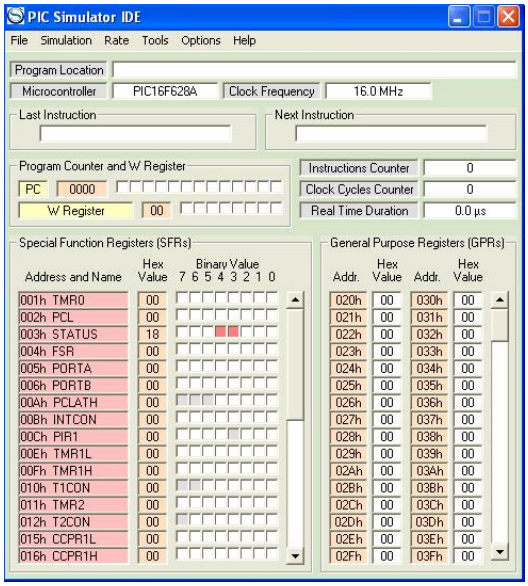


Figura 20 – PIC simulator IDE

O software para o concentrador foi desenvolvido em linguagem *C* utilizando o *Microsoft Visual C++*, tendo o objetivo de implementar todos os comandos que o medidor deverá responder.

3.3.2 Firmware do Medidor

O programa para ser executado no Medidor de Energia foi desenvolvido a partir das especificações do protocolo, das características da porta serial e do totalizador. O firmware configura o microcontrolador para utilizar as portas de entrada e saída, a porta serial e o contador *timer 0*, carregando o totalizador de 40 bits com o valor inicial da *EEPROM*.

Em seguida, o firware entra na rotina principal em que testa a chegada de um caractere pela porta serial e a mudança no contador do totalizador de consumo, gravando a nova informação em memória não-volátil. O fluxograma das condições iniciais e da rotina principal é mostrado na figura 21.

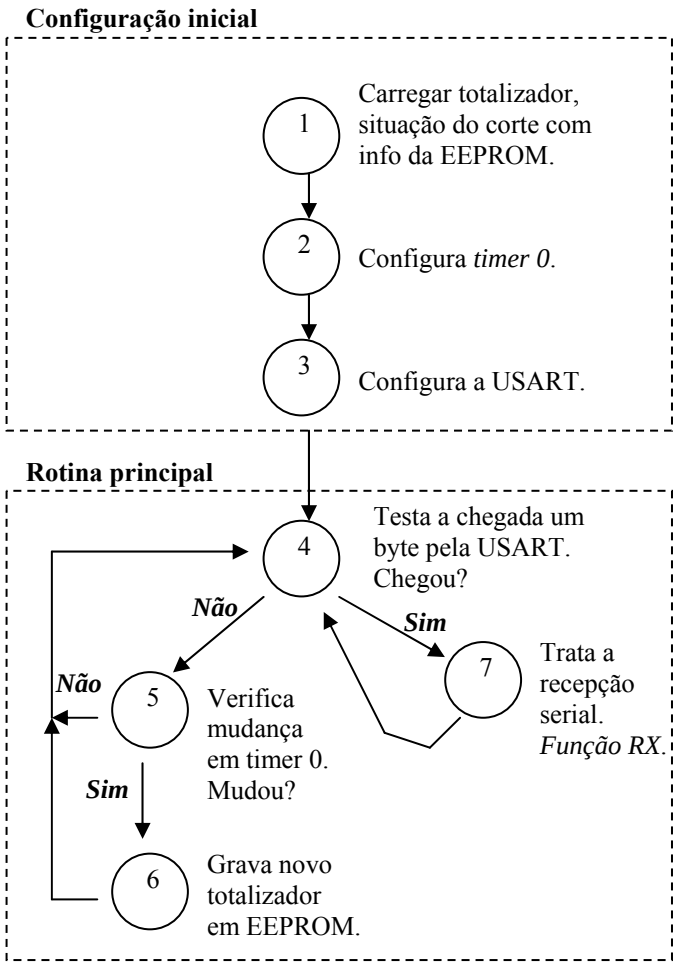


Figura 21 – Fluxograma da rotina principal do firmware Medidor

A rotina de tratamento da recepção serial, chamada de *Função RX*, no estado *7*, foi desenvolvida para seguir o protocolo de comunicação. Esta função configura o *timer 1* para ser incrementado internamente pelo clock, utilizando-o para implementar uma saída desta rotina por *time-out*.

Em seguida, testa a chegada do inicio de bloco, composto por *@@@ END1 END2*, verificando se *END1* e *END2* correspondem ao endereço do medidor. Confirmando que a mensagem destina-se a ele, o controlador testa o próximo caractere, *CMD*, para enviar uma requisição de confirmação.

Após transmitir o sinal, a função entra em um estado para aguardar a chegada do pedido. Chegando os caracteres que definem o início de bloco o programa testa o próximo byte, caractere de comando, para executar a operação desejada e enviar a resposta da execução do comando. O fluxograma da rotina de recepção e execução dos comandos é apresentado na figura 22.

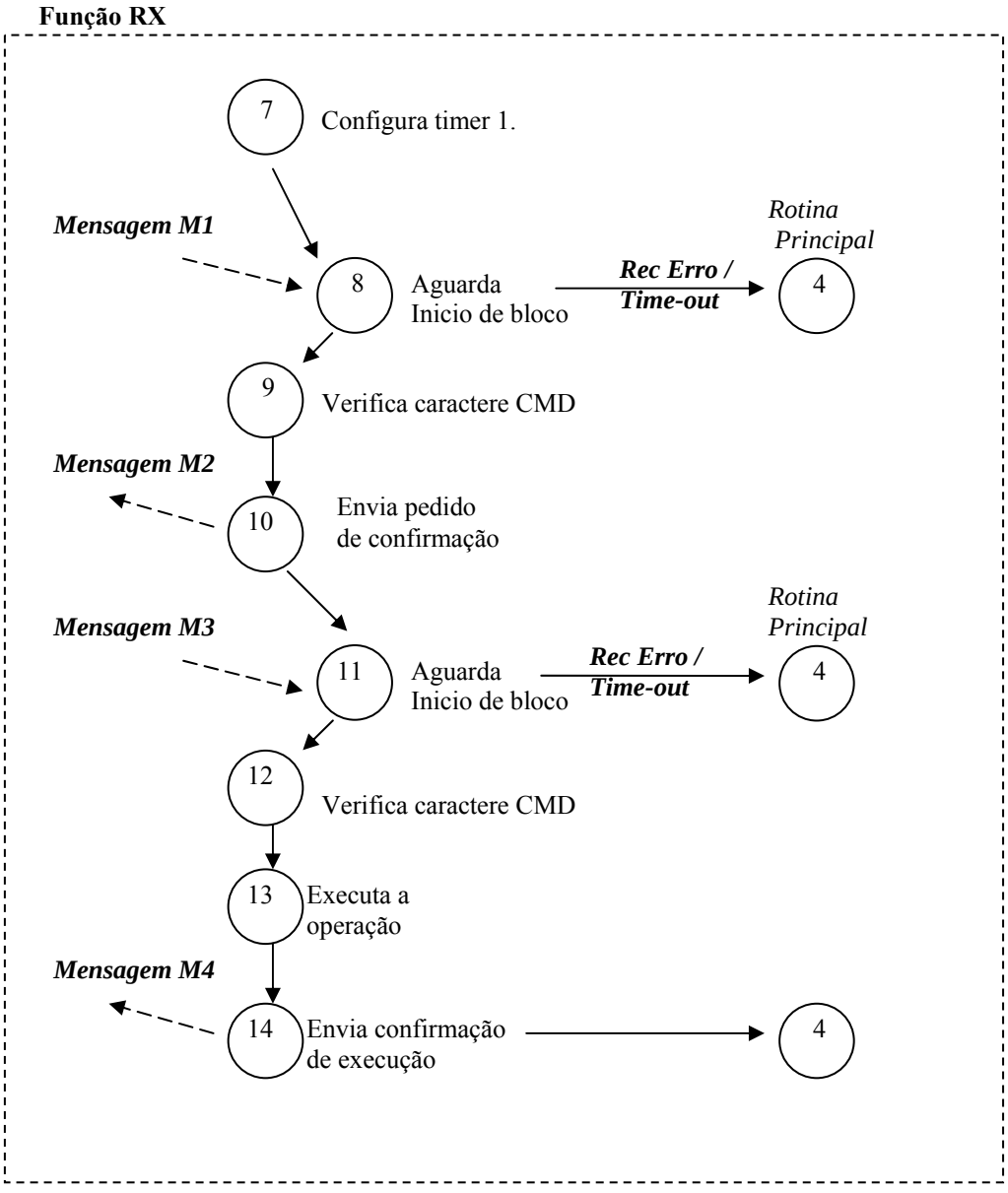


Figura 22 – Fluxograma da *Função RX*

3.3.3 Firmware do Conversor RS-232 PLC

O firmware do Conversor RS-232 PLC tem a capacidade de se comunicar tanto com o *Concentrador* como com os demais medidores ligados na rede elétrica, via PLC.

Entretanto, por possuir uma única porta serial, o PIC necessita multiplexar os sinais de Rx e Tx vindo do PC e da rede elétrica.

O estado *default* do Conversor é habilitado para recepção de sinais da rede elétrica. Nessa situação, o MUX1 dirige os dados, de forma contínua, para a entrada Rx do PIC. Se os dados recebidos são válidos, o PIC os transmite para o *Concentrador* via MUX2.

Para realizar uma transmissão, o *Concentrador* comanda o MUX1 através do pino RTS_PC. Esse mesmo sinal informa o PIC que o sinal serial provém do PC. Ao concluir a recepção, o microcontrolador comanda o MUX2, através do sinal RTS_PIC, para inserir a informação na rede PLC.

Para multiplexar os sinais foi utilizado o circuito integrado 74HC4053 [7], que possui 3 mux analógicos, dos quais são utilizados apenas dois. A lógica de conexão entre estes sinais e os multiplexadores é mostrada na figura 23.

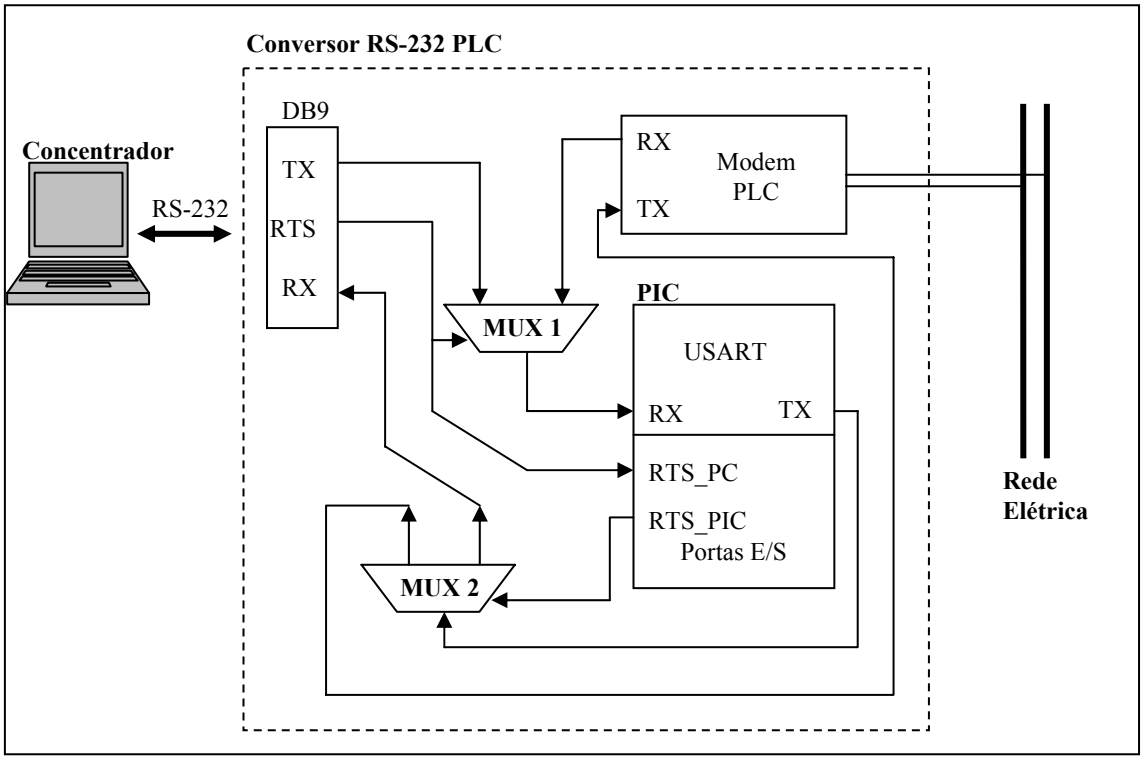


Figura 23 – Conexão entre os sinais do PC e da rede o microcontrolador

Para o funcionamento do *Conversor* o firmware configura a porta serial e entra na rotina principal, em que espera a chegada de um byte na USART. Quando o buffer está cheio o programa é desviado para tratar a recepção.

O PIC aguarda a chegada dos caracteres de início de bloco, verificando se a entrada RTS_PC está ativa e saindo da rotina por erro ou por *time-out,* caso ultrapasse o tempo previsto para espera. Se houver sucesso nestas etapas o microcontrolador amazena os bytes de chegada internamente e em seguida envia pela porta TX. A figura 24 mostra o fluxograma desta operação.

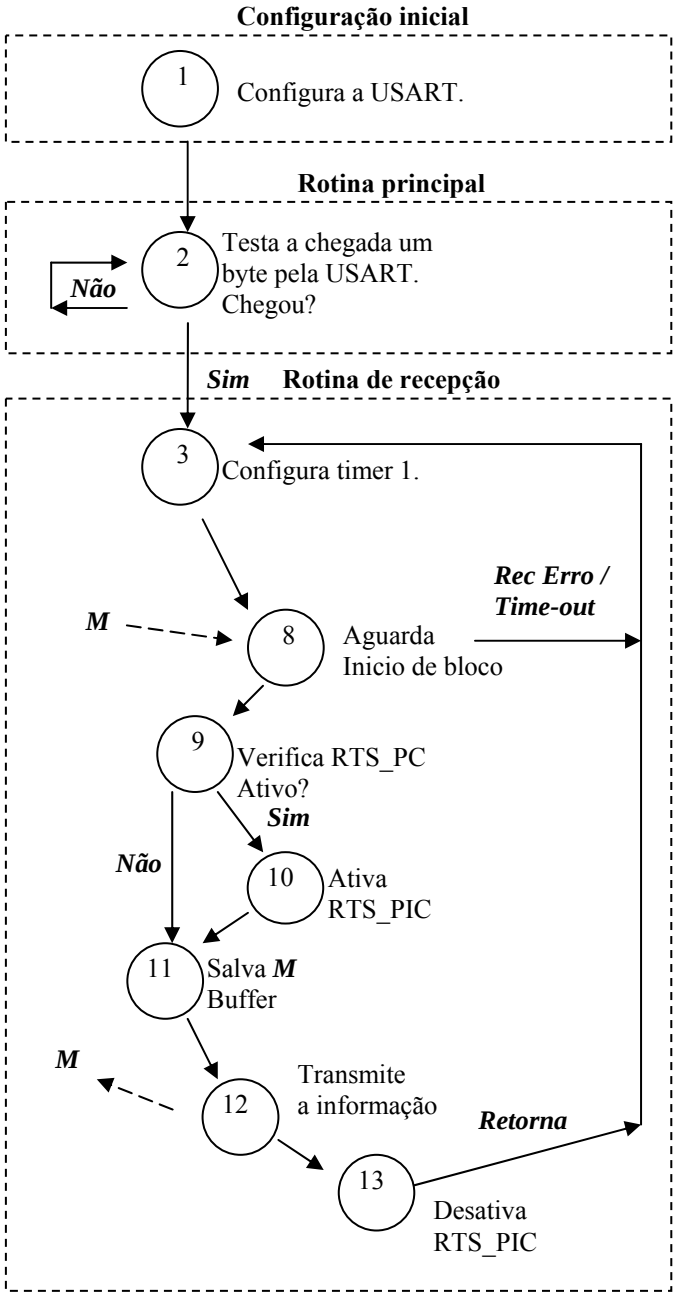


Figura 24 – Fluxograma do programa Conversor RS-232 PLC

3.3.4 Software do Concentrador

Foi desenvolvido um software para computador pessoal, em linguagem *C*, sem interface gráfica, com o objetivo de tornar possível testar todas a operação do medidor eletrônico.

Este programa, chamado de *Teste Concentrador*, inicia com um menu de opções em que se pode escolher a letra correspondente ao comando desejado. Uma vez escolhida a operação, o computador ativa o sinal de RTS, envia o conjunto de caracteres correspondente ao protocolo e, após a transmissão concluída, desliga o sinal de RTS. Ao receber um pedido de confirmação vindo do medidor, o PC envia uma nova mensagem confirmando a operação para, finalmente, receber o ultimo último conjunto de caracteres, confimar o sucesso e voltar ao menu inicial.

Caso receba alguma mensagem errada, o programa acusa a ocorrência de um erro e em seguida volta ao menu principal. A figura 25 mostra o fluxograna desse programa.

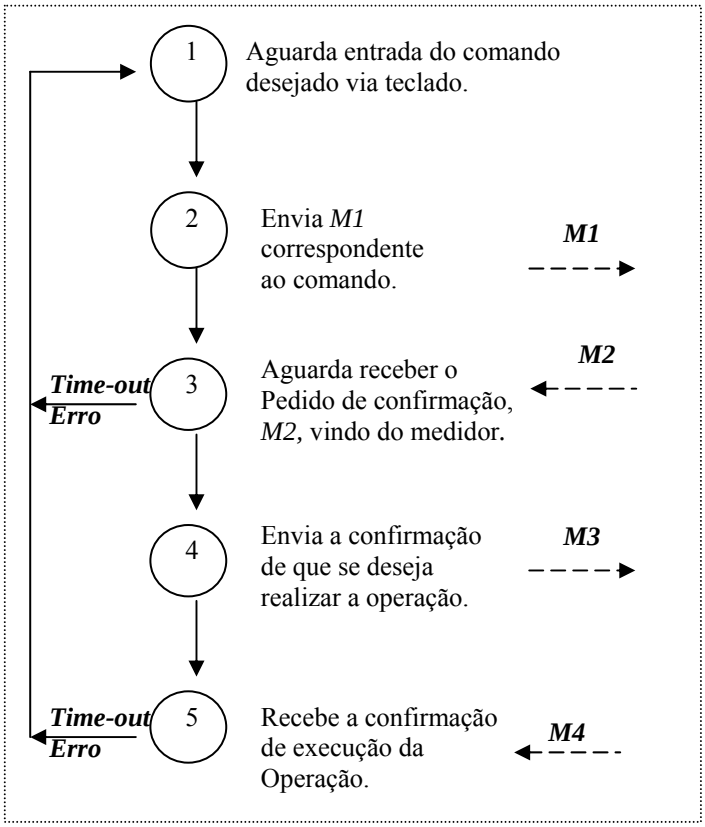


Figura 25 – Fluxograma do programa *Teste Concentrador*.

3.4 Comunicação PLC

A tensão da rede elétrica é formada por uma senóide de 60Hz, com 127V *rms* de amplitude. Para se comunicar através desta linha, o sistema utiliza uma modulação FSK, que se soma à tensão da rede. Assim, esse sinal transmite a informação por desvio de freqüência, no qual um tom de 111kHz caracteriza o envio do nível lógico baixo, e 115kHz, o alto.

O circuito receptor se baseia em um demodulador FM e por este motivo considera a freqüência média do modulador com sendo a portadora do sinal de comunicação. Com isso, utilizando um comparador na saída do receptor, pode-se recuperar a informação para sua forma digital original.

Uma instalação elétrica convencional possui um comportamento peculiar porque a dinâmica da ligação das cargas provoca uma variação da impedância ao longo do tempo, causando alterações na potência sinal.

Um outro ponto importante é que a taxa utilizada é baixa, 1200 bits por segundo, se comparado com as conexões de Internet disponíveis nos dias de hoje. Todavia, a comunicação banda estreita é interessante, pois permite aumentar a razão sinal ruído.

3.4.1 Transmissor FSK

Para modular o sinal de transmissão foi desenvolvida uma rotina que utiliza uma porta de saída e uma de entrada do PIC. Para enviar um byte, escrevemos estes bits em um registrador da porta serial, liberando o controlador para realizar outras tarefas. Através da conexão do sinal de *TX*, gerado pela *USART*, em uma porta de entrada, o firmware pode detectar o nível lógico do sinal serial. Com isso, trocando o estado da porta de saída, o PIC é capaz de gerar as duas freqüências desejadas para a comunicação. A figura 27 mostra o fluxograma da rotina de transmissão. A figura 26 mostra o esquema de realimentação do sinal serial.

