ESCOLA POLITÉCNICA DE PERNAMBUCO

Trabalho de Conclusão de Curso

Engenharia de Computação

Adaptação de um sistema microcontrolado de detecção e monitoramento de corrente de fuga para inserir comunicação sem fio

Autor: David Edson Ribeiro
Orientador: Sergio Campello Oliveira

UNIVERSIDADE DE PERNAMBUCO

ESCOLA POLITÉCNICA DE PERNAMBUCO

David Edson Ribeiro

Adaptação de um sistema microcontrolado de detecção e monitoramento de corrente de fuga para inserir comunicação sem fio

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Junho de 2010.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado forças durante todas as etapas do curso de graduação, pois sem ele temos uma vida vazia.

Agradeço a Sérgio Campello Oliveira, que além de me orientar no Trabalho de conclusão de curso, me deu a oportunidade de ser monitor por duas vezes em duas disciplinas da graduação.

Agradeço a minha família que sempre me apoiou durante os tempos difíceis e a minha Noiva, Kássia Regiane que me deu muito apoio durante esses anos de curso.

Agradeço a todos os meus colegas de curso, da minha turma e de outras turmas também, a convivência foi muito importante.

Agradeço a meus colegas de faculdade Giovane Boaviagem, Ismael Mascarenhas e a Fellipe Nery e a meu colega de trabalho Antônio Carlos Negreiros pelas contribuições a este trabalho.

Enfim, agradeço a todos que me incentivaram a seguir em frente e não desistir nos momentos difíceis, pessoas que tenho orgulho de chamar de amigos.

Resumo

Uma das formas de saber o momento realizar a manutenção de isoladores de linhas de transmissão é o monitoramento da corrente de fuga para mostrar perdas nas propriedades isolantes do material. Um sistema de detecção de corrente de fuga eficiente, que já possui resultados consolidados através de testes experimentais de campo, pode ser aperfeiçoado, melhorado em processamento, correção de falhas no circuito e adaptações a dispositivos e componentes de melhor desempenho, mantendo sempre as características de custo baixo, baixo consumo de energia e fácil transporte para que se consolide na área de monitoramento e manutenção de isoladores elétricos de linhas de transmissão. A necessidade imediata é que esse dispositivo possa prover suporte à comunicação sem fio, melhorando a coleta de informações sobre o meio monitorado.

Abstract

One way to know when to perform maintenance of insulators of transmission lines is monitoring the leakage current to show losses in the insulating properties. A system for detecting leakage current efficiency, which already has consolidated through experimental testing of field, can be refined, improved in processing, correction of faults in the circuit and adjust the devices and components for better performance, while maintaining the features of low cost, low power consumption and easy transport in order to establish the area of monitoring and maintenance of insulators electric transmission lines. The immediate need is this device can provide support for wireless communication, improving the collection of information about monitored environment.

Sumário

Índice de Figuras ix

Índice de Tabelas x

Listas de Símbolos e Siglas xi

1 Introdução 12

2 Microcontroladores 13

2.1 Arquiteturas dos microcontroladores 14

2.1.1 Arquitetura *CISC* 15

2.1.2 Arquitetura *RISC* 15

2.1.3 Arquitetura *Von Neumann* 17

2.1.4 Arquitetura *Havard* 17

2.2 Microcontroladores PIC 19

3 Sistema de detecção de corrente de fuga 21

3.1 Componentes do detector 21

3.1.1 Sensor óptico 22

3.1.2 Módulo de processamento 23

4 Desenvolvimento do projeto 25

4.1 Ferramentas utilizadas no projeto 26

4.1.1 MPLab® IDE 26

4.1.2 CCS C compile® 27

4.1.3 Proteus® 28

4.1.4 TraxMaker® 28

4.2 Alterações no Hardware 28

4.2.1 Microcontrolador PIC18F4550 _____ 29

4.2.2 Alterações no desenho da placa de circuito impresso. _____ 30

4.3 Alteração do código em linguagem C _____ 34

4.3.1 Diretivas de compilação Usadas _____ 35

4.3.2 Alteração no código fonte _____ 37

4.3.3 Integração do projeto utilizando o MPLab _____ 40

5 Conclusões e trabalhos futuros _____ 42

Bibliografia _____ 43

Apêndice A:Circuito elétrico adaptado do módulo de processamento. _____ 47

Apêndice B: Desenho da placa de circuito impresso adaptada. _____ 48

Apêndice C: Código fonte dos programas principal e contagem de níveis adaptados. _____ 49