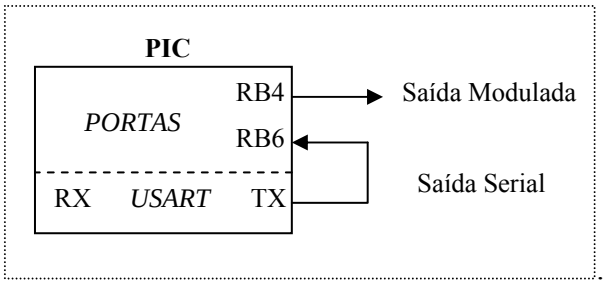


Figura 26 – Conexão da saída serial do PIC para modulação por firmware

A rotina de modulação carrega o byte que se deseja enviar no registrador *TXREG* dedicado, iniciando o processo de transmissão assíncrona. Em seguida, o programa analisa a entrada do sinal *TX* através da porta RB6. Caso esteja em nível lógico baixo, o *firmware* entra em um estado para aguardar um período de 1/115000 segundos, ou, se estiver em alto, espera 1/111000 segundos.

Após este intervalo de tempo, através de um *flag* de interrupção o controlador verifica o termino da transmissão. Se chegar ao fim, o PIC retorna à função principal, senão volta ao estado de verificação do nível de entrada para então trocar o nível da saída RB6 e aguardar o valor correspondente de tempo. A figura 27 mostra o fluxograma da rotina de modulação.

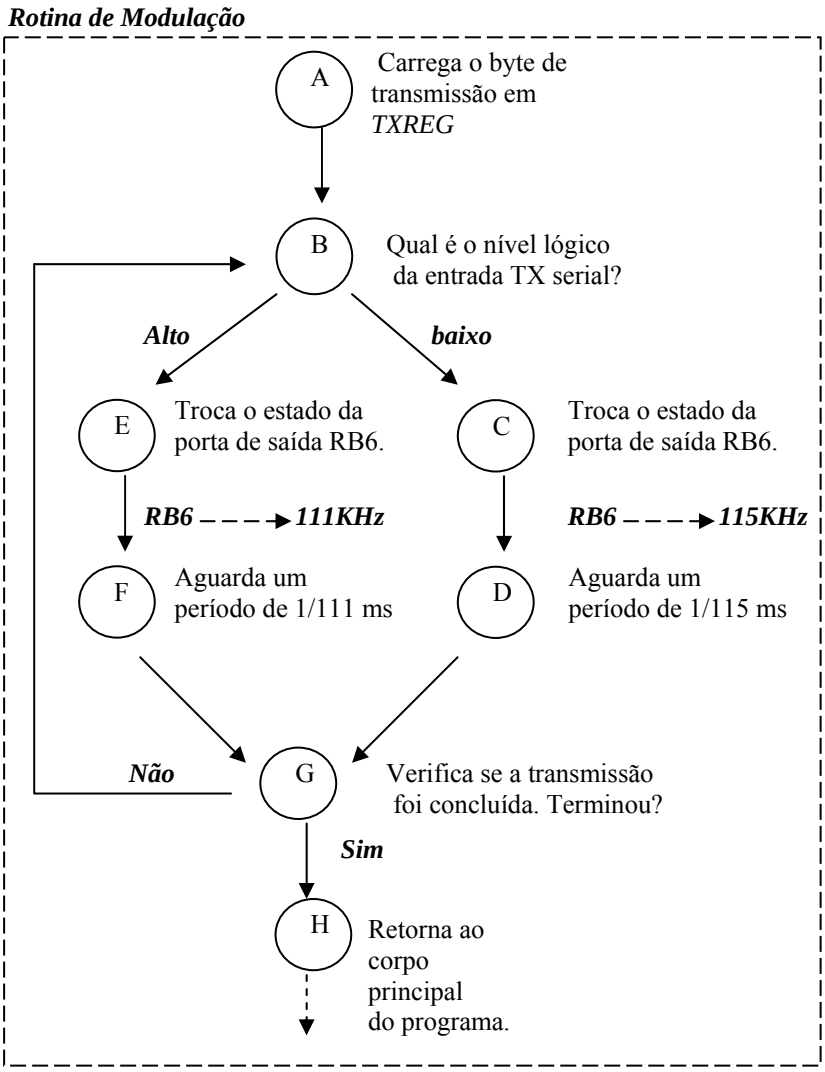


Figura 27 – Fluxograma da rotina para modular o sinal serial

3.4.2 Amplificador de Potência

Para se transmitir um sinal modulado na rede elétrica é necessário um circuito amplificador potência, pois a rede 127VAC representa uma baixa carga e o sinal sofre uma forte atenuação ao longo do caminho. O circuito projetado acopla uma tensão de 2 Vrms na rede, suficiente para esta aplicação. O circuito utilizado é um amplificador na configuração *push-pull* [8]. A impedância da rede elétrica pode ser estimada como uma carga entre 1 Ω e 10 Ω. O circuito amplificador de potência é mostrado na figura 31.

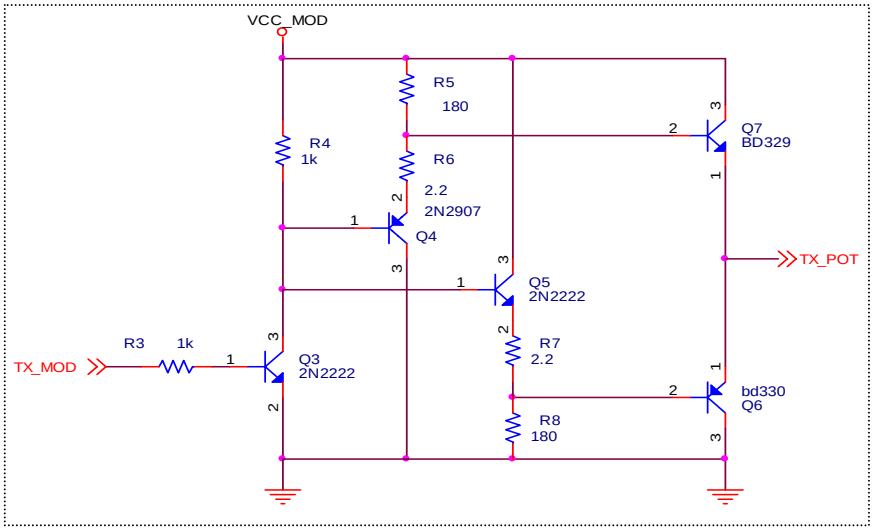


Figura 28 – Amplificador potência push-pull para o circuito transmissor

Nesse circuito o transistor Q3, que opera entre as regiões de corte e saturação, tem a função de converter o sinal serial de 0V a 5V para um sinal de valores próximos de 0V a 12V. O resistor R5 tem a função de ativar Q7 quando Q4 estiver cortado. Isto ocorre, quando Q3 também está desativado, através de um nível baixo na entrada TX_MOD. Para desligar Q7, deve-se saturar Q4 e, ao mesmo tempo, ligar Q6, através Q5, criando um caminho para o terra. Assim, quando a entrada estiver em 0V, Q7 está conduzindo e Q6 está cortado, abrindo caminho para VCC. Quando TX_MOD estiver em 5V, Q7 está cortado e Q6 conduzindo abrindo-se um caminho para o terra.

3.4.3 O acoplador para rede elétrica

O sinal foi acoplado na rede elétrica utilizando um transformador de ferrite. A figura 32 apresenta o circuito do acoplador.

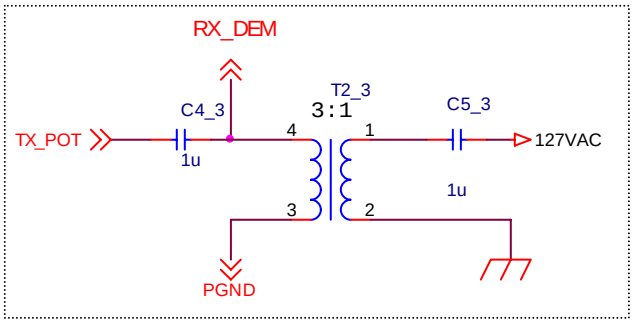


Figura 29 – Esquema do acoplador de sinal para rede elétrica.

Visando aumentar a carga vista pela saída do amplificador de potência, quando refletida pelo acoplador, foi utilizado um transformador 3:1. Nesse caso a impedância equivalente vista pelo primário do transformador é nove vezes maior que a carga.

Entretanto, apesar da exigência de corrente três vezes menor, para garantir 2V rms de sinal na rede é necessário que a saída de potência proporcione uma tensão três vezes maior, ou seja, 6V rms. A tabela 9 permite comparar os valores encontrados na saída do amplificador de potencia, *TX_POT*, para acoplar 2V rms na linha PLC.

\Saída do amplificador	Tensão	Corrente	Impedância	Potencia
transformador 1:1	2 Vrms	200 mA rms	10 Ω	400 mW
transformador 3:1	6 Vrms	66,6 mA rms	90 Ω	400 mW

Tabela 9 – Comparação entre os transformadores 3:1 e 1:1.

A presença do capacitor C5_3 tem o objetivo de impedir a passagem da tensão de 60Hz para os enrolamentos do ferrite. Entretanto, permite a passagem do sinal de comunicação, pois apresenta uma baixa impedância em altas freqüências.

Como este acoplador é utilizado tanto para a recepção de sinal quanto para transmissão, utilizamos um circuito para desligar a alimentação do amplificador quando não se deseja enviar mensagens pela linha PLC. Com isso, as freqüências altas presentes na rede passarão pelo transformador, a saída *TX_POT* será vista como um circuito aberto e o sinal PLC será encaminhado para o circuito de demodulação, representado pela saída *RX_DEM*. A figura 30 mostra o esquema utilizado para desligar a alimentação do estágio de potência.

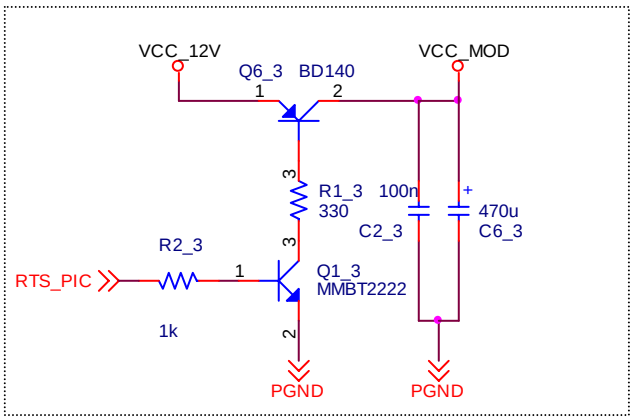


Figura 30 – Circuito para cortar a alimentação do estágio de potência.

Para iniciar a transmissão da informação pela linha PLC, o controlador ativa o sinal o *RTS_PIC*, colocando-o em nível lógico alto, para ligar o transmissor. Em seguida, como existe um intervalo de tempo que o amplificador leva para funcionar corretamente, o PIC transmite dez bytes, de valor 255, para excitar o circuito de potência, antes de enviar a informação relevante. Além disso, esses bytes também são importantes porque silenciam a entrada do receptor, ajudando a sincronizar a chegadas dos bytes.

3.4.4 O receptor FM

O circuito de recepção do sinal FM se baseia em um duplo receptor super-heteródino [9] e utiliza filtros e amplificadores para minimizar ruídos e interferências existentes. A informação é captada através do acoplador para rede elétrica, como visto no final da seção anterior. Em seguida, é utilizado um filtro sintonizado na freqüência da portadora para selecionar o sinal e limitar a energia de ruído. O sinal passa então por um amplificador sintonizado para receber um ganho de tensão na freqüência de interesse, preparando o sinal para entrar no integrado da Motorola MC13135 [10]. Esse circuito utiliza a demodulação FM por dupla conversão, possuindo internamente osciladores locais e células para multiplicação analógica. Finalmente, o sinal passa por um filtro passa baixa para retirar freqüências acima da taxa de comunicação. O diagrama de blocos do sistema receptor pode ser visto na figura 31.

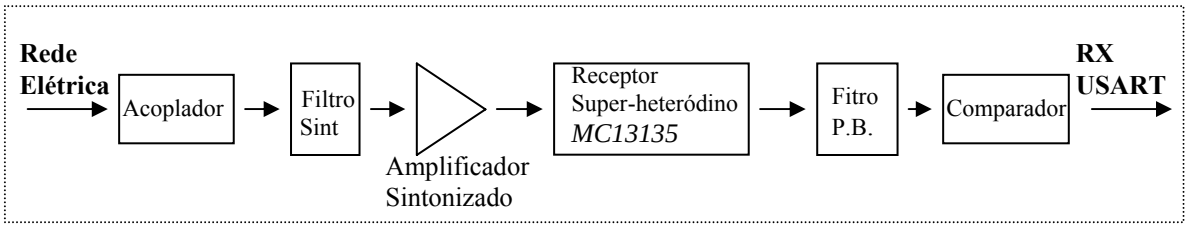


Figura 31– Diagrama de blocos do receptor PLC.

Como a rede elétrica é um meio bastante ruidoso, utilizamos um filtro sintonizado passivo para selecionar apenas a faixa de freqüência utilizada na comunicação. Este filtro é formado por dois ressonadores LC e um capacitor de 10pF. O circuito é mostrado na figura 32.

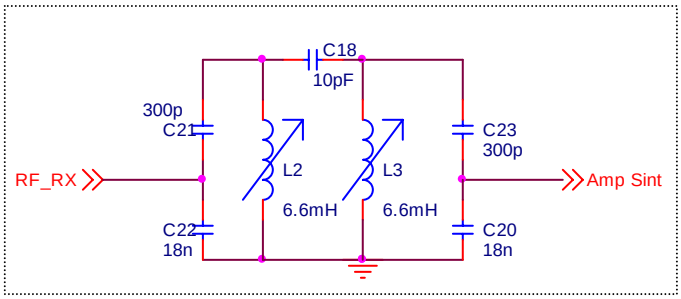


Figura 32 – Filtro passivo sintonizado

Os divisores formados pelos capacitores C21 e C22, e C20 e C23, tem a função de aumentar a impedância de entrada e saída do filtro, respectivamente. Além disso, o filtro também realiza um casamento de impedância, o impede que haja uma grande atenuação do sinal ao passar pelo filtro.

Se dimensionarmos C20 e C22 para serem muito maiores que C21 e C23 podemos considerar que a sintonia do filtro é feita através do tanque LC formado por L2 e C21, e L3 e C23. Os indutores permitem um ajuste fino da sintonia durante o funcionamento do circuito para se obter a máxima sensibilidade na recepção. A sintonia é calculada pela equação 2.

$$\sqrt{\frac{1}{L\,C}} = 2\,\pi\,Fc$$

Equação 2 – Freqüência de ressonância do filtro LC

Quando a sintonia está devidamente ajustada, os ressonadores LC apresentam uma impedância alta na freqüência da portadora permitindo que o sinal passe pelo capacitor C18. As demais freqüências são atenuadas pelo filtro.

Após a filtragem do sinal PLC, é utilizado um amplificador para proporcionar um ganho de tensão ao sinal filtrado. Este amplificador utiliza um transformador de alta freqüência proporcionando uma saída diferencial, *S_Diff*, para entrar no integrado de demodulação.

A freqüência de ressonância é calculada através da equação 2 utilizando o conjunto formado por C19 e a indutância presente no primário do transformador entre os pinos 1 e 3. O esquema do amplificador pode ser visualizado na figura 33.

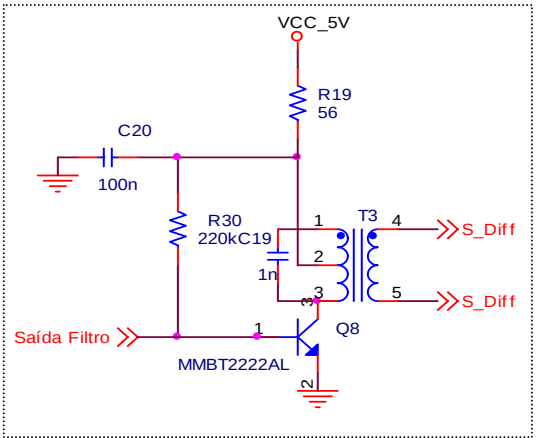


Figura 33 – Circuito amplificador sintonizado

A figura 34 mostra o circuito integrado MC13135, da Motorola.

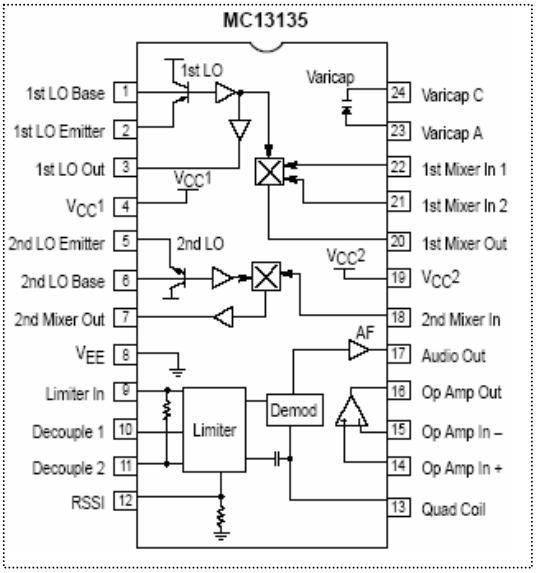


Figura 34 – Circuito integrado de demodulação

O sinal modulado em FSK, que já foi filtrado e amplificado, entra neste circuito pelos pinos 21 e 22. Neste ponto o integrado está preparado receber e multiplicar o sinal por um oscilador local a cristal em 10,813MHz, conectado ao pino 1. O sinal resultante é proporcional à soma e a diferença das duas freqüências, como mostrada na equação 3.

$$\sin(a)\sin(b) = \frac{1}{2}\cos(a-b) - \frac{1}{2}\cos(a+b)$$

Equação 3 – Formula da multiplicação de dois senos.

Em seguida, obtendo um sinal em 10,700MHz (10,813 – 0,113) e um em 10,926MHz (10,813 + 0,113), foi utilizado um filtro cerâmico de 10,7MHz, valor disponível no mercado, conectado aos pinos 20, 19 e 18, para selecionar apenas as

freqüências desejadas. O sinal é encaminhado a um segundo misturador, através do pino 18, que multiplica o sinal por um oscilador local a cristal em 10,245MHz. Valendo-se novamente da equação 3, temos uma resultante em 455kHz (10,700MHz-10,245MHz). O sinal passa por um filtro cerâmico, sintonizado nesta freqüência, presente nos pinos 7, 8 e 9, e é encaminhado para o circuito interno de demodulação. Neste circuito mistura-se o sinal resultante com um ressonador em 455kHz, pino 13, recuperando a banda base. A figura 35 mostra o diagrama de blocos da operação interna do *MC13135*.

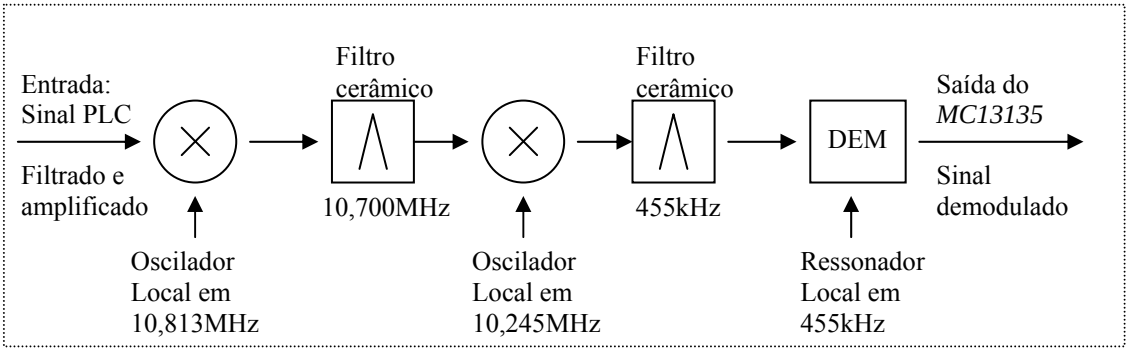


Figura 35 – Diagrama de blocos do demodulador *MC13135*.