Índice de Figuras

Figura 1: Organização interna dos microcontroladores 12

Figura 2: Gráfico da estatística de instruções de *Tanenbaum* 15

Figura 3: *Von Neumann*: compartilhamento de dados e instruções 16

Figura 4: Comparativo entre as arquiteturas *Havard e Von neumann* 17

Figura 5: Evolução da familia *PIC* 19

Figura 6: Sistema detector de corrente de fuga 21

Figura 7: *LED* conectado a cabeça do sensor óptico 22

Figura 8: Diagrama de blocos do modulo de processamento 23

Figura 9: Fluxo de projeto do trabalho 25

Figura 10: Encapsulamento *PDIP* de 40 pinos do *PIC18F4550* 28

Figura 11: Esquema elétrico das ligações do *PIC18F4550* 32

Figura 12: Ateração da pinagem na placa de circuito impresso 32

Figura 13: Interface da *IDE CCS C compiler* 33

Figura 14: Opções do *menu view CCS C compile* 34

Figura 15: Organização modular do código fonte do módulo de processamento 37

Figura 16: Opção de compilar no *CCS C compile* 39

Figura 17: mensagem de saída do compilador 40

Figura 18: Compilação do projeto no *MPLab* *41*

Índice de Tabelas

Tabela 1 - Comparativo entre as pinagens anterior e posterior a adaptação_____30

Lista de Símbolos e Siglas

Em Ordem Alfabética:

- *ALU: arithmetic logic unit*
- *ANSI: American National Standards Institute*
- *BNC: Bayonet Neill-Concelman conector*
- *CCS: Custom Computer Services*
- *CISC: Complex Instruction Set Computer*
- *CPU*: *Central Processing Unit*
- *EEPROM: Electrically-Erasable Programmable Read-Only Memory*
- *FC: Fiber Optic Connector*
- *GPIB: General Purpose Interface Bus*
- *I2C: Inter-Intergrated Circuit*
- *LCD: Liquid Crystal Display*
- *LED: Light-Emitting Diode*
- *PCB: Printed Circuit Board*
- *PIC: Peripherical Interface Controller*
- *PWM: Pulse Width Modulation*
- *RISC: Reduced Instruction Set Computer*
- *ROM: Read Only Memory*
- *UART: Universal Synchronous Receiver/Transmitter*
- *USB: Universal Serial Bus*

1 Introdução

Sistemas de monitoramento de corrente de fuga se mostram eficientes para direcionar a manutenção de cadeias de isoladores elétricos de linhas de transmissão de energia. Neste momento existe um sistema consolidado e de comprovada eficiência, testada experimentalmente, em uso e monitorando a corrente de fuga em seis diferentes localidades da região Nordeste do Brasil. Porém há a necessidade de atualizar o hardware e o software deste sistema para que ele continue competitivo e se mantenha como opção de uso na aplicação.

Para atingir os objetivos do trabalho, deve-se entender o problema para o qual o sistema foi proposto, a arquitetura desenvolvida para o dispositivo e explorar pontos que possam ser melhorados dentre suas funcionalidades, observando as metas propostas por seus desenvolvedores e também por pessoas que trabalham atualmente em sua melhoria.

Uma das funcionalidades imediatas que se busca oferecer nesse sistema é o suporte à comunicação sem fio. Para tal o sistema requer uma atualização de hardware e nessa adaptação, devem ser preservadas a função de contagem de eventos de descarga de corrente de fuga e seu respectivo armazenamento para a coleta de dados e manter a compatibilidade com os sistemas existentes para facilitar a transição de tecnologia, e como objetivos podemos descrever três etapas do trabalho: Adaptar o Hardware existente, modificar o software atual e alterar o desenho da placa de circuito impresso do dispositivo.

2 Microcontroladores

Os microcontroladores [1] são dispositivos eletrônicos dotados de processamento, memória e interface de entrada e saída, exatamente como os computadores, porém com todos esses recursos agregados em uma única pastilha de circuito integrado (*CI*) [2]. Dentro dele há um processador programável, e inserido no mesmo *CI* com sua memória para armazenamento de instruções e dados, suas interfaces de entrada e saída (pinos) para a interação com o meio externo. São produzidos na forma de circuitos integrados, com diversos tipos e formatos de encapsulamento e com pinagens variáveis de acordo com as necessidades de aplicação. Largamente utilizados na automação industrial e na construção de aparelhos eletrônicos de uso geral tornando-se um dos componentes eletrônicos de maior sucesso em aplicação, por terem reduzido tamanho, custo e consumo de energia, se comparados à forma de utilização de microprocessadores convencionais, aliados a facilidade de desenvolvimento de aplicações, juntamente com o seu baixo custo, os microcontroladores são uma alternativa eficiente para controlar muitos processos e aplicações.

Destacam-se vantagens da integração, junto da Unidade Central de Processamento (*CPU*) [3], dos circuitos necessários aos sistemas de controle e dessa forma, não é preciso um processador sofisticado, com uma grande capacidade de processamento. Oferece um conjunto de instruções simples, que geram programas pequenos e de rápida execução, ou seja, instruções de tamanho

reduzido. Além disso, é interessante que a Unidade Central de Processamento possa efetuar expressões *booleanas* [4], para facilitar a construção de sistemas de controle. É preciso ainda oferecer uma forma simples de se interfacear com outros periféricos que venham a ser adicionados, com pinos específicos para entrada de *clock [5]*, sinais analógicos, digitais e de controle do próprio dispositivo. A organização de um microcontrolador é descrita na Figura 1.

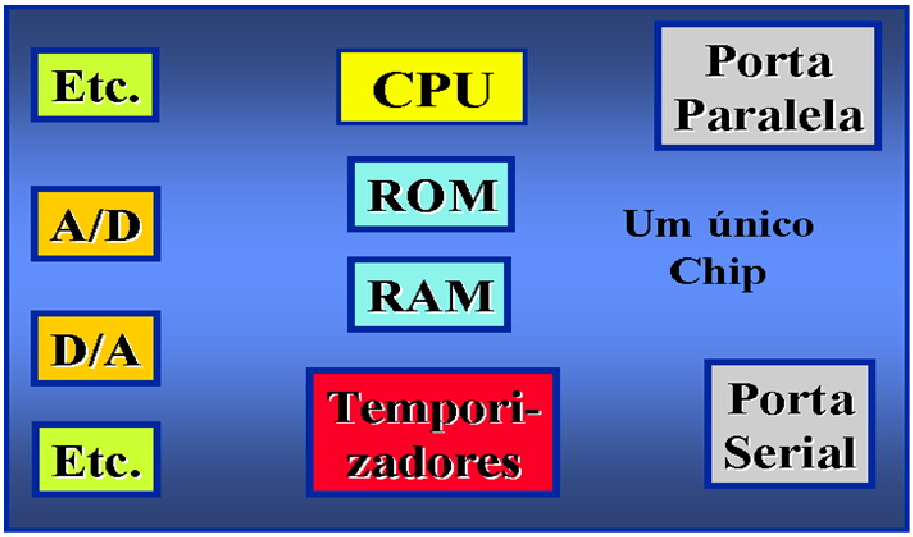


Figura 1. Organização interna dos microcontroladores.