Na saída do *MC13135* é conectado um filtro passa-baixa ativo para eliminar as altas freqüências do sinal, com o intuito de melhorar o desempenho do comparador. Este filtro tem a função de retirar do sinal as freqüências acima de 700Hz e o comparador tem o objetivo de converter o sinal analógico em digital, na forma de uma onda quadrada, para que seja compreendido pela porta serial do PIC. Com isso, a saída *RX_DEM* é conectada diretamente ao microcontrolador. A Figura 36 mostra o esquema utilizando os amplificadores operacionais *U2A*, como filtro, e *U2B*, como comparador.

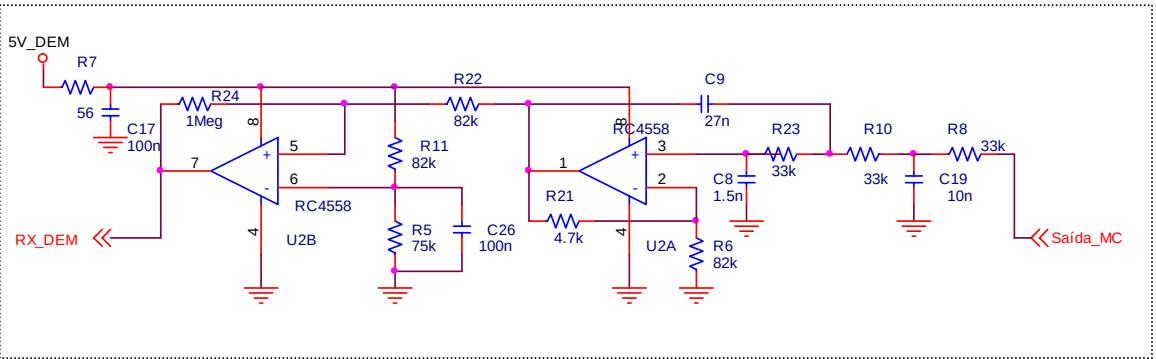


Figura 36 – Filtro ativo passa baixa.

Capítulo 4: Testes e resultados

Para aferir o funcionamento do medidor, deveriam ser realizados testes em bancada com instrumentos que normalmente só são encontrados em laboratórios de homologação e certificação. A dificuldade no acesso a esses centros levou ao desenvolvimento de uma metodologia própria, simplificada, com o objetivo apenas de confirmar as principais especificações: a precisão quanto à medição de energia elétrica e o sucesso na comunicação PLC.

A metodologia empregada está descrita no Apêndice 2.

4.1 Erro na Medição do Consumo

A tabela 10 apresenta os resultados dos testes e mostra a diferença entre os valores teóricos e práticos encontrados para correntes entre 1 e 70 amperes, com a voltagem da rede mantida constante em 127 Vrms.

IL teórico **(A)**	**FCF** teórico **(Hz)**	**FCF** prático **(Hz)**	**Erro** (%)
71,8870	16,229777	16,252234	0,1382
39,1190	8,831814	8,839388	0,0857
21,8810	4,940027	4,945109	0,1028
12,4734	2,816093	2,820938	0,1717
8,6124	1,944403	1,944952	0,0282
3,8934	0,879005	0,878735	-0,0307
2,2032	0,497412	0,498132	0,1446
1,0274	0,231961	0,232288	0,1409

Tabela 10 – Resultados das medidas do consumo de energia

È importante observar que o erro obtido encontra-se dentro da faixa de 0,2 %. A figura 37 mostra o graficamente este resultado.

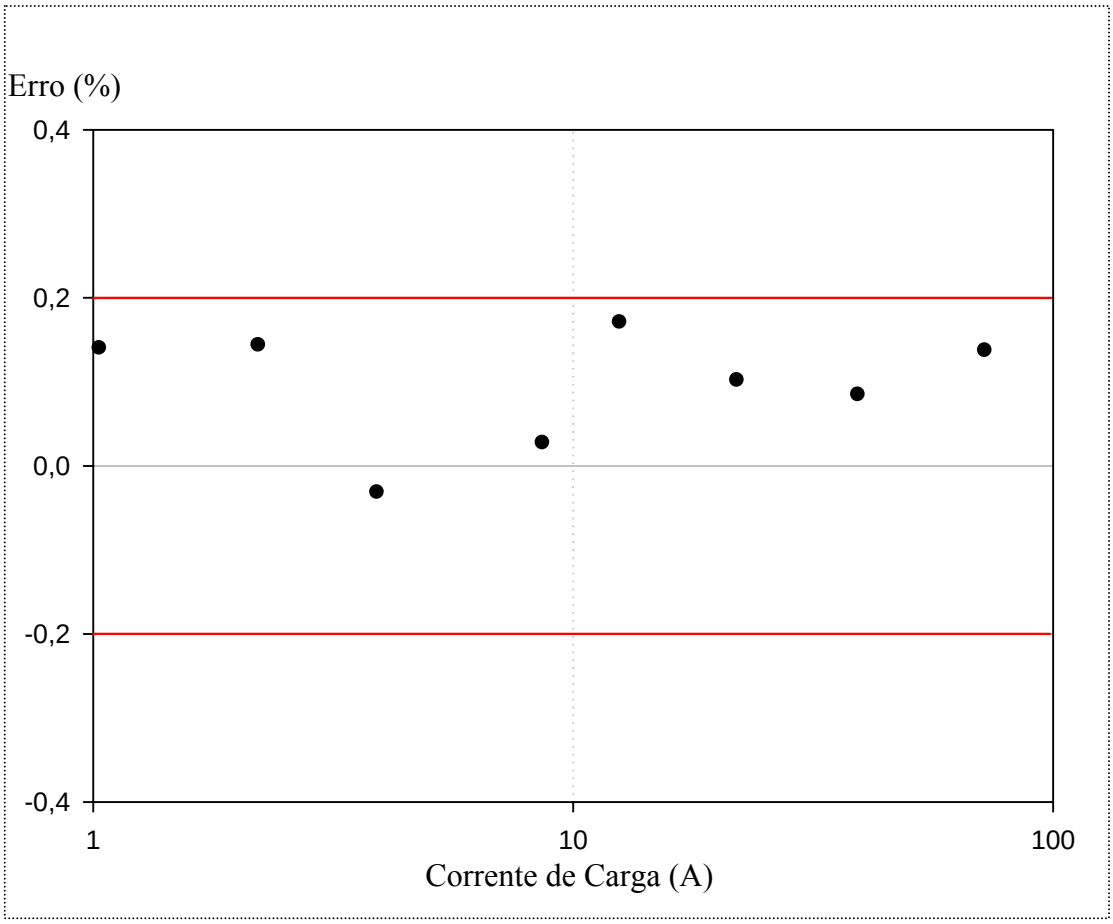


Figura 37 – Erro na medição do consumo de energia em função da corrente.

4.2 Desempenho da Comunicação PLC

Para avaliar o desempenho da comunicação através da rede elétrica foi desenvolvido um método aproximado para medir a taxa de erro de bit (BER), em diferentes níveis do sinal. É importante destacar que a rede elétrica possui um ruído de fundo alto, que influencia o desempenho da comunicação, reduzindo a sensibilidade do receptor.

Os testes foram realizados para vários níveis de potência do sinal na entrada do receptor, em uma condição aproximadamente constante do nível de ruído, em -60dBm. A tabela 11 apresenta o resultado dos testes e mostra a média do número de bits corrompidos em uma transmissão de um milhão de bits.

A figura 38 mostra o gráfico da taxa de erro em função da potência no receptor.

Potência do sinal no receptor (**-** dBm)	**Erro** (bits)
45	0
50	121
55	228
60	9511

Tabela 11 – Resultados da comunicação PLC em 1 milhão de bits enviados

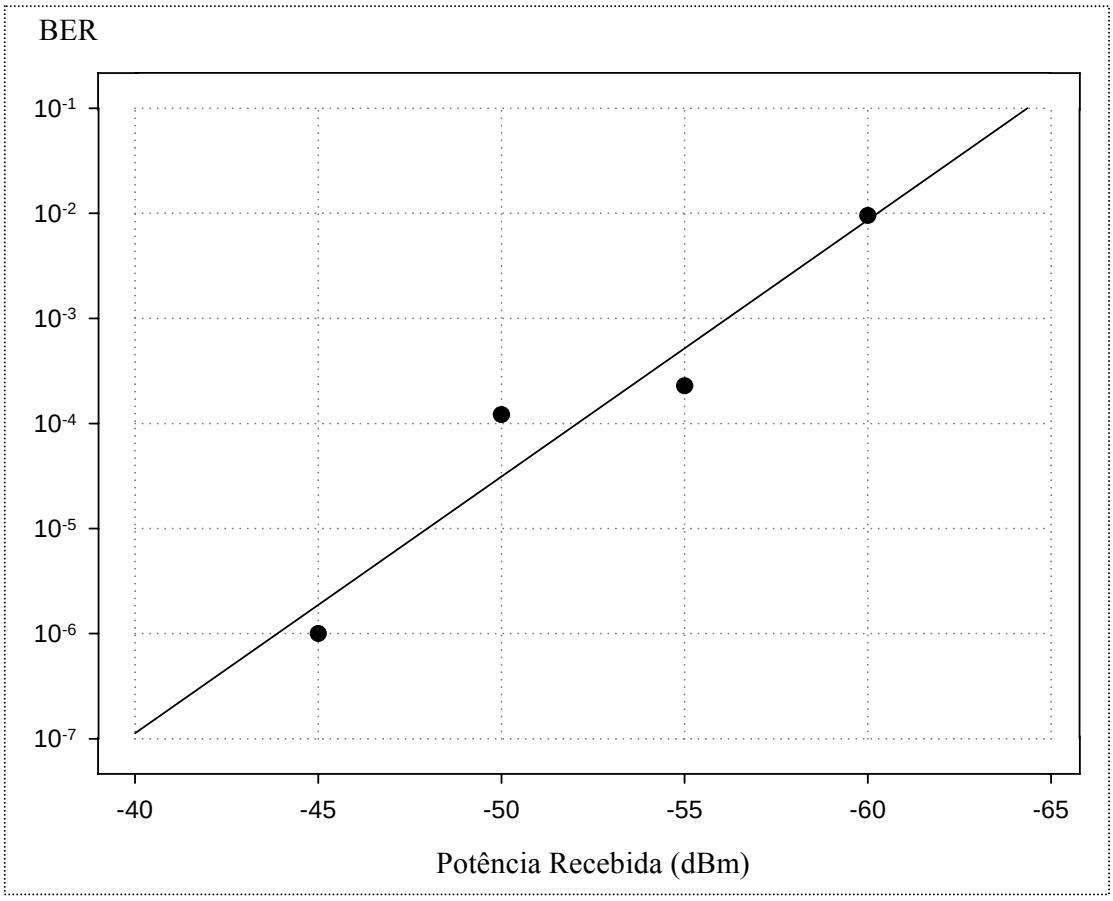


Figura 38 –Taxa de erro em função da potência recebida

Capítulo 5: Conclusões e trabalhos futuros

Os testes, mesmo sendo aproximados, mostraram que o projeto funcionou conforme o esperado. O circuito para medição de energia apresentou um erro menor que 0,2%.

Embora o chip apresente em sua especificação uma acuidade menor que 0,1%, permitindo então ser comparável aos equipamentos já utilizados atualmente, os resultados foram bastante animadores.

Um outro fator relevante para o aumento do erro foi a classe dos instrumentos empregados nos testes, que apresentaram variações superiores ao patamar de 0,1% pretendido para a aferição. Com isso, pode-se atribuir, em maior parte, a diferença entre os valores teóricos e práticos aos instrumentos de teste: a instabilidade do gerador de sinal para simulação da tensão da rede, e as imprecisões do voltímetro e do frequencímetro.

Os testes da comunicação através da rede mostraram, como já era esperado, que o ruído de fundo presente na rede elétrica influencia bastante a informação presente na linha.

Resultados de observação prática, confirmados pelos testes, mostraram que para se obter sucesso deve-se manter um nível de sinal em torno de 15 dBm acima do ruído, junto à unidade receptora. O teste foi realizado em condições de ruído em torno de -60 dBm e os resultados para um nível de sinal no receptor em torno -45 dBm apresentaram BER menor que $1x10^{-6}$.

Assim como os resultados encontram-se dentro das especificações e as placas já possuem um grau elevado de acabamento, pode-se concluir que o protótipo construído já se encontra perto de um produto final. Entretanto, existem trabalhos que poderiam ser realizados para melhorar o produto final, tornando sua utilização mais confiável pela empresa distribuidora.

Devido à inviolabilidade inerente aos medidores de energia e a necessidade de se enviar comandos através de funcionários, pode-se pensar em desenvolver uma interface infra-vermelho (IrDa). Esta solução permite que os comandos, como leitura e corte, possam ser enviados diretamente ao medidor através de instrumentos portáteis. Nesta filosofia, o funcionário poderia consultar o totalizador, reinicia-lo e cortar ou religar o fornecimento no local.

Além disso, a metodologia aqui empregada poderia também ser utilizada para construir um medidor trifásico de energia baseando-se no chip ADE7752 da *Analog Devices* ou similar. O projeto aqui desenvolvido visa cobrir as comunidades de baixa renda nas quais as concessionárias empregam medidores monofásicos. Entretanto, ainda existiriam os clientes trifásicos para se implantar o sistema.

Por fim, pode-se dizer que este projeto contribuiu bastante para a formação de um engenheiro, pois reuniu várias áreas da eletrônica e permitiu o exercício da criatividade ao encontrar soluções de firmware e hardware viabilizando um protótipo de medidor de consumo para AMR.

REFERÊNCIAS

[1] *Analog Devices Products* – ADE7757 datasheet. Disponível em: < http://www.analog.com/UploadedFiles/Data_Sheets/36315634546902ADE7757_a.pdf > acesso em: out 2005

[2] *Analog Devices* – ADE7757 Application Note AN-679. Disponível em: < http://www.analog.com/UploadedFiles/Application_Notes/113759387AN-679_0.pdf >. Acesso em: out 2005.

[3] *Microchip Products - PIC 16F628 datasheet*. Disponível em: < http://ww1.microchip.com/downloads/en/DeviceDoc/40044b.pdf>. Acesso em: out. 2005.

[4] *Microchip Technology Inc*. Disponível em: < http://www.microchip.com>. Acesso em: out. 2005.

[5] *Novus Produtos Eletrônicos LTDA*. Disponível em: < http://www.novus.com.br/site/default.asp>. Acesso em: out 2005.

[6] *Oshon Software – PIC Simulator IDE*. Disponível em: < http://www.oshonsoft.com/>. Acesso em: out. 2005.

[7] *Texas Instruments Products – 74HC4053 datasheet*. Disponível em: < http://focus.ti.com/lit/ds/symlink/cd54hc4053.pdf >. Acesso em: out 2005.

[8] *Pinto, Fernando Baruqui - Apostila de Eletrônica IV: Amplificadores de Potência*. Disponível em: < http://www.del.ufrj.br/~baruqui/Apostila_EletIV.pdf> Acesso em: out. 2005.

[9] *Hakin, Symon - Sistemas de Comunicação*. 4ª. Edição, 2005. Receptor super-heteródino.

[10] *Motorola - MC3362 datasheet.* Disponível em: < http://www.oselectronics.com/downloads/mc3362.pdf>. Acesso em out. 2005.

Apêndice 1 – Cálculos de projeto para o Medidor

Este apêndice contém a descrição dos cálculos da rede de resistores do circuito integrado ADE7757 para medição do consumo de energia. As especificações do projeto encontram-se na tabela 3, ao final da seção 3.1.

I_{MIN} = 70A / 400 = 175mA, Ib = 50 I_{MIN} → *I_B = 8,75A*, corrente de base.

Potencia consumida com I_B: 8,75A * 127V → *Pot = 1111,25W*

Considerando que a constante do totalizador eletromecânico é de 100 imp/kWh, temos:

C = 100 / 3600s = 0,02778 Hz para 1kWh → *C = 0,02778 Hz*

Freq = 1,11125 * 0,02778 Hz = 0,03087 Hz

Tensão no canal V1 (em I_B):

V1 = 8,75A * 300 μΩ = 2,65mV → *V1rms = 2,65mV*

Tensão no canal V2 (em I_B):

Para obter a tensão no canal V2 aplicamos a equação 1 considerando $F_{1\text{-}4}$ = 1,72

$$Freq = \frac{515.84 \times V1_{rms} \times V2_{rms} \times F_{1-4}}{V_{ref}^{\ 2}}$$

Equação 1:

V2 = ((2,5)^2 * 0,03087) / (1,72 * 515,85 * 0,00265) → *V2rms = 82,83mV*

Projeto da rede resistiva:

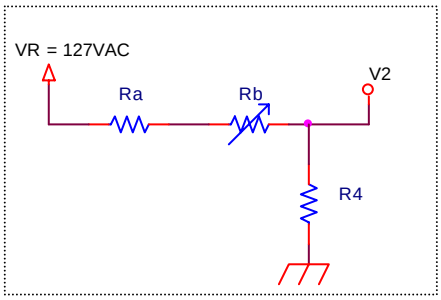


Figura 39 – Projeto da rede resistiva.

Considerando Ra >> R4, podemos admitir que:

1) 127V*R4 / (Ra+Rb) = V1min
2) 127V*R4 / Ra = V1max

Considerando uma calibração de ± 8% para compensar FCF devido imprecisões dos componentes utilizados no projeto.

V1max = 82,83mV + 8% = 89,45mV
V1mim = 82,83mV – 8% = 76,20mV

Então: → *Ra = 5,75 Rb*

Se: Rb = 150kΩ → Ra = 862,5 kΩ

Assim, obtemos a rede resistiva final com os valores da figura 40.

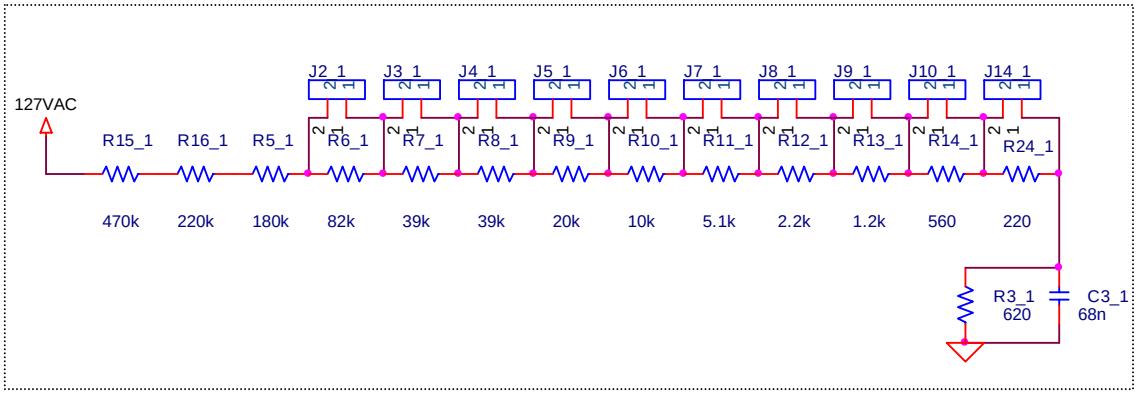


Figura 40 – Circuito final da rede resistiva.

Como os valores utilizados foram aproximados para os que estão disponíveis no mercado, devemos refazer os cálculos para encontrar novamente os valores máximo e mínimo da tensão do canal V1.

Vmin = (127 * 0,62k) / (470k + 220k + 180k + 160,33k + 0,62k) = 76,38mV
Vmax = (127 * 0,62k) / (470k + 220k + 180k + 0,62k) = 90,44mV

Com isso, esta rede deve permitir compensar pequenas variações no valor do shunt, variações em torno da freqüência do oscilador e da tensão de referência interna de 2,5V, que provocam um erro na freqüência de saída FCF. Como mencionado anteriormente, esta rede permite um resolução de 9 bits. Realizando os cálculos, temos:

Vres = (92,44mV – 76,38mV) / 2^9 = 26μV
26μV / 80mV = 0,03 % de ajuste na tensão *V1rms*.

Uma vez estabelecidos os valores da rede resistiva, do resistor shunt e das constantes C e VREF, podemos deduzir uma equação para calcular a freqüência de saída FCF que informa o valor de energia medido pelo ADE7757.

Quando a tensão da rede é 127Vrms, a tensão *V1rms* pode ser calibrada em um valor entre 76,38mV e 90,44mV. Considerando um valor médio de 83,82mV para *V1rms*, podemos encontrar uma relação com *VR*, tal que:

V1rms = V_R (82,83mV / 127V)

E uma relação entre *V2rms* e a corrente *IL* do consumidor. Considerando que o resistor shunt utilizado no caso real é de 300μΩ, temos que:

V2rms = 0,0003 I_L

Substituindo as variáveis V1 e V2 na equação 1, encontramos a equação 4:

$$F_{CF} = 0{,}0017777 \times V_R \times I_L$$

Equação 4 – Equação final do medidor de energia

Apêndice 2 – Descrição dos procedimentos de teste

Calibração do medidor

Conforme visto no capítulo 3, o medidor utiliza um resistor *shunt* de Manganina de 300μΩ para medir a corrente e uma rede atenuadora de resistores para medir a tensão da rede. Com esta informação, o integrado ADE7757 é capaz de calcular a energia, apresentando este informação em forma de freqüência no pino *CF*. A figura 41 mostra a separação entre estes dois circuitos que realizam a amostragem de tensão e corrente.

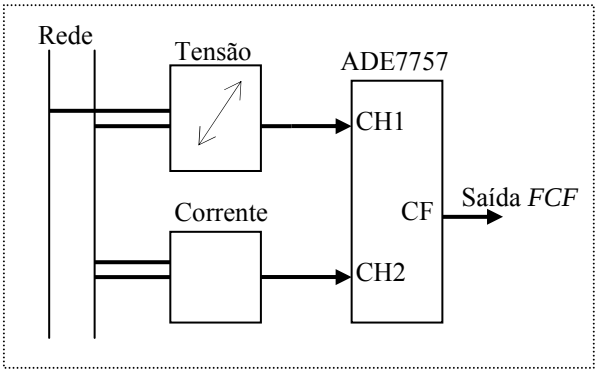


Figura 41 – Funcionamento do medidor ADE7757

Além disso, a rede de resistores permite a calibração do valor de F_{CF} em torno da freqüência de projeto para compensar diferenças entre os valores de projeto e práticos. Uma vez definido o método de calibração, deve-se utilizar um valor próximo ao da corrente base para efetuar a correção.

Método de calibração

Como a tensão da rede é instável, oscilando em torno de 127V, não é possível validar o circuito utilizando esta tensão como um valor de referência. Assim, foi pensado utilizar um equipamento gerador de sinal, disponível no laboratório LASPI, fornecendo uma tensão AC, estável em 60Hz, para simular a rede elétrica. Como o gerador possui uma tensão máxima de 7 Vrms, foi necessário injetar sua saída em um ponto intermediário da rede de resistores, sendo este procedimento equivalente a conectar 127V no seu início. Dessa forma, deve-se calcular a tensão V_{RS} de simulação,

necessária para apresentar a mesma tensão de saída da situação real. A figura 42 mostra o ponto da rede onde foi inserido o sinal do gerador e a saída para o medidor.

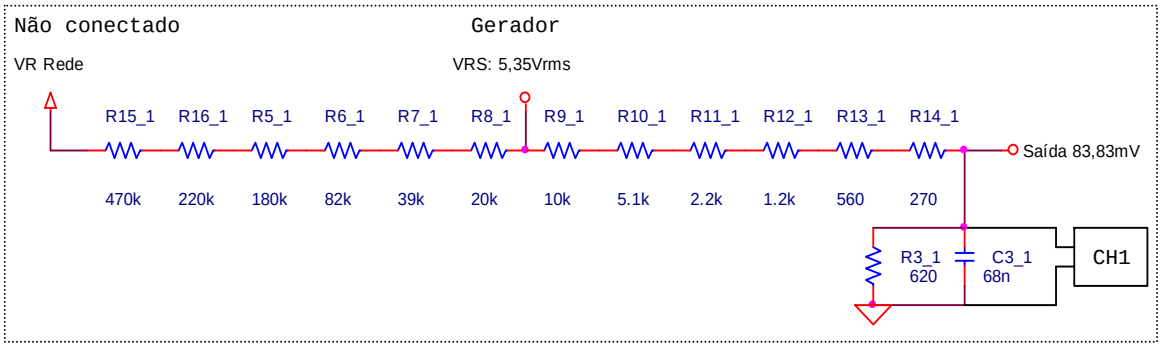


Figura 42- Circuito para simulação da tensão da rede

Devido à dificuldade de gerar correntes altas e estáveis e encontrar um resistor shunt de 300μΩ, foi pensado aumentar o valor do resistor shunt para simular uma corrente alta e utilizar baixas cargas. Nessa filosofia é calculado um valor de resistência próximo a dezenas de ohms que permita gerar a mesma diferença de potencial sobre shunt do caso real, quando passam correntes entre 1 e 70 amperes. Com isso, através de correntes da ordem de micro amperes, podemos simular a situação real de trabalho do medidor. A figura 43 mostra o circuito divisor resistivo utilizado para simular o caso real.

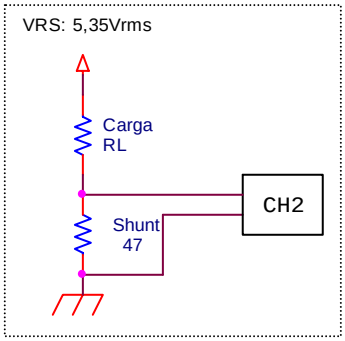


Figura 43 – circuito para simulação de corrente

Projeto

Inicialmente deve-se calcular a tensão de saída do gerador, V_{RS}, para ser inserida na rede resistiva conforme a figura 41.

V_{RS} = (0,62k * 82,83mV) / (10k+5.1k+2.2k+1.2k+0,56k+0,27k+0,62k)

***V_{RS}* = 5,35 Vrms**

Uma vez definida esta tensão, deve-se estipular qual a corrente máxima que se poderá fornecer para a carga. Dessa forma, como a impedância de saída do gerador é de 600Ω, a máxima carga para simulação foi sugerida em um valor de 12kΩ. Isso ocorre para evitar que a tensão de saída do gerador seja atenuada de forma que não se possa ajustá-la. Assim, deve-se calcular o valor do resistor shunt para simulação, *Rs*.

Sabendo que o valor de projeto para corrente máxima é de 70 amperes e 300μΩ para o valor do shunt, considerando 12kΩ >> Rs, temos que:

300μΩ * 70A = *Rs* * (5,35Vrms/ 12kΩ)

***Rs* = 47 Ω**

I_{Lreal} = (*Rs* * 5,35Vrms) / (300μΩ * (*RL*+47Ω))

Com isso, devem-se utilizar diferentes cargas para simular correntes entre 1 e 70A e ajustar o gerador para estabilizar em 5,35Vrms, registrando a freqüência de saída F_{CF} para as diferentes situações. Para aproximar ao máximo os resultados práticos e teóricos os resistores utilizados foram aferidos em equipamento tipo ponte RLC disponível no laboratório LASPI. A tabela 12 mostra os valores de resistência obtidos: as correntes teóricas equivalentes para o caso real, I_L *teórico*, e a freqüência teórica de saída em *CF*.

RL aferido **(k)**	**IL** teórico **(A)**
11,777	71,8870
21,682	39,1190
38,800	21,8810
68,101	12,4734
98,652	8,6124
218,280	3,8934
385,780	2,2032
827,300	1,0274
Resistor shunt utilizado **Rs** = 47,666	

Tabela 12 – Valores dos resistores utilizados para as correntes teóricas.

Procedimento de teste

Para realizar o procedimento de teste foi utilizado um multímetro de 4,5 dígitos para manualmente fixar a tensão de saída do gerador em 5,350Vrms. Em seguida foi utilizado um frequencímetro que possibilitou medir o período em milisegundos com uma precisão de duas casas decimais. A tabela 13, também apresentada no capítulo 4, mostra a comparação entre os valores práticos e teóricos e o erro devido à diferença entre esses valores. A linha em destaque indica o ponto um que foi realizada à calibração.

IL teórico **(A)**	**FCF** teórico **(Hz)**	**FCF** prático **(Hz)**	**Erro** (%)
71,8870	16,229777	16,252234	0,1382
39,1190	8,831814	8,839388	0,0857
21,8810	4,940027	4,945109	0,1028
12,4734	2,816093	2,820938	0,1717
8,6124	1,944403	1,944952	0,0282
3,8934	0,879005	0,878735	-0,0307
2,2032	0,497412	0,498132	0,1446
1,0274	0,231961	0,232288	0,1409

Tabela 13 – Comparação entre valores práticos e teóricos

Teste da taxa de erro de bit (BER) na rede PLC

Visando avaliar a comunicação através da linha PLC, foi utilizada a taxa de erro de bit (*BER*) como fator de qualidade da transmissão da informação. A *BER* é um índice que mede a quantidade de bits errados entregues ao receptor. Os testes foram realizados na rede local do laboratório LASPI/UFRJ.

Utilizando as placas *Conversor* e *Medidor*, foi necessário desenvolver dois programas: um dedicado a transmitir uma seqüência fixa de bits e o outro responsável por contar o número de bits errados recebidos.

Fimware transmissor

Para testar a comunicação foi definido o byte 0x55 (hexadecimal) para ser enviado pelo transmissor. Este caractere foi escolhido por produzir uma onda quadrada contínua modulada na freqüência de comunicação, sendo seu valor binário "01010101".

O programa é iniciado configurando os registradores da USART e em seguida chama a função de enviar e modular o byte, já criada no projeto, realizando esta tarefa enquanto estiver ligado. A figura 44 mostra o fluxograma.

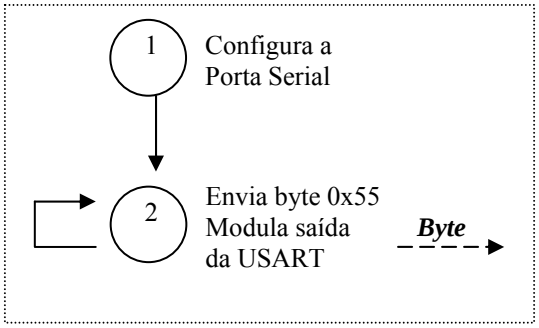


Figura 44 – Fluxograma do firmware para transmissão de bytes 0x55

Firmware receptor

O programa medidor da BER configura a comunicação serial na taxa de dados correta e em seguida entra em uma rotina que um que aguarda receber um byte pelo circuito receptor. Em seguida, ao receber um caractere, o firmware incrementa o contador de bits em dez unidades, pois um byte transmitido assincronamente possui dois

bits de sinalização, um para início e um fim de bloco, e, em seguida, entra na rotina de comparação.

Ao realizar a comparação, o PIC verifica se o byte recebido confere com o esperado. Caso este byte esteja errado, primeiramente o controlador verifica se houve erro de frame. Caso tenha ocorrido um erro no frame, então o PIC incrementa o contador da BER em duas unidades. Quando o frame do byte recebido está correto, mas ocorreu erro nos bits, é possível contar o número de bits errados e incrementar no contador. Após a comparação o programa verifica se já chegou a contagem de um milhão de bits e volta a rotina recepção caso ainda não tenha alcançado este valor. Ao final, o PIC grava os valores encontrados em memória EEPROM. È utilizado então o leitor para ler a memória e verificar os valores encontrados. A figura 45 mostra o fluxograma para calcular a BER.

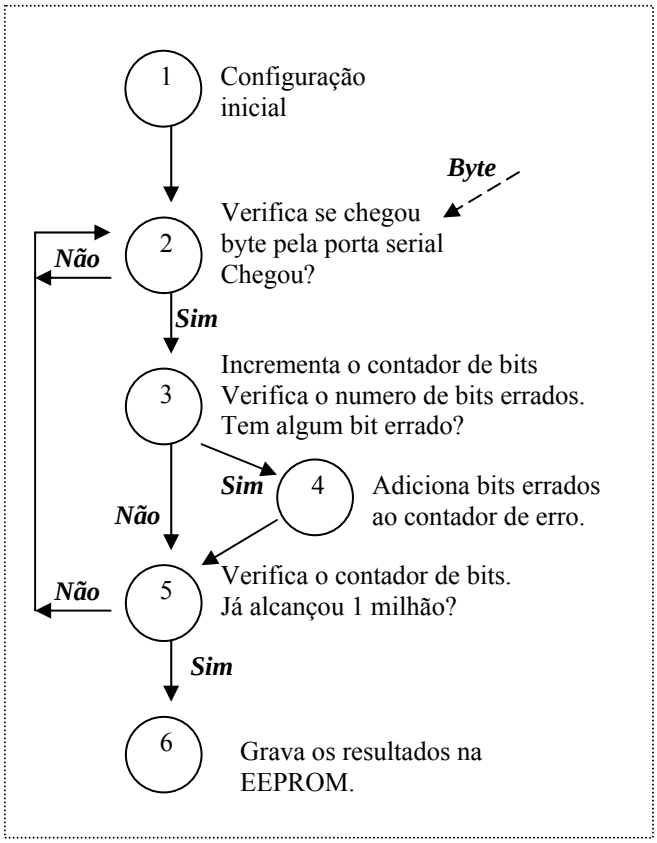


Figura 45 – Fluxograma do firmware para calcular a BER

Resultados da BER

É importante mencionar que a rede elétrica é um canal muito ruidoso e, por este motivo, apesar de se acoplar uma potencia significativa na rede, o sinal transmissão pode ser corrompido pelo o ruído de fundo. Assim, foi utilizado um analisador de espectro, disponível o laboratório LASPI, para caracterizar o ruído presente na linha.

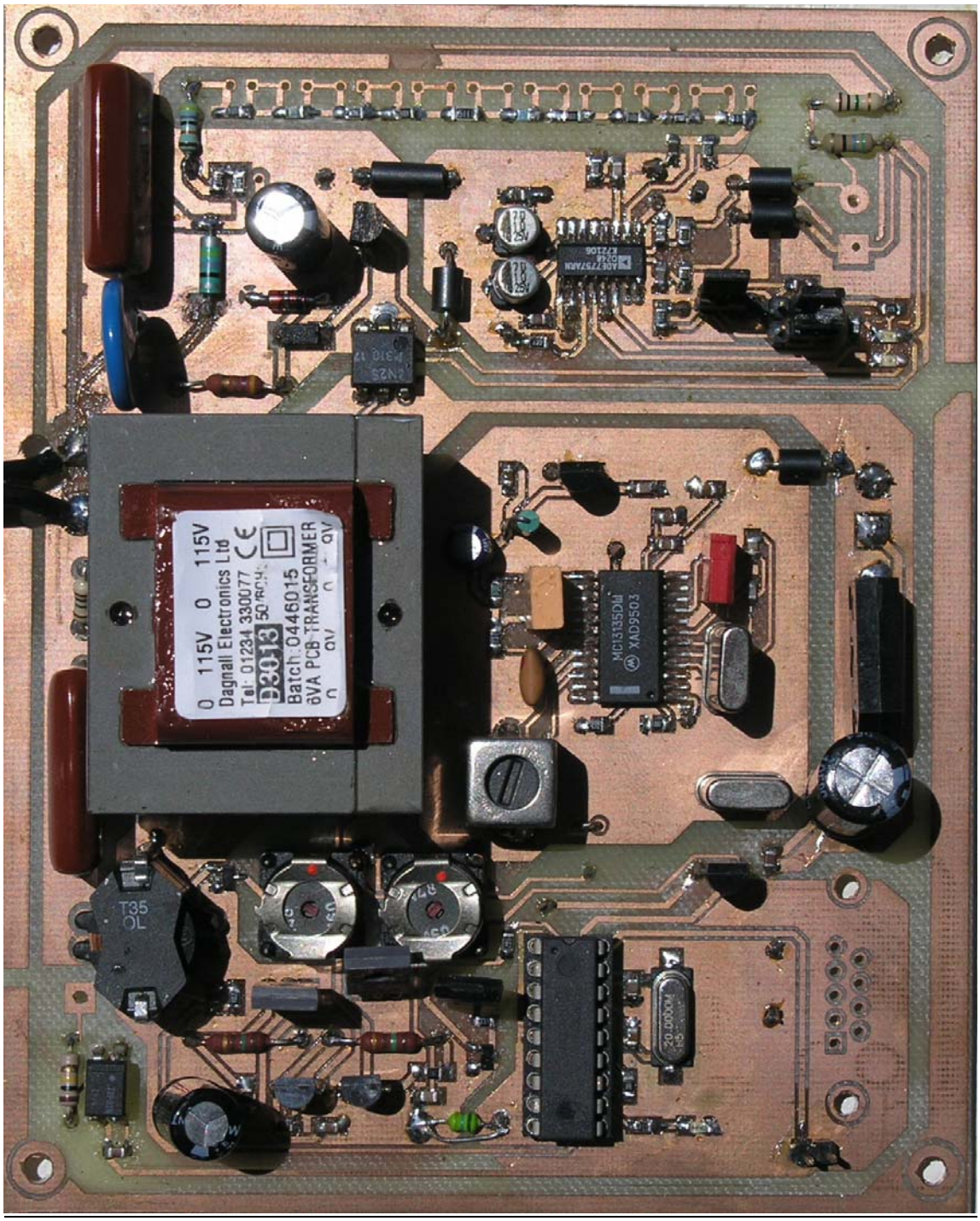
Dessa forma, foram realizados vários testes para diferentes potencias no transmissor, mantendo-se o nível do ruído aproximadamente constante em -60dBm. A tabela 14 mostra a média do número de bits errados recebidos para diferentes níveis de potencia no receptor, ao se transmite 1 milhão de bits através da linha PLC.

Potencia (**-** dBm)	**Erro** (bits em 10^6)
45	0
50	121
55	228
60	9511

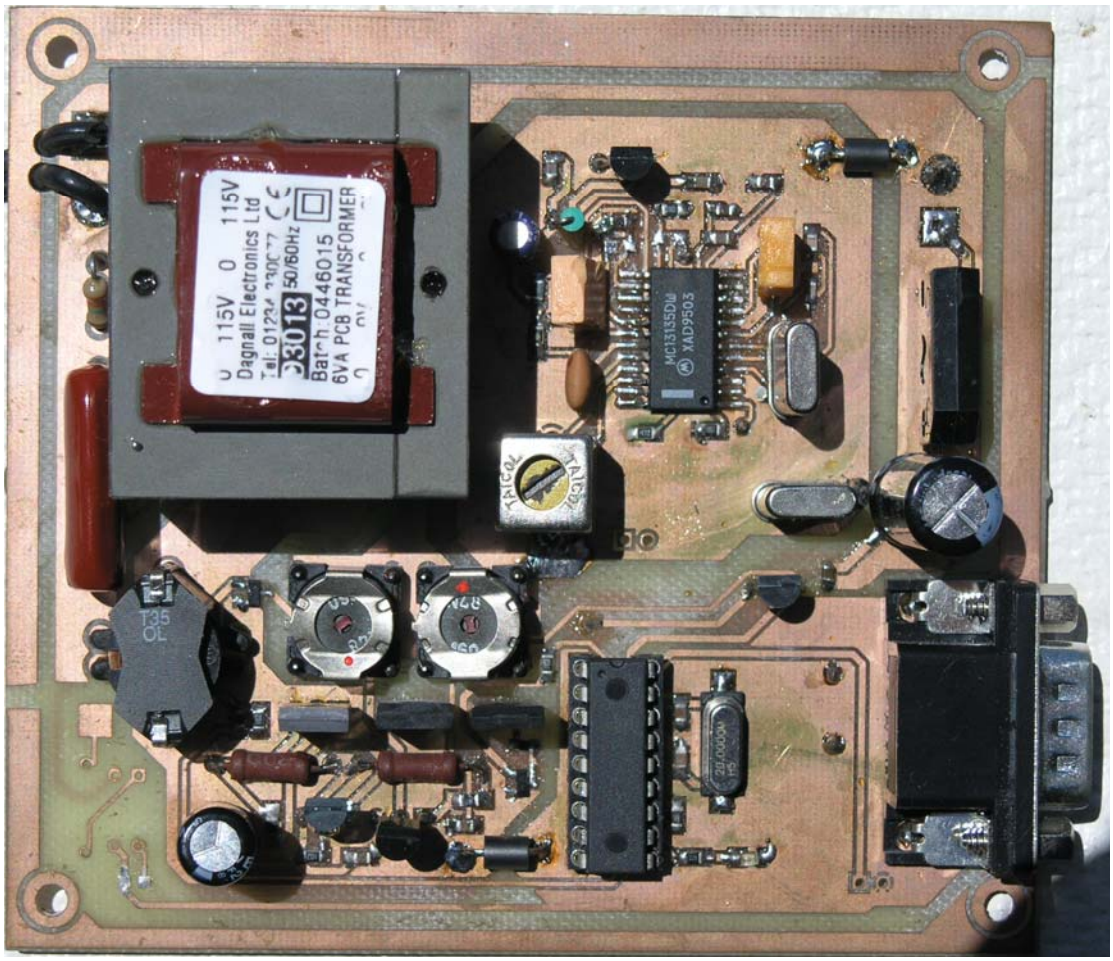
Tabela 14 – Erro de bits em 1 milhão para diferentes potências.