2.1 Arquiteturas dos microcontroladores

Os Microcontroladores são organizados internamente de várias formas e isso definirá aspectos em seu desempenho, capacidade de armazenamento, consumo e programação. Várias formas de organização são utilizadas e implementadas pelos fabricantes, aplicando conceitos consolidados de computação.

2.1.1 Arquitetura *CISC*

Uma arquitetura *CISC* (“*Complex Instruction Set Computer*”) [6] como tendo instruções mais complexas objetivando diminuir o número de instruções que um programa necessita para sua implementação.

O número de ciclos por instruções pode aumentar assim como o próprio tempo de relógio. Um exemplo de aplicação seria a família do microcontrolador 8051 e seus derivados, originalmente da Intel e com pouco mais de 100 instruções.

2.1.2 Arquitetura *RISC*

Uma arquitetura *RISC* (“*Reduced instruction Set Computer*” ou computador com conjunto de instruções reduzido) [7] se caracteriza pela redução do tempo médio de execução das instruções de máquina ocasionando menor número de ciclos por instruções, porém o número de instruções executadas por programa aumenta. Um exemplo é a família de microcontroladores *PIC* [8], como exemplo os microcontroladores da família 16f que possuem pouco mais de 30 instruções.

Os elementos básicos de uma arquitetura *RISC* são: grande número de registradores de propósito geral ou uso de tecnologias de compilação na otimização do uso de registradores, conjunto simples e limitado (reduzido) de instruções e enfoque na utilização de pipeline de instruções.

A arquitetura *RISC* possui regras: Sacrifique tudo para reduzir o tempo de ciclo da via de dados. Seguem os conceitos da filosofia básica:

1. Analisar as aplicações para encontrar as operações chave e elaborar uma

máquina ótima para as realizadas pelos programas alvo. Segundo *Tanenbaum* [9] quase 50% das instruções de um programa são comandos de atribuição, 23% de comandos de seleção, 15% de chamadas a funções, 6% de laços de repetição e, 10%, de outros comandos, distribuição exibida na figura 2.

2. Projetar um caminho dados que seja ótimo para as operações chave. Esse é o coração de uma máquina *RISC*. Composto pelos registradores, pela *ULA* e pelos barramento internos que os conectam. Em um único ciclo, os operandos devem ser lidos dos registradores, operados pela *ULA* e o resultado armazenado em um registrador.
3. Projetar as instruções que executem as operações chaves, estas devem utilizar de forma ótima os caminhos de dados, ou seja, cada instrução deve utilizar um único caminho de dados.
4. Adicionar novas instruções somente se elas não diminuírem a velocidades da máquina.
5. Repetir este processo para outras Instruções por ciclo do caminho de dados.

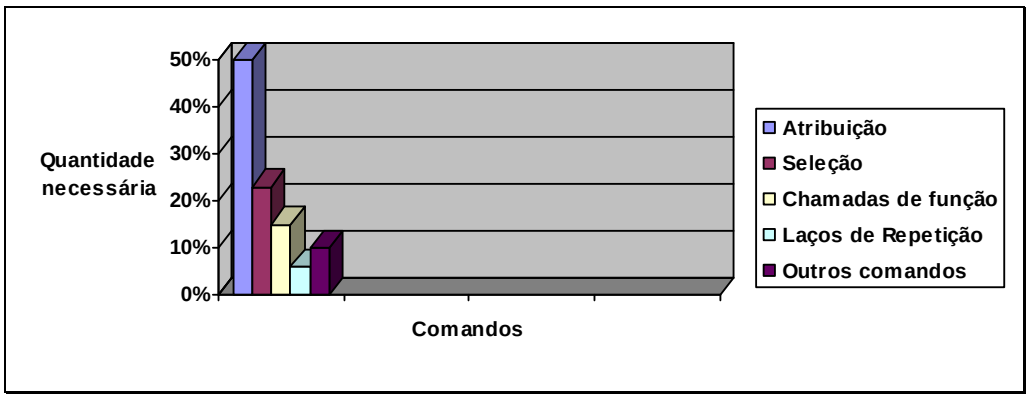


Figura 2. Gráfico da estatística de instruções de *Tanenbaum*.