Anexo 1 – Foto das placas desenvolvidas

Placa Medidor de Energia Elétrica

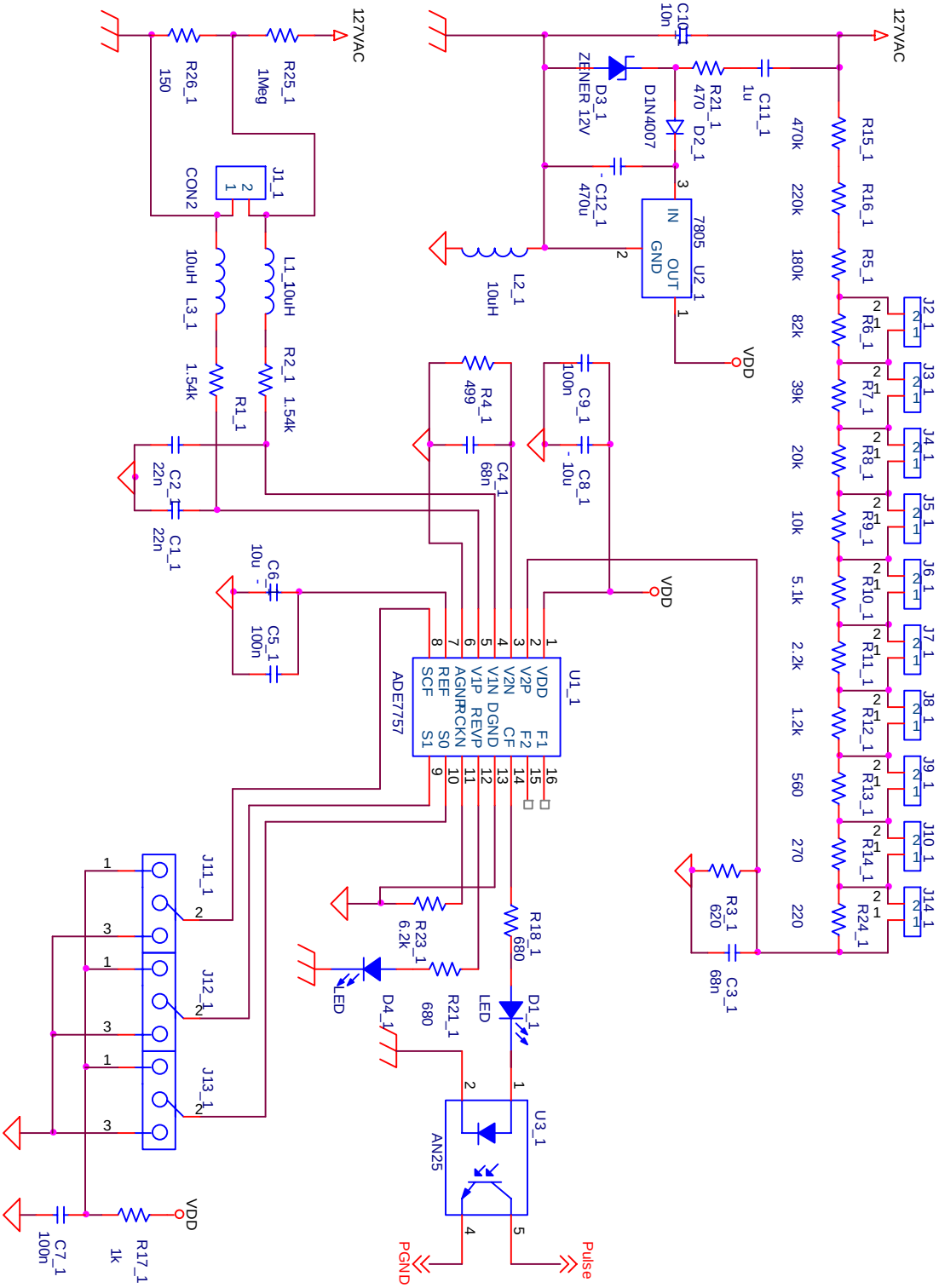


Placa Conversor RS-232 PLC

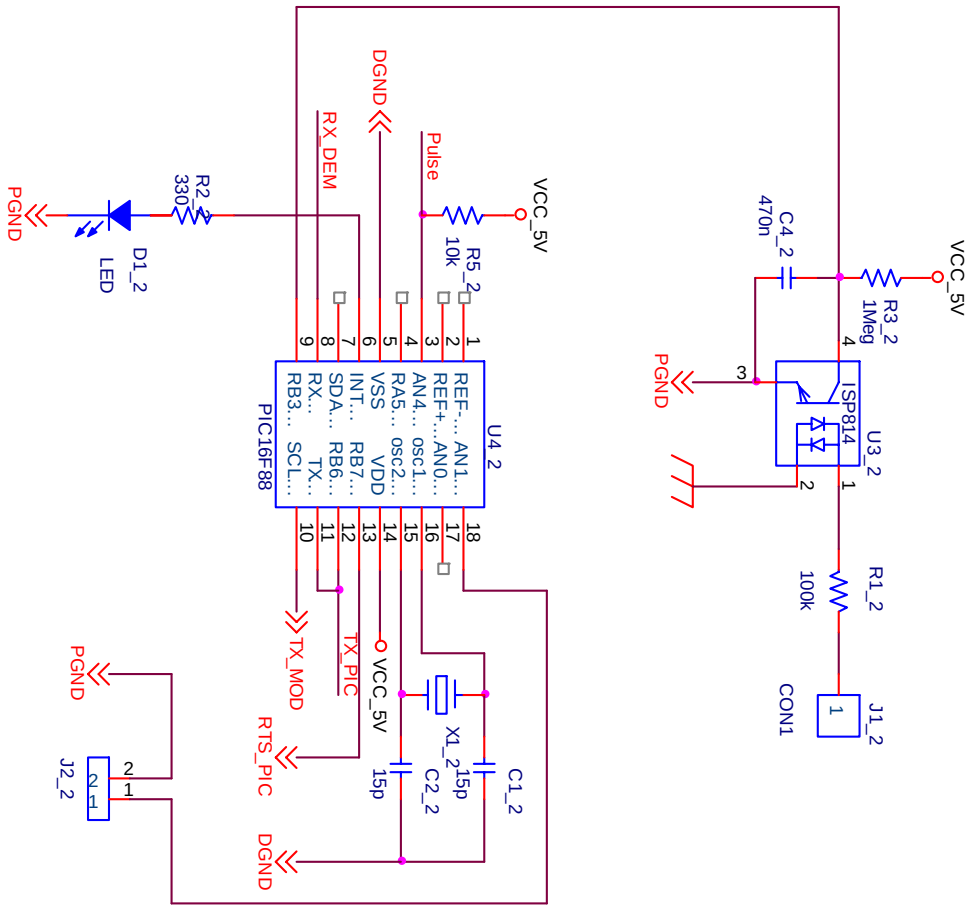


Anexo 2 – Esquemático das placas desenvolvidas

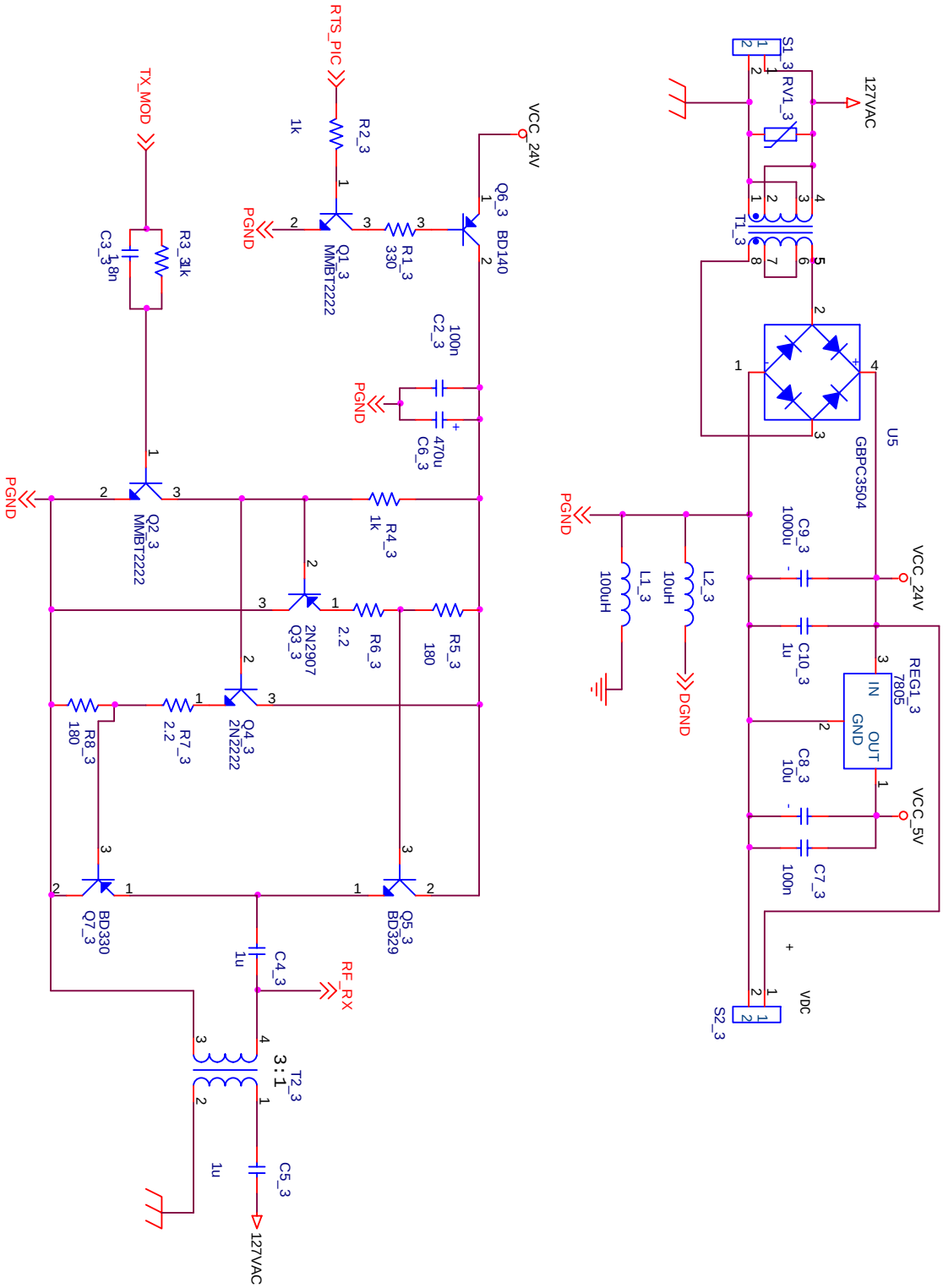
Placa Medidor de Energia Elétrica – esquema 1: Medidor de ADE7757



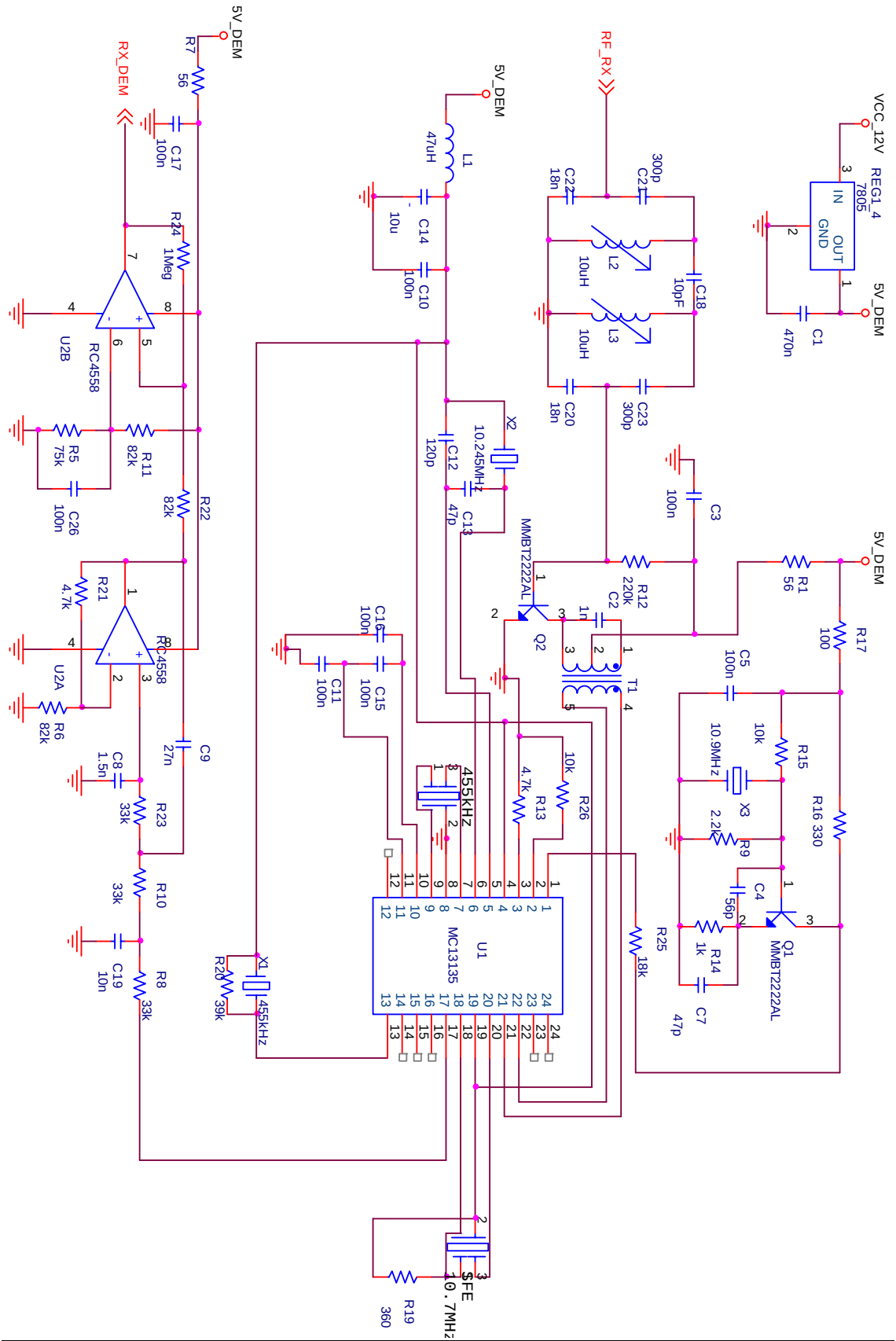
Placa Medidor de Energia Elétrica – esquema 2: Microcontrolador PIC



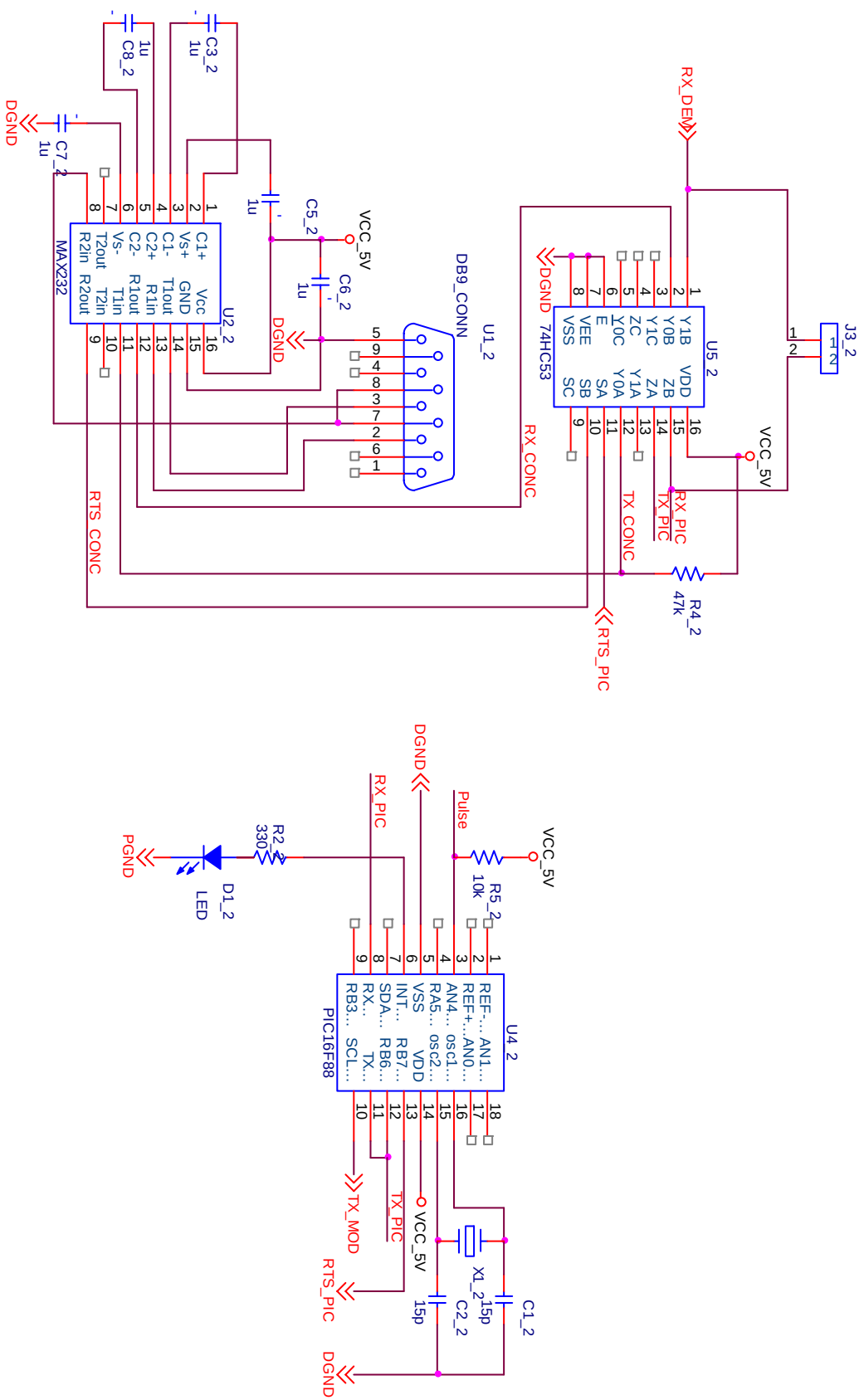
Placa Medidor de Energia Elétrica – esquema 3: Estágio de Potência e fonte