2.1.3 Arquitetura *Von Neumann*

A Arquitetura proposta por *Jonh Von Neumann* define uma máquina que reúne os seguintes componentes: uma memória, uma unidade aritmética e lógica (*ALU*), uma unidade central de processamento (*CPU*), composta por diversos registradores, e uma Unidade de Controle (*UC*), com finalidade equivalente a tabela de controle da Máquina de Turing universal [10]: efetuar a busca de um programa na memória, instrução por instrução, a fim de executá-lo sobre os dados de entrada, compartilhando a memória com dados e programa como mostrado na Figura 3.

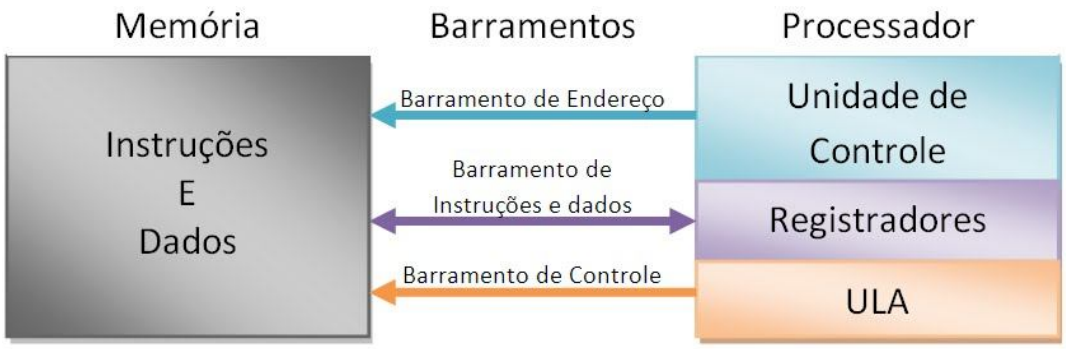


Figura 3. Von Neumann: compartilhamento de dados e instruções.

2.1.4 Arquitetura *Havard*

A Arquitetura de *Harvard* baseia-se em um conceito mais recente que a de *Von-Neumann*, tendo vindo da necessidade de por desempenho em processamento. É uma arquitetura computacional que possui como principal característica duas memórias diferentes e independentes em termos de barramento e ligação ao processador, consequentemente separando os barramentos de dados do

barramento de programa, permitindo que um processador possa acessar as duas simultaneamente, obtendo um desempenho melhor no acesso a recursos, pela capacidade de poder buscar uma nova instrução enquanto executa a anterior. Essa dupla ligação às memórias de dados e programa (código), permite que o processador leia uma instrução ao mesmo tempo que faz um acesso à memória de dados. A arquitetura *Havard* também possui um repertório com menos instruções que a de *Von-Neumann*, e essas são executadas apenas num único ciclo de relógio.

Os microcontroladores com arquitetura *Havard* são também conhecidos como "microcontroladores *RISC*" (Computador com Conjunto Reduzido de Instruções), e os microcontroladores com uma arquitetura Von-Neumann, de "microcontroladores *CISC*" (Computador com um Conjunto Complexo de Instruções) e a Figura 4 mostra um comparativo entre as duas arquiteturas.