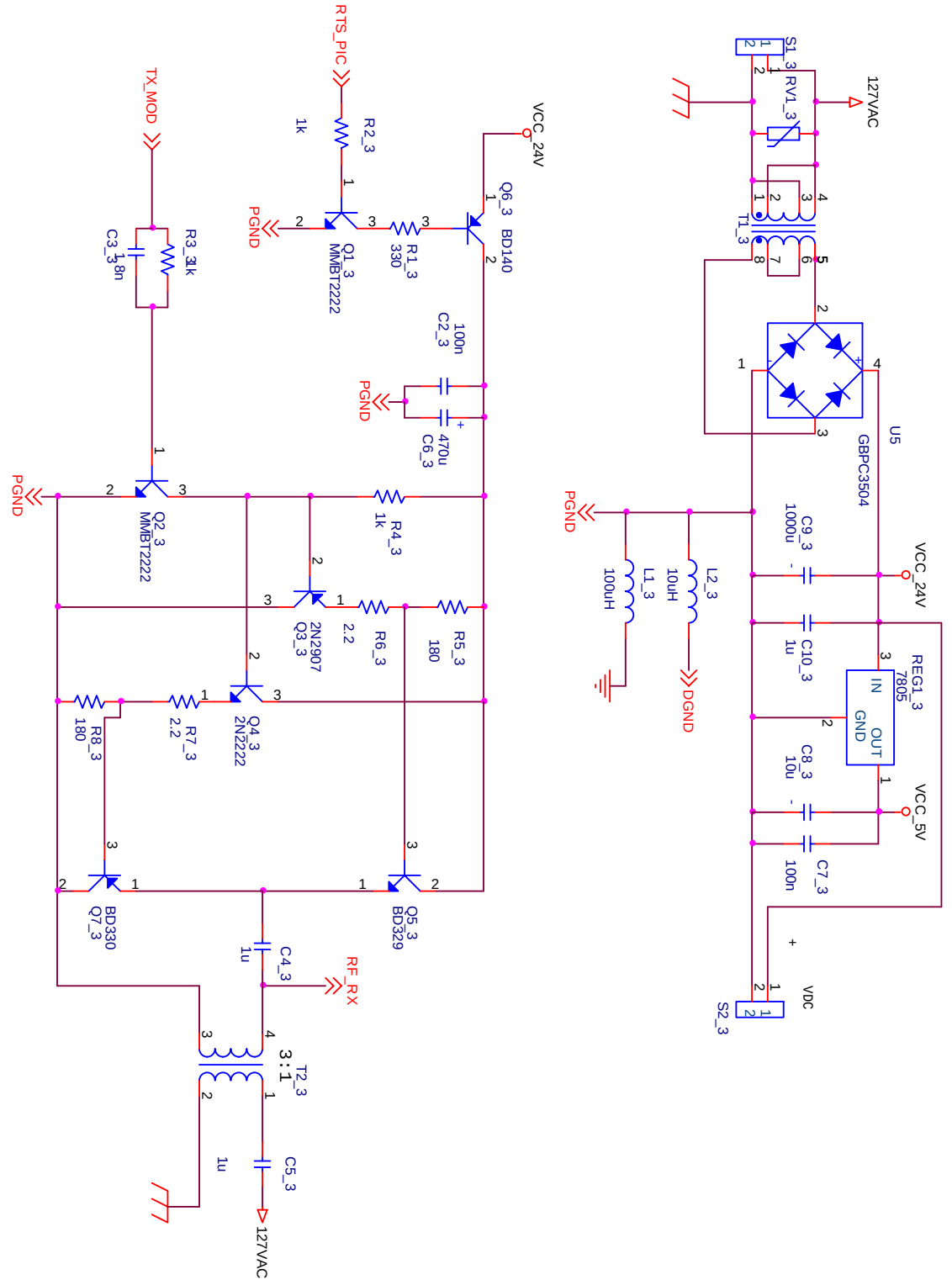
Placa Medidor de Energia Elétrica – esquema 4: Demodulador



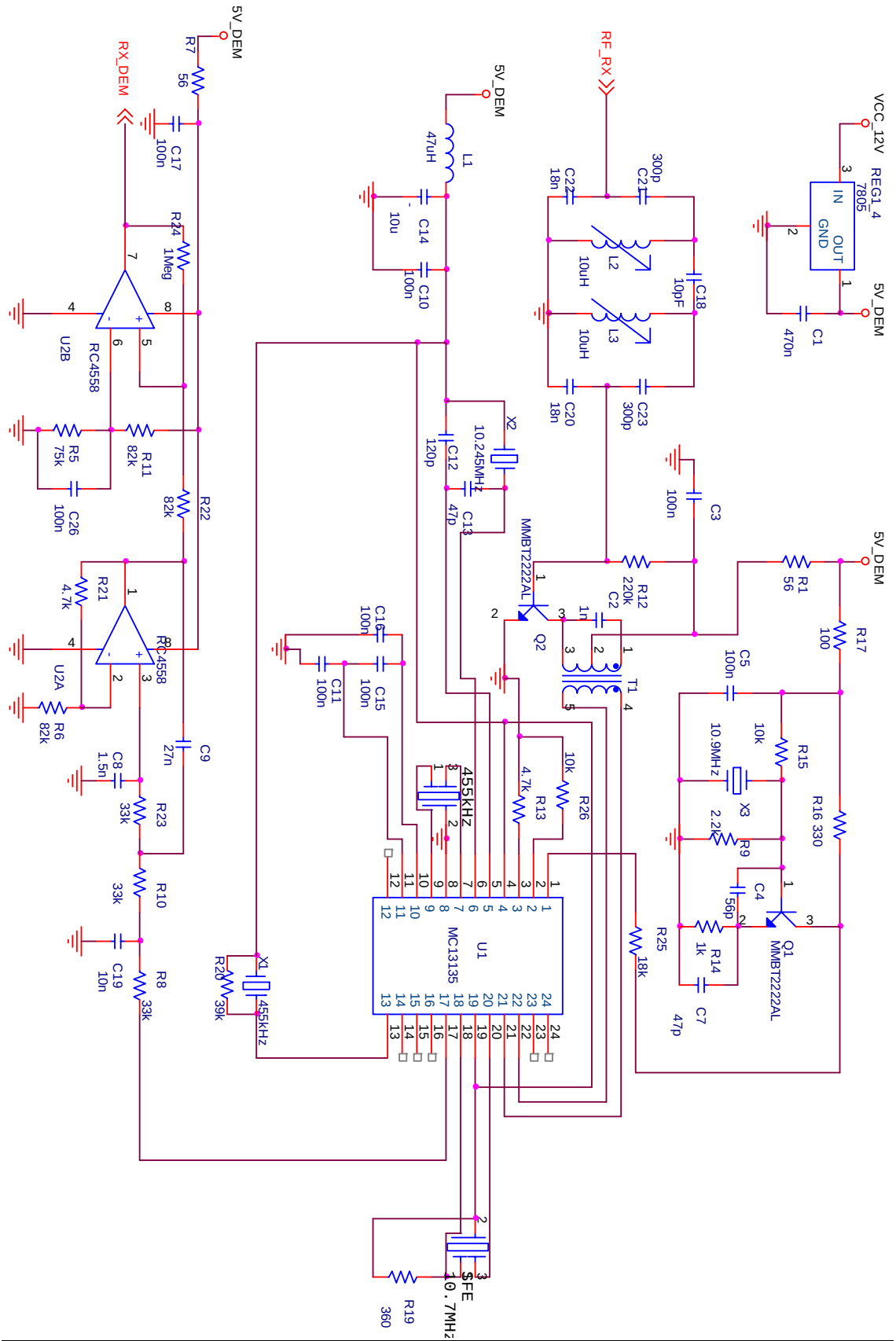
Placa Conversor RS-232 PLC – esquema 1: Micorcontrolador e interface RS-232