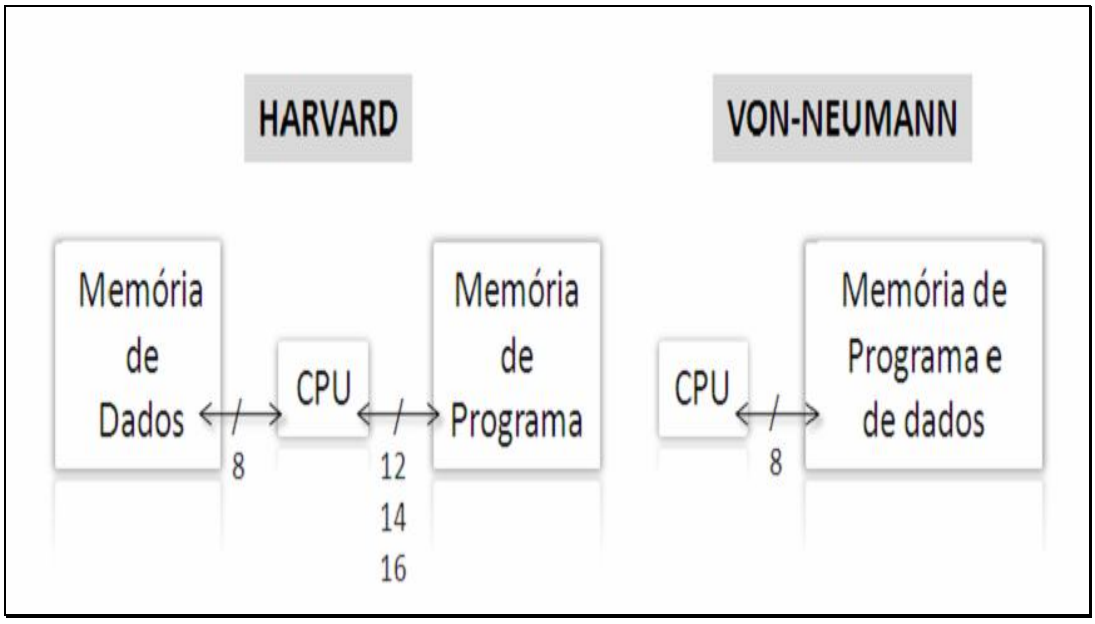


Figura 4. Comparativo entre as arquiteturas Havard e Von neumann.

2.2 Microcontroladores PIC

Os PIC são uma família de microcontrolador é tipo *RISC* fabricados pela *Microchip Technology Inc.* e derivados do *PIC1650*, originalmente desenvolvido pela divisão de microeletrônica da *General Instrument*.

O nome atual não é um acrônimo, na verdade significa *PICmicro* ou ainda utiliza-se a denominação *Peripheral Interface Controller* (controlador de interface periférico).

O *PIC* original foi projetado para ser usado com a nova *CPU* de 16 bits *CP16000*. Sendo em geral uma boa *CPU*, porém com muitos problemas de compatibilidade com dispositivo de entrada e saída (*E/S*), levando a empresa a desenvolver o *PIC* de oito bits em 1975 para corrigir essa falha. O *PIC* utilizava *microcódigo* [11] simples e armazenado em *ROM* [12] para realizar estas tarefas, se trata de um desenho RISC que executa uma instrução a cada quatro ciclos do oscilador. Em 1985 a divisão de microeletrônica da *General Instrument* separa-se como companhia independente que é incorporada como filial (em 14 de dezembro de 1987 muda o nome para *Microchip Technology* e em 1989 é adquirida por um grupo de investidores) e o novo proprietário cancelou quase todos os desenvolvimentos, que para essas datas a maioria estavam ultrapassados. O *PIC*, no entanto, evoluiu com *EPROM* [13] para conseguir um controlador de canal programável. Atualmente, Os Microcontroladores *PIC* vêm com vários periféricos incluídos: módulos de comunicação série, *UARTs*, núcleos de controle de motores, memória de programa que pode chegar a 32 *Kilobytes*. A Figura 5 mostra a evolução da família PIC.

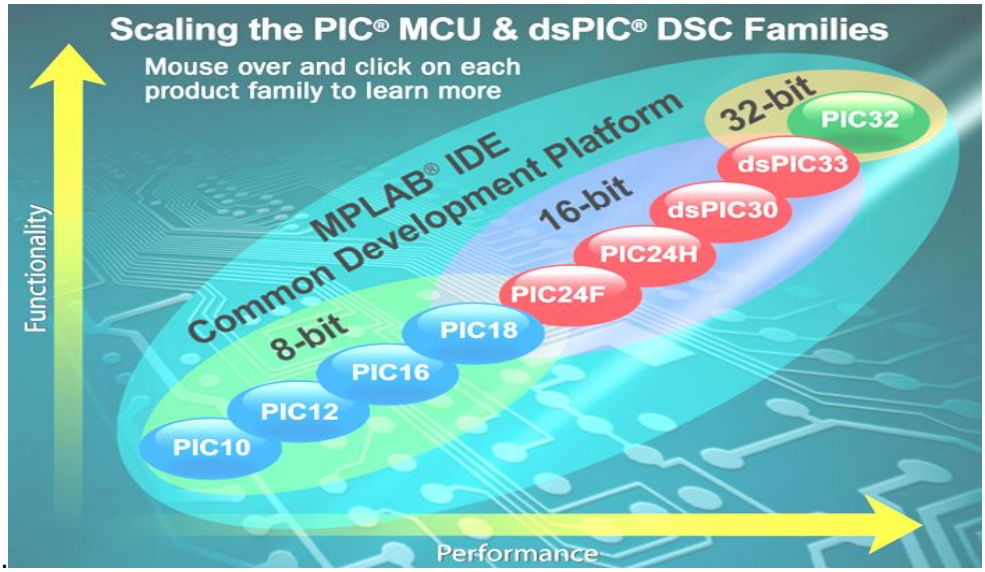


Figura 5. Evolução da família PIC (extraído de [8]).

3 Sistema de detecção de corrente de fuga

A corrente de fuga de uma cadeia de isoladores de linhas de transmissão de energia elétrica é a forma pela qual o sistema a ser adaptado nos mostra se há perdas nas propriedades dielétricas do isolador. O sistema a ser adaptado já possui comprovação experimental [14] e os estudos de sua construção e funcionamento serão usados para a execução do projeto de adaptação.

Nesse sentido é preciso entender como a informação sobre essas descargas de corrente são coletadas e levadas ao módulo de processamento onde é armazenado o número de eventos de descarga por níveis de tensão, umidade e temperatura, fatores que determinam a forma pelo qual o sistema fará o processamento de informações.

3.1 Componentes do detector

O detector pode ser dividido em duas partes principais: o sensor óptico, responsável pela detecção das descaras de corrente de fuga e o módulo de processamento que faz o armazenamento e classificação desses sinais.

O arranjo desse sistema é estruturado de acordo com a Figura 6, onde o enlace que comunica o sensor e o módulo de processamento é uma fibra óptica e mostra as etapas de tratamento da informação a ser analisada, a interface GPIB (osciloscópio) e a porta serial para acesso ao computador.