Placa Conversor RS-232 PLC – esquema 2: Transmissor e fonte



Placa Conversor RS-232 PLC – esquema 3: Demodulador



Anexo 3 – Códigos-fonte dos programas desenvolvidos

Programa Medidor

```
;===============================================================================

;       FEDERAL UNIVERSITY OF RIO DE JANEIRO
;       ENGINEERING SCHOOL


;       PROJETO:        Projeto final - Medidor Eletrônico de energia com PLC
;       FIRMWARE:       medidor_PF


;       AUTOR:          Paulo Gentil Gibson Fernandes
;       ORIENTADOR:     Carlos José Ribas D'Ávila

;===============================================================================

;       OBSERVAÇÕES:

;       Programa desenvolvido para ser executado em microcontroladores tipo PIC16F88.
;       Este código implementa a inteligência de um medidor eletrônico.
;       Ao receber pulsos invertidos em sua porta RA4/TOCKI incrementa o Timer 0 a cada 3200 pulsos.
;       O firmware se comunica ao concentrador à uma taxa de 1200 bps.


;------Declarando as variáveis globais ----------------------------------------

        COUNTER                 EQU 0x30
        CODRET1                 EQU 0x31
        COUNTER2                EQU 0x32
        COUNTER3                EQU 0x33
        TOTAL_1                 EQU 0x34
        TOTAL_2                 EQU 0x35
        TOTAL_3                 EQU 0x36
        TOTAL_4                 EQU 0x37
        SAVE_W                  EQU 0x38
        SAVE_STATUS             EQU 0x39
        MACK_1                  EQU 0x3A
        MACK_2                  EQU 0x3B
        DATA_EE                 EQU 0x3C
        DATA                    EQU 0x3D
        DATARET                 EQU 0x3E
        EOT                     EQU 0x3F
        PREV_TMR0               EQU 0x40
        STATE                   EQU 0x42
        COUNTER_MOD             EQU 0x43
        FLAG_MOD                EQU 0x44
        COUNTER4                EQU 0x45
        CONF_COUNTER            EQU 0x46
        COUNTER5                EQU 0x47
        COUNTER6                EQU 0x48
        SEND_TMR0               EQU 0x49
        SEND_TOTAL_1            EQU 0x4A
        SEND_TOTAL_2            EQU 0x4B
        SEND_TOTAL_3            EQU 0x4C
        SEND_TOTAL_4            EQU 0x4D

;------Declarando as constantes -----------------------------------------------
```

```
        END_EE_TMR0             EQU 0x00
        END_EE_TOTAL_1          EQU 0x01
        END_EE_TOTAL_2          EQU 0x02
        END_EE_TOTAL_3          EQU 0x03
        END_EE_TOTAL_4          EQU 0x04
        END_EE_MACK_1           EQU 0x08
        END_EE_MACK_2           EQU 0x09
        END_EE_STATE            EQU 0x0A        ;bit0 --> ligado ou desligado fornecimento
                                                ;bit1 --> ligado ou desligado fase

        CHAR_A                  EQU 65
        CHAR_C                  EQU 67
        CHAR_E                  EQU 69
        CHAR_K                  EQU 75
        CHAR_L                  EQU 76
        CHAR_O                  EQU 79
        CHAR_R                  EQU 82
        CHAR_T                  EQU 84
        CHAR_!                  EQU 33
        CHAR_?                  EQU 63
        CHAR_@                  EQU 64
        CHAR_U                  EQU 255
;-------------------------------------------------------------------------------
        ORG     0x0000
        GOTO    MAIN_PROG

        ORG     0x0004
        GOTO    INTERRUPT

;----------------------function to configure oscilator ----------------------------------
OSC:
        BSF             STATUS, RP0
        BSF             OSCCON, 6
        BSF             OSCCON, 5
        BSF             OSCCON, 4
        BSF             OSCCON, 3
        BSF             OSCCON, 2
        BCF             STATUS, RP0
        RETURN

;---------Write E2PROM routine-----------------------------------------------------------
;OBS: EEADR and EEDATA must be defined before called WRITE_EE
;design for PIC16F88, uses EE_ADR and EE_DATA

WRITE_EE:
        BCF             INTCON, GIE
        BSF             STATUS, RP1
        MOVWF           EEADR
        BCF             STATUS, RP1
        MOVF            DATA_EE, W
        BSF             STATUS, RP1
        MOVWF           EEDATA

        BSF             STATUS, RP1         ;select bank 2
        BSF             STATUS, RP0         ;select bank 3
        BCF             EECON1, EEPGD       ;point to data memory
        BSF             EECON1, WREN        ;enables writes

gie_EE:
        BCF             INTCON, GIE         ;Desable all interrupts in a security way and to perform
        BTFSC           INTCON, GIE         ; a savety communication
        GOTO            gie_EE
        MOVLW           0x55
        MOVWF           EECON2
        MOVLW           0xAA
        MOVWF           EECON2
        BSF             EECON1, WR          ;Begin Write
```

```
        BCF             STATUS, RP0
        BCF             STATUS, RP1
Loop_WRT:
        BTFSS           PIR2, EEIF          ;EEIF is set when finish the write process
        GOTO            LOOP_WRT
        BCF             PIR2, EEIF          ;Must be cleared by software

        BSF             INTCON, GIE
        BCF             EECON1, WREN        ;disable writes
        RETURN

;-----------Read E2PROM routine-------------------------------------------------------
;Receive address in W reg and return data in W reg.
;design for PIC16F88

READ_EE:
        BSF             STATUS, RP1         ;select bank 2
        MOVWF           EEADR
        BSF             STATUS,     RP0     ;select bank 3
        BCF             EECON1, EEPGD
        BSF             EECON1, RD
        BCF             STATUS, RP0
        MOVF            EEDATA, W
        BCF             STATUS, RP1
        RETURN
;-----Function to send data from W register-----------------------------------------
TXFUNC2:
        BCF             RCSTA, CREN ;DESABLE RECEIVE MODE

                                    ;If TXIF = '0', transmit buffer is full
        BTFSS           PIR1,TXIF
        GOTO            TXFUNC

                                    ;If transmit buffer is empty,
                                    ;buffer is ready to load new data to transmit.
        MOVWF           TXREG

        ;If TRMT = '0', transmit buffer is full
        BSF             STATUS, RP0
Y1:     BTFSS           TXSTA,TRMT
        GOTO            Y1
        BCF             STATUS, RP0

        BSF             RCSTA, CREN
        RETURN
;-----Function MODULATE DATA-------------------------------------------------------
TXFUNC:
        BCF             RCSTA, 4            ;DESeabilita a recepção serial
        BTFSS           PIR1,TXIF           ;If TXIF = '0', transmit buffer is full
        GOTO            TXFUNC
        MOVWF           TXREG
MAIN:
        BSF             STATUS, RP0
        BTFSC           TXSTA, TRMT
        GOTO            END_TXFUNC
        BCF             STATUS, RP0

        MOVLW           3                   ; alterar o contador em 1 unidade o contador
                                            ; reflete em 3 ciclos de clock
        MOVWF           COUNTER_MOD
        INCF            FLAG_MOD, F
LOOP:
        DECFSZ          COUNTER_MOD, F
        GOTO            LOOP
        ;NOP                                ;incrementar NOPS para aumentar o periodo do ciclo
        BTFSC           FLAG_MOD, 0         ;inferior e superior: 1 NOP = 2 cilcos de clk de atraso
        GOTO            SET_HI
```

```
SET_LOW:
        BCF             PORTB, 4
        BTFSS           PORTB, 6        ;para inverter a modulação trocar BTFSC para BTFSS, e vice-vesa
        GOTO            MAIN
        NOP
        GOTO            MAIN

SET_HI:
        BSF             PORTB, 4
        NOP
        ;NOP                                    ;incrementar NOPs para aumentar o semiperiodo inferior
                                                ; 1 NOP = 1 ciclo de clk de atraso
        GOTO            MAIN

END_TXFUNC:
        BCF             STATUS, RP0
        BSF             RCSTA, 4                ;reabilita a recepção serial
        RETURN

;-----Function to receive data and return data in W reg-------------------------
RCFUNC_0:
        CLRF            CODRET1

        BTFSC           PIR1,RCIF               ;If RCIF = '0', receive buffer is empty
        GOTO            GETRC_1

        MOVLW           140
        ;MOVLW          3
        MOVWF           COUNTER3

WAIT_3:
        MOVLW           30
        ;MOVLW          3
        MOVWF           COUNTER2

WAIT_2:
        MOVLW           27
        ;MOVLW          3
        MOVWF           COUNTER

WAIT_1:
        BTFSC           PIR1,RCIF               ;If RCIF = '0', receive buffer is empty
        GOTO            GETRC_1
        DECFSZ          COUNTER, F
        GOTO            WAIT_1

        DECFSZ          COUNTER2, F
        GOTO            WAIT_2

        DECFSZ          COUNTER3, F
        GOTO            WAIT_3

        BSF             CODRET1,0       ; when time_out error, CODRET = %11
        BSF             CODRET1,1
        RETURN

GETRC_1:
                                        ;If receive buffer is full, must be captured.
        CLRF            CODRET1
        MOVF            RCREG, W
        RETURN

;-------------------------------------------------------------------------------------------
SEND_U:
        MOVLW           30
        MOVWF           COUNTER4
TX_U:
```

```
        MOVLW   CHAR_U
        CALL    TXFUNC
        DECFSZ  COUNTER4, F
        GOTO    TX_U

        RETURN
;-------------------------------------------------------------------------------------------
TRM_MACK:
        MOVLW   CHAR_@
        CALL    TXFUNC

        MOVLW   CHAR_@
        CALL    TXFUNC

        MOVLW   CHAR_@
        CALL    TXFUNC

        MOVF    MACK_1, W
        CALL    TXFUNC

        MOVF    MACK_2, W
        CALL    TXFUNC

        RETURN
;-------------------------------------------------------------------------------------
REC_MACK:
        CLRF    CODRET1
        MOVLW   CHAR_@          ;Test EOT ('@'): char to begin the comunication
        CALL    COMP_W_DATA
        BTFSS   CODRET1,0
        GOTO    EOT_OK
        RETURN

EOT_OK:
        MOVLW   CHAR_@
        CALL    COMP_W_DATA
        BTFSC   CODRET1,0
        RETURN

        MOVLW   CHAR_@
        CALL    COMP_W_DATA
        BTFSC   CODRET1,0
        RETURN

        MOVF    MACK_1, W
        CALL    COMP_W_DATA
        BTFSC   CODRET1,0
        RETURN

        MOVF    MACK_2, W
        CALL    COMP_W_DATA
        RETURN


;-------------------------------------------------------------------------------------
WAIT_REC_MACK:
        MOVLW   %00110001       ;Start TIMER 1 (16 bits) in timer mode (internal clock) with 1:8
prescaler
        MOVWF   T1CON
        MOVLW   254
        MOVWF   TMR1H           ; when tmr1 overfolw 0.5s has passed
        MOVWF   TMR1L

        CLRF    CODRET1
        BCF     PIR1,TMR1IF     ;limpa o flag de overflow
        MOVLW   10              ;teste de 10 bytes 255 para detectar a portadora
        MOVWF   COUNTER5
```

```
        MOVLW       10
        MOVWF       COUNTER6

TEST_REC_MACK:
        MOVLW       CHAR_@
        CALL        COMP_W_DATA
        BTFSC       CODRET1, 0
        GOTO        TEST_TIME_OUT           ;se não chegou 255 testa se deu time-out

        CALL        EOT_OK
        BTFSC       CODRET1, 0
        GOTO        TEST_TIME_OUT           ;se não chegou 255 testa se deu time-out
        GOTO        END_WAIT_REC_MACK

TEST_TIME_OUT:
        BTFSS       PIR1,TMR1IF             ;se deu overflow então time-out
        GOTO        TEST_REC_MACK           ;se não, continua o teste de 255

        BCF         PIR1, TMR1IF            ;limpa o flag de overflow
        DECFSZ      COUNTER6, F
        GOTO        TEST_REC_MACK

        BSF         CODRET1, 0
        BSF         CODRET1, 1

END_WAIT_REC_MACK:
        BCF         T1CON, TMR1ON           ;ao sair da rotina desliga o timer 1
        BCF         PIR1, TMR1IF            ;limpa o flag de overflow
        RETURN


;-------------------------------------------------------------------------------------
COMP_W_DATA:
        MOVWF       DATA
        CALL        RCFUNC_0    ;RCFUNC_0 returns CODRET1 = %11 on time out error and %00
on sucess
        BTFSC       CODRET1,1
        RETURN                  ;COMP_W_DATA returns CODRET1 = %11 timeout error
        CLRF        CODRET1
        MOVWF       DATARET     ;DATARET receive data in RCREG
        SUBWF       DATA, W
        BTFSC       STATUS, Z
        RETURN
        BCF         CODRET1,1   ;COMP_W_DATA returns CODRET1 = %01 on Comparation error
        BSF         CODRET1,0
        RETURN
;-------------------------------------------------------------------------------------
LOAD_EE_REGS:
        MOVLW       END_EE_TMR0
        CALL        READ_EE
        MOVWF       TMR0

        MOVLW       END_EE_TOTAL_1
        CALL        READ_EE
        MOVWF       TOTAL_1

        MOVLW       END_EE_TOTAL_2
        CALL        READ_EE
        MOVWF       TOTAL_2

        MOVLW       END_EE_TOTAL_3
        CALL        READ_EE
        MOVWF       TOTAL_3

        MOVLW       END_EE_TOTAL_4
        CALL        READ_EE
        MOVWF       TOTAL_4
```

```
        MOVLW       END_EE_MACK_1
        CALL        READ_EE
        MOVWF       MACK_1

        MOVLW       END_EE_MACK_2
        CALL        READ_EE
        MOVWF       MACK_2

        MOVLW       END_EE_STATE
        CALL        READ_EE
        MOVWF       STATE

        RETURN
;-------------------------------------------------------------------------------------------
WRITE_EE_TOTAL:
        MOVF        TOTAL_1, W
        MOVWF       DATA_EE
        MOVLW       END_EE_TOTAL_1
        CALL        WRITE_EE

        MOVF        TOTAL_2, W
        MOVWF       DATA_EE
        MOVLW       END_EE_TOTAL_2
        CALL        WRITE_EE

        MOVF        TOTAL_3, W
        MOVWF       DATA_EE
        MOVLW       END_EE_TOTAL_3
        CALL        WRITE_EE

        MOVF        TOTAL_4, W
        MOVWF       DATA_EE
        MOVLW       END_EE_TOTAL_4
        CALL        WRITE_EE

        RETURN

WRITE_EE_STATE:
        MOVF        STATE, W            ;salvando STATE na EEPROM
        MOVWF       DATA_EE
        MOVLW       END_EE_STATE
        CALL        WRITE_EE

        RETURN

;------------WAIT 500ms-----------------------------------------------------------------------
WAIT_100ms:
        MOVLW       50
        MOVWF       COUNTER3
B2:     CALL        WAIT_10ms
        DECFSZ      COUNTER3, F
        GOTO        B2

        RETURN
;------Wait 1ms, 9ms or 10ms-------------------------------------------------------------

WAIT_1ms:

        MOVLW       6
        MOVWF       COUNTER2
        GOTO        B0

WAIT_9ms:
        MOVLW       54
        MOVWF       COUNTER2
        GOTO        B0
```

```
WAIT_10ms:
        MOVLW       60
        MOVWF       COUNTER2
        GOTO        B0

B0:
        ;MOVLW      27
        MOVLW       50
        MOVWF       COUNTER

B1:     DECFSZ      COUNTER, F
        GOTO        B1

        DECFSZ      COUNTER2, F
        GOTO        B0

        RETURN
;-----------------------------------------------------------------------------------------
INTERRUPT:

GIE_CK_01:
        BCF         INTCON, GIE
        BTFSC       INTCON, GIE
        GOTO        GIE_CK_01

        MOVWF       SAVE_W          ;saving the context of the main program
        MOVF        STATUS,W
        MOVWF       SAVE_STATUS

        BTFSS       INTCON, TMR0IF
        GOTO        END_INT

        MOVLW       1
        ADDWF       TOTAL_1, F
        BTFSC       STATUS, C
        ADDWF       TOTAL_2, F
        BTFSC       STATUS, C
        ADDWF       TOTAL_3, F
        BTFSC       STATUS, C
        ADDWF       TOTAL_4, F

SAVE_TOTAL:
        BCF         INTCON, TMR0IF
        CALL        WRITE_EE_TOTAL

        ;movlw      250
        ;movwf      tmr0

        BCF         INTCON, TMR0IF
        GOTO        END_INT

END_INT:
        MOVF        SAVE_STATUS,W   ;reestoring the context of the main program
        MOVWF       STATUS
        MOVF        SAVE_W, W

        BSF         INTCON, GIE     ;ENABLE all interrupts
        RETFIE                      ; end of INTERRUPT

;-----------------------------------------------------------------------------------------

MAIN_PROG:
        ;---------carregando dados da EEPROM-------------------------------------------------------
        CALL        LOAD_EE_REGS

        ;---------configurando TMR0 como TOTALIZADOR -----------------------------------------------
```

```
        BSF             STATUS, RP0
        MOVLW           %00111000
        MOVWF           OPTION_REG
        BCF             TRISB,0                 ; PORTB,0 como saída para o LED
        BSF             TRISA, 4                ;RA4, entrada clock para o contador TMR0
        BCF             STATUS, RP0

        ;-----Configurando Registradores para Comunicacao Serial TX e RX------------------------------
        BSF             STATUS,RP0
        MOVLW           255             ;BAUD RATE = 1200bps for Fosc = 20MHz crystal
        MOVWF           SPBRG
        BSF             TRISB,2
        BSF             TRISB,5
        MOVLW           %00100010       ;BRGH = 0 (Low Speed), SYNC = 0 (Assynchronous), //TX enable
        MOVWF           TXSTA
        BCF             STATUS,RP0
        MOVLW           %10010000
        MOVWF           RCSTA

        ;------configurando a INT----------------------------------------------------------------
        MOVLW           %11100000
        MOVWF           INTCON
        BSF             STATUS, RP0
        BCF             PIE1, RCIE
        BCF             STATUS, RP0

        ;------configurando as portas do AD para digital ----------------------------------------------
        BSF             STATUS, RP0
        CLRF            ANSEL
        BCF             STATUS, RP0

        MOVF            TMR0, W             ;valor do TMR0 igual a PREV_TMR0 para mudança
        MOVWF           PREV_TMR0           ;RB0 led para entrada do consumo
        BSF             PORTB, 0            ;acender o led para indicar o funcionamento
        BSF             STATUS, RP0
        BCF             TRISB, 4            ;RB4, saída TX modulada
        BSF             TRISB, 6            ;RB6, entrada realimentada do sinal serial TX modular
        BSF             TRISB, 3            ;RB3, entrada do sensor de fase
        BCF             TRISB, 7            ;RB7, saída RTS do transmissor
        BCF             TRISA, 1            ;RA1, saída corte/religação de energia
        BCF             STATUS, RP0

        BCF             PORTB, 7            ;manter o transmissor desligado

        BCF             PORTA, 1            ;fornecimento desligado
        BTFSC           STATE, 0            ;testa para ver se o fornecimento anteior estava ligado ou
desligado
        BSF             PORTA, 1


MAIN_FUNC:
        BTFSC           PIR1, RCIF
        GOTO            REC_INT

        MOVF            TMR0, W             ;compara TRM0 com o anterior , ver se houve mudança
        SUBWF           PREV_TMR0, W
        BTFSC           STATUS, Z           ;se houve, flag Z não é zero, então vai para SAVE_TMR0
e registra
        GOTO            MAIN_FUNC


SAVE_TMR0:
        BCF             INTCON, GIE         ;desabilita as interrupções
        BTFSC           INTCON, GIE
        GOTO            SAVE_TMR0

        MOVF            TMR0, W
```

```
        MOVWF       DATA_EE                 ;salva TMR0 na EEPROM
        MOVLW       END_EE_TMR0
        CALL        WRITE_EE

        MOVF        TMR0, W                 ;faz TMR0 anterior igual ao novo valor de
TMR0
        MOVWF       PREV_TMR0

        BSF         INTCON, GIE
        GOTO        MAIN_FUNC


SAVE_PHASE:
        BCF         INTCON, GIE             ;desabilita as interrupções
        BTFSC       INTCON, GIE
        GOTO        SAVE_PHASE

        BCF         STATE, 1
        CALL        WRITE_EE_STATE

        BSF         INTCON, GIE
        GOTO        MAIN_FUNC

        ;-------------------------------------------------------------------------------
        ; comunicação serial - implementa o protocolo de comunicação
REC_INT:
        BCF         PORTB,0

        CALL        REC_MACK
        BTFSC       CODRET1,0
        GOTO        END_REC

        CALL        RCFUNC_0
        BTFSC       CODRET1,0
        GOTO        END_REC
        MOVWF       DATA

        MOVLW       CHAR_L
        SUBWF       DATA, W
        BTFSC       STATUS, Z
        GOTO        REC_LER

        MOVLW       CHAR_C
        SUBWF       DATA, W
        BTFSC       STATUS, Z
        GOTO        REC_CORTAR

        MOVLW       CHAR_A
        SUBWF       DATA, W
        BTFSC       STATUS, Z
        GOTO        REC_ACIONAR

        MOVLW       CHAR_R
        SUBWF       DATA, W
        BTFSC       STATUS, Z
        GOTO        REC_RESET

        GOTO        END_REC


        ;-------------------------------------------------------------------------------
REC_LER:                                ;recebimento do comando "@@@ end1 end2 L" está ok
                                        ;enviando o valor do totalizador "@@@ end1 end2 L b1 b2 b3 b4
b5"
        CALL        WAIT_100ms
        CALL        WAIT_100ms
        BSF         PORTB, 7
```

```
CALL        SEND_U

CALL        TRM_MACK
MOVLW       CHAR_L
CALL        TXFUNC

MOVF        TMR0, W
MOVWF       SEND_TMR0
CALL        TXFUNC
MOVF        TOTAL_1, W
MOVWF       SEND_TOTAL_1
CALL        TXFUNC
MOVF        TOTAL_2, W
MOVWF       SEND_TOTAL_2
CALL        TXFUNC
MOVF        TOTAL_3, W
MOVWF       SEND_TOTAL_3
CALL        TXFUNC
MOVF        TOTAL_4, W
MOVWF       SEND_TOTAL_4
CALL        TXFUNC

BCF         PORTB, 7
;GOTO       END_REC         ;não vai ainda ao final do protocolo
                            ;recebendo a confirmação do totalizador

CALL        WAIT_REC_MACK
BTFSC       CODRET1, 0
GOTO        END_REC

MOVLW       CHAR_L
CALL        COMP_W_DATA
BTFSC       CODRET1, 0
GOTO        END_REC

MOVF        SEND_TMR0, W
CALL        COMP_W_DATA
MOVF        DATARET, W
MOVWF       TMR0
BTFSC       CODRET1, 0
GOTO        SEND_NACK

MOVF        SEND_TOTAL_1, W
CALL        COMP_W_DATA
MOVF        DATARET, W
MOVWF       TOTAL_1
BTFSC       CODRET1, 0
GOTO        SEND_NACK

MOVF        SEND_TOTAL_2, W
CALL        COMP_W_DATA
MOVF        DATARET, W
MOVWF       TOTAL_2
BTFSC       CODRET1, 0
GOTO        SEND_NACK

MOVF        SEND_TOTAL_3, W
CALL        COMP_W_DATA
MOVF        DATARET, W
MOVWF       TOTAL_3
BTFSC       CODRET1, 0
GOTO        SEND_NACK

MOVF        SEND_TOTAL_4, W
CALL        COMP_W_DATA
MOVF        DATARET, W
MOVWF       TOTAL_4
```

```
        BTFSC   CODRET1, 0
        GOTO    SEND_NACK

        MOVLW   CHAR_?
        CALL    COMP_W_DATA
        BTFSC   CODRET1, 0
        GOTO    SEND_NACK

SEND_ACK_L:                                     ;confirmação do totalizador válida enviar ack
        CALL    WAIT_100ms
        CALL    WAIT_100ms
        BSF     PORTB, 7
        CALL    SEND_U

        CALL    TRM_MACK
        MOVLW   CHAR_L
        CALL    TXFUNC
        MOVLW   CHAR_O
        CALL    TXFUNC
        MOVLW   CHAR_K
        CALL    TXFUNC

        BCF     PORTB, 7
        GOTO    END_REC

SEND_NACK:
        CALL    WAIT_100ms
        CALL    WAIT_100ms
        BSF     PORTB, 7
        CALL    SEND_U

        CALL    TRM_MACK
        MOVLW   CHAR_L
        CALL    TXFUNC
        MOVLW   CHAR_?
        CALL    TXFUNC
        MOVLW   CHAR_?
        CALL    TXFUNC

        BCF     PORTB, 7
        GOTO    END_REC
        ;-------------------------------------------------------------------------------
REC_CORTAR:
        MOVLW   CHAR_T
        CALL    COMP_W_DATA
        BTFSC   CODRET1, 0
        GOTO    END_REC

        CALL    WAIT_100ms
        BSF     PORTB, 7
        CALL    SEND_U

        CALL    TRM_MACK
        MOVLW   CHAR_C
        CALL    TXFUNC
        MOVLW   CHAR_T
        CALL    TXFUNC
        MOVLW   CHAR_?
        CALL    TXFUNC

        BCF     PORTB, 7

        CALL    WAIT_REC_MACK
        BTFSC   CODRET1, 0
        GOTO    END_REC

        MOVLW   CHAR_C
```

```
        CALL    COMP_W_DATA
        BTFSC   CODRET1, 0
        GOTO    END_REC

        MOVLW   CHAR_T
        CALL    COMP_W_DATA
        BTFSC   CODRET1, 0
        GOTO    END_REC

        MOVLW   CHAR_!
        CALL    COMP_W_DATA
        BTFSC   CODRET1, 0
        GOTO    END_REC

        CALL    WAIT_100ms

        BSF     PORTB, 7
        CALL    SEND_U

        CALL    TRM_MACK
        MOVLW   CHAR_C
        CALL    TXFUNC
        MOVLW   CHAR_T
        CALL    TXFUNC
        MOVLW   CHAR_O
        CALL    TXFUNC
        MOVLW   CHAR_K
        CALL    TXFUNC

        BCF     PORTB, 7

        ;efetuando o corte
        BCF     PORTA, 1
        BCF     STATE, 0
        CALL    WRITE_EE_STATE

        GOTO    END_REC

        ;-------------------------------------------------------------------------------
REC_ACIONAR:

        MOVLW   CHAR_C
        CALL    COMP_W_DATA
        BTFSC   CODRET1, 0
        GOTO    END_REC

        CALL    WAIT_100ms
        BSF     PORTB, 7
        CALL    SEND_U

        CALL    TRM_MACK
        MOVLW   CHAR_A
        CALL    TXFUNC
        MOVLW   CHAR_C
        CALL    TXFUNC
        MOVLW   CHAR_?
        CALL    TXFUNC

        BCF     PORTB, 7

        CALL    WAIT_REC_MACK
        BTFSC   CODRET1, 0
        GOTO    END_REC

        MOVLW   CHAR_A
        CALL    COMP_W_DATA
        BTFSC   CODRET1, 0
```

```
        GOTO        END_REC

        MOVLW       CHAR_C
        CALL        COMP_W_DATA
        BTFSC       CODRET1, 0
        GOTO        END_REC

        MOVLW       CHAR_!
        CALL        COMP_W_DATA
        BTFSC       CODRET1, 0
        GOTO        END_REC

        CALL        WAIT_100ms

        BSF         PORTB, 7
        CALL        SEND_U

        CALL        TRM_MACK
        MOVLW       CHAR_A
        CALL        TXFUNC
        MOVLW       CHAR_C
        CALL        TXFUNC
        MOVLW       CHAR_O
        CALL        TXFUNC
        MOVLW       CHAR_K
        CALL        TXFUNC

        BCF         PORTB, 7

        ;efetuando o acionamento
        BSF         PORTA, 1
        BSF         STATE, 0
        CALL        WRITE_EE_STATE

        GOTO        END_REC

        ;-------------------------------------------------------------------------------------
REC_RESET:


        MOVLW       CHAR_E
        CALL        COMP_W_DATA
        BTFSC       CODRET1, 0
        GOTO        END_REC

        CALL        WAIT_100ms
        CALL        WAIT_100ms
        BSF         PORTB, 7
        CALL        SEND_U

        CALL        TRM_MACK
        MOVLW       CHAR_R
        CALL        TXFUNC
        MOVLW       CHAR_E
        CALL        TXFUNC
        MOVLW       CHAR_?
        CALL        TXFUNC

        BCF         PORTB, 7

        CALL        WAIT_REC_MACK
        BTFSC       CODRET1, 0
        GOTO        END_REC

        MOVLW       CHAR_R
        CALL        COMP_W_DATA
        BTFSC       CODRET1, 0
```

```
        GOTO        END_REC

        MOVLW       CHAR_E
        CALL        COMP_W_DATA
        BTFSC       CODRET1, 0
        GOTO        END_REC

        MOVLW       CHAR_!
        CALL        COMP_W_DATA
        BTFSC       CODRET1, 0
        GOTO        END_REC

        CALL        WAIT_100ms
        CALL        WAIT_100ms

        BSF         PORTB, 7
        CALL        SEND_U

        CALL        TRM_MACK
        MOVLW       CHAR_R
        CALL        TXFUNC
        MOVLW       CHAR_E
        CALL        TXFUNC
        MOVLW       CHAR_O
        CALL        TXFUNC
        MOVLW       CHAR_K
        CALL        TXFUNC

        BCF         PORTB, 7

        ;efetuando o reset do contador

        CLRF        TMR0
        CLRF        TOTAL_1
        CLRF        TOTAL_2
        CLRF        TOTAL_3
        CLRF        TOTAL_4

        CALL        WRITE_EE_TOTAL

END_REC:
        BCF         PIR1, RCIF
        BSF         INTCON, GIE
        BSF         PORTB, 0
        GOTO        MAIN_FUNC

        END
```

Programa Modem

```
;==========================================================================================

;       FEDERAL UNIVERSITY OF RIO DE JANEIRO
;       ENGINEERING SCHOOL


;       PROJETO:        Projeto final - Medidor Eletrônico de energia com PLC
;       FIRMWARE:       medidor_PF_ver1


;       AUTOR:          Paulo Gentil Gibson Fernandes
;       ORIENTADOR:     Carlos José Ribas D'Ávila

;==========================================================================================

;       OBSERVAÇÕES:

;       Programa desenvolvido para ser executado em microcontroladores tipo PIC16F88.
;       Este código implementa um conversor RS-232 PLC.
;       O firmware se comunica ao Medidor à uma taxa de 2400 bps e
;       ao concentrador a uma taxa de 1200 bps.



;------Declarando as variáveis globais ----------------------------------------------

        COUNTER                 EQU 0x30
        CODRET1                 EQU 0x31
        COUNTER2                EQU 0x32
        COUNTER3                EQU 0x33
        COUNTER5                EQU 0x34
        COUNTER6                EQU 0x35
        SAVE_W                  EQU 0x36
        SAVE_STATUS             EQU 0x37
        DATA                    EQU 0x38
        DATARET                 EQU 0x39
        EOT                     EQU 0x3A
        COUNTER_MOD             EQU 0x3B
        FLAG_MOD                EQU 0x3C
        FLAG_SEND_TO_PLC        EQU 0x3D
        COUNTER4                EQU 0x3E

;------Declarando as constantes -----------------------------------------------------

        CHAR_U                  EQU 255
        CHAR_@                  EQU 64
;------------------------------------------------------------------------------------

        ORG             0x00
        GOTO            MAIN_PROG
        ORG             0x04
        GOTO            INTERRUPT

;-----Function MODULATE DATA---------------------------------------------------------
TXFUNC:
        BCF             RCSTA, 4        ;DESeabilita a recepção serial
        BTFSS           PIR1,TXIF       ;If TXIF = '0', transmit buffer is full
        GOTO            TXFUNC
        MOVWF           TXREG
MAIN:
        BSF             STATUS, RP0
        BTFSC           TXSTA, TRMT
        GOTO            END_TXFUNC
```

```
        BCF     STATUS, RP0

        MOVLW   3                       ; alterar o contador em 1 unidade o contador
                                        ; reflete em 3 ciclos de clock
        MOVWF   COUNTER_MOD
        INCF    FLAG_MOD, F
LOOP:
        DECFSZ  COUNTER_MOD, F
        GOTO    LOOP
        ;NOP                            ;incrementar NOPS para aumentar o periodo do ciclo
        BTFSC   FLAG_MOD, 0             ;inferior e superior: 1 NOP = 2 cilcos de clk de atraso
        GOTO    SET_HI
SET_LOW:
        BCF     PORTB, 4
        BTFSS   PORTB, 6
        GOTO    MAIN
        NOP
        GOTO    MAIN

SET_HI:
        BSF     PORTB, 4
        NOP
        ;NOP                            ;incrementar NOPs para aumentar o semiperiodo inferior
                                        ; 1 NOP = 1 ciclo de clk de atraso
        GOTO    MAIN

END_TXFUNC:
        BCF     STATUS, RP0
        BSF     RCSTA, 4        ;reabilita a recepção serial
        RETURN

;-----Function to receive data and return data in W reg-------------------------
RCFUNC_0:
        CLRF    CODRET1

        BTFSC   PIR1,RCIF       ;If RCIF = '0', receive buffer is empty
        GOTO    GETRC_1

        MOVLW   190
        ;MOVLW  2
        MOVWF   COUNTER3

WAIT_3:
        MOVLW   40
        ;MOVLW  2
        MOVWF   COUNTER2

WAIT_2:
        MOVLW   15
        ;MOVLW  2
        MOVWF   COUNTER

WAIT_1:
        BTFSC   PIR1,RCIF       ;If RCIF = '0', receive buffer is empty
        GOTO    GETRC_1
        DECFSZ  COUNTER, F
        GOTO    WAIT_1

        DECFSZ COUNTER2, F
        GOTO    WAIT_2

        DECFSZ COUNTER3, F
        GOTO    WAIT_3

        BSF     CODRET1,0       ;  when time_out error, CODRET = %11
        BSF     CODRET1,1
        RETURN
```

```
GETRC_1:
        ;If receive buffer is full, must be captured.
        CLRF        CODRET1
        MOVF        RCREG, W
        RETURN

;-------------------------------------------------------------------------------------
COMP_W_DATA:
        MOVWF       DATA
        CALL        RCFUNC_0        ; CODRET1 = 11 on time out error and 00 on sucess
        BTFSC       CODRET1,1
        RETURN                      ;COMP_W_DATA returns CODRET1 = %11 timeout error
        CLRF        CODRET1
        MOVWF       DATARET         ;DATARET receive data in RCREG
        SUBWF       DATA, W
        BTFSC       STATUS, Z
        RETURN
        BCF         CODRET1,1       ;COMP_W_DATA returns CODRET1 = %01 on Comparation error
        BSF         CODRET1,0
        RETURN

;-------------------------------------------------------------------------------------------------
REC_FROM_PC:
        BSF         FLAG_SEND_TO_PLC, 0
CONF_BR2400:
        BSF         STATUS,RP0
        MOVLW       129                     ;BAUD RATE = 2400bps for Fosc = 20MHz crystal
        MOVWF       SPBRG
        BCF         STATUS,RP0
        RETURN

;-------------------------------------------------------------------------------------------------
REC_FROM_PLC:
        BCF         FLAG_SEND_TO_PLC, 0
CONF_BR1221:
        BSF         STATUS,RP0
        MOVLW       255                     ;BAUD RATE = 1221bps for Fosc = 20MHz crystal
        MOVWF       SPBRG
        BCF         STATUS,RP0
        RETURN

;-------------------------------------------------------------------------------------------------
SEND_TO_PLC:
        CALL        CONF_BR1221
        BSF         PORTB, 7
        MOVLW       35
        MOVWF       COUNTER4
SEND_U:
        MOVLW       CHAR_U
        CALL        TXFUNC
        DECFSZ      COUNTER4, F
        GOTO        SEND_U

        RETURN


;-------------------------------------------------------------------------------------------------
MAIN_PROG:

        ;----Configurando Registradores para Comunicacao Serial TX e RX-------------------------------
        BSF         STATUS,RP0
        MOVLW       255             ;BAUD RATE = 1221bps for Fosc = 20MHz crystal
        MOVWF       SPBRG
        BSF         TRISB,2
        BSF         TRISB,5
```

```
        MOVLW   %00100010       ;BRGH = 0 (Low Speed), SYNC = 0 (Assynchronous Mode), //TX
enable
        MOVWF   TXSTA
        BCF     STATUS,RP0
        MOVLW   %10010000
        MOVWF   RCSTA

        ;------configurando a INT--------------------------------------------------------------
        MOVLW   %11000000
        MOVWF   INTCON
        BSF     STATUS, RP0
        BSF     PIE1, RCIE
        BCF     STATUS, RP0

        ;------configurando as portas do AD para digital ------------------------------------------------
        BSF     STATUS, RP0
        CLRF    ANSEL
        BCF     STATUS, RP0

        BSF     STATUS, RP0
        BCF     TRISB, 0        ;RB0 como saída para o LED
        BCF     TRISB, 4        ;RB4, saída TX modulada
        BSF     TRISB, 6        ;RB6, entrada realimentada do sinal serial TX para modular
        BSF     TRISB, 3        ;RB3, entrada do sensor de fase
        BCF     TRISB, 7        ;RB7, saída RTS do transmissor
        BCF     TRISA, 1        ;RA1, saída corte/religação de energia
        BSF     TRISA, 0        ;RA0, entrada do RTS do concentrador
        BCF     STATUS, RP0
        CLRF    FLAG_SEND_TO_PLC        ;limpa o flag de envio para o PLC

RC_LOOP:
        BSF     PORTB, 0        ;acende o LED
        BCF     PORTB, 7        ;manter o transmissor desligado
        GOTO    RC_LOOP

INTERRUPT:
        BCF     INTCON, GIE     ;Desable all interrupts in a security way and to perform a savety
communication
        BTFSC   INTCON, GIE
        GOTO    INTERRUPT

        MOVWF   SAVE_W          ;saving the context of the main program
        MOVF    STATUS,W
        MOVWF   SAVE_STATUS

        BCF     PORTB, 0        ;apaga o LED
        BTFSS   PIR1, RCIF
        GOTO    END_INT
                                        ;programa já configurado para receber do PLC
        BTFSS   PORTA, 0                ;testa o RTS_PC
        CALL    REC_FROM_PC             ;configurar o programa para receber do PC

        MOVLW   CHAR_@                  ;verifica se chegou o @
        CALL    COMP_W_DATA
        BTFSC   CODRET1,0
        GOTO    END_INT

        MOVLW   CHAR_@                  ;verifica se chegou o @
        CALL    COMP_W_DATA
        BTFSC   CODRET1,0
        GOTO    END_INT

        MOVLW   CHAR_@                  ;verifica se chegou o @
        CALL    COMP_W_DATA
        BTFSC   CODRET1,0
        GOTO    END_INT
```

```
        MOVLW   0x20                    ;define o valor inicial do registrador de acesso indireto
        MOVWF   FSR
        MOVWF   COUNTER5

REC_BUFFER:

        CALL    RCFUNC_0
        MOVWF   DATA
        BTFSC   CODRET1, 0              ;se deu time-out então começa transmitir
        GOTO    BEGIN_SEND

        MOVF    COUNTER5, W             ;joga o valor de counter5 para o registrador
        MOVWF   FSR
        MOVF    DATA, W
        MOVWF   INDF

        INCF    COUNTER5, F             ;receive until 10 bytes
        MOVLW   0x29
        SUBWF   COUNTER5, W             ;test if COUNTER == 10
        BTFSS   STATUS, Z
        GOTO    REC_BUFFER

BEGIN_SEND:

        CALL    CONF_BR2400             ;configura o programa para enviar ao PC
        BTFSC   FLAG_SEND_TO_PLC, 0     ;testa o Flag para envio ao PLC
        CALL    SEND_TO_PLC             ;configura para enviar ao PLC

        MOVLW   CHAR_@
        CALL    TXFUNC
        MOVLW   CHAR_@
        CALL    TXFUNC
        MOVLW   CHAR_@
        CALL    TXFUNC
        MOVLW   0x20                    ;define the initial address of buffer (the value is defined in
INDF register)
        MOVWF   FSR
        MOVWF   COUNTER6                ;initialize counter with the first address

SEND_BUFFER:

        MOVF    COUNTER6, W             ;define the correct address
        MOVWF   FSR

        MOVF    INDF, W                 ;get the value of register pointed by FSR
        CALL    TXFUNC

        INCF    COUNTER6, F             ;receive until 14 bytes
        MOVF    COUNTER5, W
        SUBWF   COUNTER6, W             ;test if COUNTER = 14
        BTFSS   STATUS, Z
        GOTO    SEND_BUFFER
        GOTO    END_INT

END_INT:
        CALL    REC_FROM_PLC            ;configura para receber do PLC

        MOVF    SAVE_STATUS,W           ;reestoring the context of the main program
        MOVWF   STATUS
        MOVF    SAVE_W, W

        BSF     INTCON, GIE             ;ENABLE all interrupts
        RETFIE                          ; end of INTERRUPT


        END
```

Programa Teste Concentrador

Função Principal:

Programa para ser executado no PC e se comunicar com o Conversor RS-232 PLC a uma taxa de 2400bps. Este programa implementa o protocolo de comunicação para se comunicar com o microcontrolador PIC16F88.

```
#include<iostream.h>
#include<stdlib.h>
#include<time.h>
#include<stdio.h>
#include <string>
#include "SerialPort5.h"
using namespace std;


long int CharToInt( char bytes[], int tam, unsigned short int num[] )
{
	//int num [10];
	long int valor = 0;

	for(int i=0; i< tam; i++)
	{
		if( bytes[i] < 0)
			num[i] = (int) bytes[i]+256;
		else
			num[i] = (int) bytes[i];
	}
	for(i=0; i<5; i++)
		printf(" byte%d: %d\n", i, num[i]);

	   valor = num[0] + num[1]*256 + num[2]*256*256 + num[3]*256*256*256 + num[4]*256*256*256*256;
	return( valor );
}

int main()
{
	char receber[30], receber2[30], bytes[10], enviar[20], option, rec_eot[2];
	int end1, end2, num_bytes;
	unsigned short int num[10];
	BYTE valor;
	long int totalizador;
	//FILE *arq;
	CSerialPort serial;


	if (!serial.OpenPort())
	{
		cout << "Nao e possível abrir porta" << endl;
		return 0;
	}

	if (!serial.ConfigurePort (CBR_2400,8,1,NOPARITY,ONESTOPBIT, 0))
	{
		cout << "Nao e possível configurar porta" << endl;
		serial.ClosePort();
		return 0;
	}

	if (!serial.SetCommunicationTimeouts(500,500,100,10000,10000))
	{
		cout << "Nao é possível ajustar timeout da porta" << endl;
		serial.ClosePort();
		return 0;
	}

	serial.ClosePort();
	cout << "Aplicativo pronto para operar." << endl;
```

```
	for(;;)
	{
		cout << endl;
		cout << "Digite:"<<endl;
		cout << "  L para leitura do consumo" << endl;
		cout << "  C para corte do fornecimento ao cliente"<<endl;
		cout << "  A para religar o fornecimento" << endl;
		cout << "  R para reset do totalizador"<< endl;
		cin >> option;

		if (!serial.OpenPort())
		{
			cout << "Nao e possível abrir porta" << endl;
			return 0;
		}

		if (option=='l' || option=='L')
		{


//-------------------------------------TESTE CHAR-----------------------------------------
			num[0] = 0;
			sprintf(enviar, "%c%c%c", num[0], num[0], num[0]);

//-------------------------------------PEDIDO LEITURA-------------------------------------
			sprintf(enviar,"@@@12L");
			serial.SendStr (enviar, 6);

			serial.ReadByte(receber, 6);
			receber[6] = '\0';

			printf("Rx: %s\n", receber);

		if ( strcmp(receber, enviar) == 0)
		{
			serial.ReadByte(bytes, 5);
			receber[5] = '\0';


			totalizador = CharToInt( bytes, 5, num );

			serial.ClosePort();
			serial.OpenPort();

			sprintf(enviar,"@@@12L%c%c%c%c%c?", num[0], num[1], num[2], num[3], num[4]);
			serial.SendStr (enviar, 12);

			serial.ReadByte(receber2, 8);
			receber2[8] = '\0';
			printf("Rx: %s\n", receber2);

		}


			printf ("totalizador: %d\n", totalizador);
		}
//-----------------------------------------------------------------------------------------

		if (option=='C' || option=='c')
		{
			sprintf(enviar,"@@@12CT");
			serial.SendStr (enviar, 7);
			cout << "Tx: " << enviar << endl;

			serial.ReadByte(receber, 8);
			receber[8] = '\0';
			printf("Rx: %s\n", receber);

			if ( strcmp(receber, "@@@12CT?" ) == 0)
			{
				serial.ClosePort();
				serial.OpenPort();

				sprintf(enviar,"@@@12CT!");
```

```
                    serial.SendStr (enviar, 8);
                    cout << "Tx: " << enviar << endl;

                    serial.ReadByte(receber2, 9);
                    receber2[9] = '\0';

                    printf("Rx: %s\n", receber2);

                    if ( strcmp(receber2, "@@@12CTOK" ) == 0)
                            printf("Fornecimento cortado!\n\n\n");
                    else
                            printf("Nao foi possivel cortar o fornecimento.\n\n\n");
            }
            else
                    printf("Nao foi possivel cortar o fornecimento.\n\n\n");

}

if (option=='a' || option=='A')
{
            sprintf(enviar,"@@@12AC");
            serial.SendStr (enviar, 7);
            cout << "Tx: " << enviar << endl;

            serial.ReadByte(receber, 8);
            receber[8] = '\0';
            printf("Rx: %s\n", receber);

            if ( strcmp(receber, "@@@12AC?" ) == 0)
            {
                    serial.ClosePort();
                    serial.OpenPort();

                    sprintf(enviar,"@@@12AC!");
                    serial.SendStr (enviar, 8);
                    cout << "Tx: " << enviar << endl;

                    serial.ReadByte(receber2, 9);
                    receber2[9] = '\0';

                    printf("Rx: %s\n", receber2);

                    if ( strcmp(receber2, "@@@12ACOK" ) == 0)
                            printf("Fornecimento religado!\n\n\n");
                    else
                            printf("Nao foi possivel religar o fornecimento.\n\n\n");
            }
            else
                    printf("Nao foi possivel religar o fornecimento.\n\n\n");

}

if (option=='R' || option=='r')
{
            sprintf(enviar,"@@@12RE");
            serial.SendStr (enviar, 7);
            cout << "Tx: " << enviar << endl;

            serial.ReadByte(receber, 8);
            receber[8] = '\0';
            printf("Rx: %s\n", receber);

            if ( strcmp(receber, "@@@12RE?" ) == 0)
            {
                    serial.ClosePort();
                    serial.OpenPort();

                    sprintf(enviar,"@@@12RE!");
                    serial.SendStr (enviar, 8);
                    cout << "Tx: " << enviar << endl;

                    serial.ReadByte(receber2, 9);
                    receber2[9] = '\0';

                    printf("Rx: %s\n", receber2);
```

```
                                        if ( strcmp(receber2, "@@@12REOK" ) == 0)
                                                printf("Totalizador reiniciado!\n\n\n");
                                        else
                                                printf("Nao foi possivel reiniciar o totalizador.\n\n\n");
                                }
                                else
                                        printf("Nao foi possivel reiniciar o totalizador.\n\n\n");

                        }

                        serial.ClosePort();
                }
}
```

Classe Para a porta serial:

```
#include <iostream.h>
#include "SerialPort5.h"

CSerialPort::CSerialPort()
{
}

CSerialPort::~CSerialPort()
{
}


BOOL CSerialPort::OpenPort()
{


	hComm = CreateFile("COM2",
		GENERIC_READ | GENERIC_WRITE,
		0,
		0,
		OPEN_EXISTING,
	FILE_FLAG_OVERLAPPED | FILE_ATTRIBUTE_NORMAL | FILE_FLAG_WRITE_THROUGH |
FILE_FLAG_NO_BUFFERING,
		0);

	if(hComm==INVALID_HANDLE_VALUE){
		return false;}
	else
		return true;

}

BOOL CSerialPort::ConfigurePort(DWORD BaudRate, BYTE ByteSize, DWORD fParity, BYTE Parity, BYTE StopBits,
BOOL Rts)
{


	if((m_bPortReady = GetCommState(hComm, &m_dcb))==0){
		cout << "ERRO1";
		CloseHandle(hComm);
		return false;
	}



	m_dcb.BaudRate =BaudRate;
	m_dcb.ByteSize = ByteSize;
	m_dcb.Parity =Parity ;
	m_dcb.StopBits =StopBits;
	m_dcb.fBinary=TRUE;
	m_dcb.fDsrSensitivity=false;
	m_dcb.fParity=fParity;
	m_dcb.fOutX=false;
	m_dcb.fInX=false;
	m_dcb.fNull=false;
	m_dcb.fAbortOnError=TRUE;
	m_dcb.fOutxCtsFlow=FALSE;
	m_dcb.fOutxDsrFlow=false;
	m_dcb.fDtrControl=DTR_CONTROL_DISABLE;


	if ( Rts == 1)
		m_dcb.fRtsControl=RTS_CONTROL_ENABLE;
	else
		m_dcb.fRtsControl=RTS_CONTROL_DISABLE;

	m_dcb.fOutxCtsFlow=false;


	m_bPortReady = SetCommState(hComm, &m_dcb);
	if(m_bPortReady ==0){
		cout << "ERRO2";
```

```
                    CloseHandle(hComm);
                    return false;}
            return true;
}

BOOL CSerialPort::SetCommunicationTimeouts(DWORD ReadIntervalTimeout, DWORD ReadTotalTimeoutMultiplier,
DWORD ReadTotalTimeoutConstant, DWORD WriteTotalTimeoutMultiplier, DWORD WriteTotalTimeoutConstant)
{
            if((m_bPortReady = GetCommTimeouts (hComm, &m_CommTimeouts))==0)
                    return false;
            m_CommTimeouts.ReadIntervalTimeout =ReadIntervalTimeout;
            m_CommTimeouts.ReadTotalTimeoutConstant =ReadTotalTimeoutConstant;
            m_CommTimeouts.ReadTotalTimeoutMultiplier =ReadTotalTimeoutMultiplier;
            m_CommTimeouts.WriteTotalTimeoutConstant = WriteTotalTimeoutConstant;
            m_CommTimeouts.WriteTotalTimeoutMultiplier =WriteTotalTimeoutMultiplier;
            m_bPortReady = SetCommTimeouts (hComm, &m_CommTimeouts);
            if(m_bPortReady ==0){
                    cout << "Erro3";
                    CloseHandle(hComm);
                    return false;}
            return true;
}

int CSerialPort::WriteByte(const char * Data, const int Length)
{
            COMSTAT             cs;
            unsigned long dwError = 0;
   OVERLAPPED ol;

            ClearCommError(hComm, &dwError, &cs);

            unsigned long retSize = 0;

   memset(&ol, 0, sizeof(OVERLAPPED));
   ol.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

   if(WriteFile(hComm, Data, Length, &retSize, &ol) == FALSE)
   {
      if(GetLastError() == ERROR_IO_PENDING)
      {
         WaitForSingleObject(ol.hEvent, INFINITE);
         GetOverlappedResult(hComm, &ol, &retSize, TRUE);
      }
      else
         ClearCommError(hComm, &dwError, &cs);
   }

   if(ol.hEvent != INVALID_HANDLE_VALUE)
      CloseHandle(ol.hEvent);

            return retSize;

}

int CSerialPort::SendByte(char Sbyte)
{
            char byte2[2];
            int i=0, x=0;

            byte2[0] = Sbyte;
            WriteByte(byte2,1);

            return 1;
}

int CSerialPort::SendStr( char* Str, int tam)
{
            int i=0, x=0;
            if (! ConfigurePort (CBR_2400,8,1,NOPARITY,ONESTOPBIT, 1))
            {
                    cout << "Nao e possível configurar porta" << endl;
                    ClosePort();
                    return 0;
            }
            for ( i=0; i<5000000; i++);
```

```
		for ( i=0; i < tam; i++)
			SendByte( Str[i]);


		for ( i=0; i<5000000; i++);


			if (! ConfigurePort (CBR_2400,8,1,NOPARITY,ONESTOPBIT, 0))
			{
			cout << "Nao e possível configurar porta2" << endl;
			ClosePort();
			return 0;
			}

		return 1;
}

int CSerialPort::ReadByte(char * Data, const int Length)
{

		unsigned long	dwLength = 0, dwError, dwReadLength;
		COMSTAT		cs;
   OVERLAPPED ol;

   if(hComm == INVALID_HANDLE_VALUE)
      return 0;

   ClearCommError(hComm, &dwError, &cs);

   if(dwLength > (int)cs.cbInQue)
      dwReadLength = cs.cbInQue;
   else
      dwReadLength = Length;


   memset(&ol, 0, sizeof(OVERLAPPED));
   ol.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

   if(dwReadLength > 0)
   {
      if(ReadFile(hComm, Data, dwReadLength, &dwLength, &ol) == FALSE)
      {
         if(GetLastError() == ERROR_IO_PENDING)
         {
            WaitForSingleObject(ol.hEvent, INFINITE);
            GetOverlappedResult(hComm, &ol, &dwLength, TRUE);
         }
         else
            ClearCommError(hComm, &dwError, &cs);
      }
   }

   if(ol.hEvent != INVALID_HANDLE_VALUE)
      CloseHandle(ol.hEvent);

		return dwLength;

}

BOOL CSerialPort::aWrite(BYTE bybyte)
{
		iBytesWritten=0;
		if(WriteFile(hComm,&bybyte,1,&iBytesWritten,NULL)==0)
			return false;
		else return true;
}

void CSerialPort::ClosePort()
{
		CloseHandle(hComm);
		return;
}
```

Cabeçalho:

```
#include"windows.h"

class CSerialPort
{
public:
	CSerialPort();

public:
	void ClosePort();

	int ReadByte(char * Data, const int Length);

	int WriteByte(const char * Data, const int Length);

	int CSerialPort::SendByte(char Sbyte);

	int CSerialPort::SendStr( char* Str, int tam);



	BOOL aWrite(BYTE bybyte);
	BOOL OpenPort();

	BOOL SetCommunicationTimeouts(DWORD ReadIntervalTimeout,DWORD
ReadTotalTimeoutMultiplier,DWORD ReadTotalTimeoutConstant,DWORD WriteTotalTimeoutMultiplier,DWORD
WriteTotalTimeoutConstant);
	BOOL ConfigurePort(DWORD BaudRate,BYTE ByteSize,DWORD fParity,BYTE  Parity,BYTE StopBits,
BOOL Rts);
	HANDLE hComm;
	DCB	m_dcb;
	COMMTIMEOUTS m_CommTimeouts;
	BOOL	m_bPortReady;
	BOOL	bWriteRC;
	BOOL	bReadRC;
	DWORD iBytesWritten;
	DWORD iBytesRead;
	DWORD dwBytesRead;
	virtual ~CSerialPort();
}
```