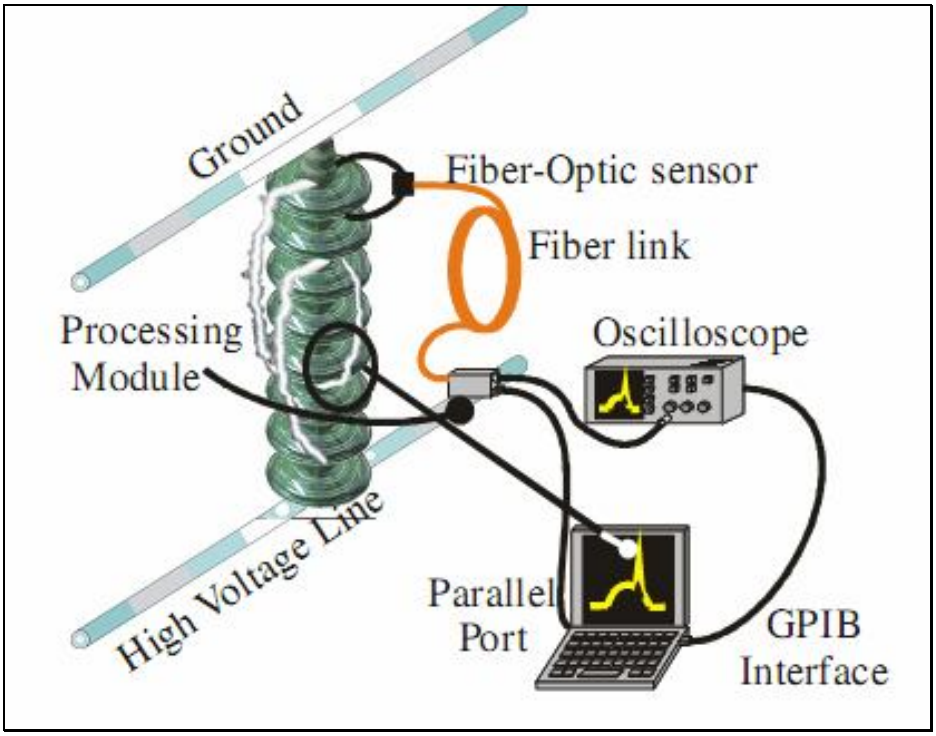


Figura 6. Sistema detector de corrente de fuga (retirado de [14]).

3.1.1 Sensor óptico

O sensor Óptico do sistema é composto por dois terminais elétricos, que por sua vez são conectados o mais próximo possível do potencial de referência (terra) na torre de transmissão. Essa forma de instalação visa facilitar a manutenção e o manuseio do equipamento e também a segurança de quem for colocá-lo no local.

O sensor compreende um diodo emissor de luz (*LED*), que opera em uma região de comprimento de onda próximo a um µm, colado a uma fibra óptica *multimodo*. Os terminais elétricos da cabeça do sensor estão interconectados a um diodo *shunt,* com o catodo do *LED* conectado ao anodo do diodo e vice-versa de forma a proteger eletricamente o *LED* de polarização reversa em seus terminais. O LED conectado a cabeça do sensor óptico é exibido na Figura 7.

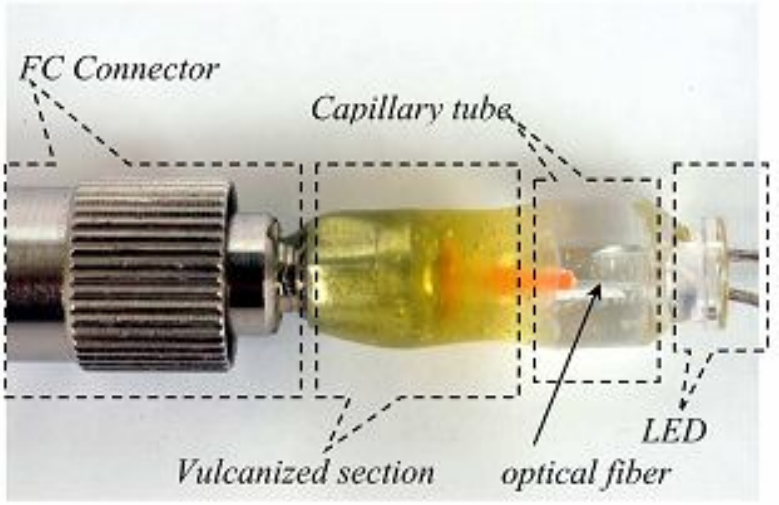


Figura 7. LED conectado a cabeça do sensor óptico (retirado de [14]).

3.1.2 Módulo de processamento

O módulo de processamento foi desenvolvido com a finalidade de detectar, amplificar e armazenar. O sinal de corrente de fuga captado pelo sensor óptico é transmitido pela fibra óptica e recebido pelo módulo de processamento. Este módulo possui um conector FC fêmea utilizado para conexões ópticas, uma interface BNC para monitoramento de sinais em tempo real por um osciloscópio e uma porta serial para prover comunicação com um computador pessoal.

A organização desse módulo é descrita na figura 8, com os estágios de amplificação e comparação que definem os níveis de corrente de fuga a serem contados em valores de 5, 10, 20 e 40 *mA* respectivamente, e processamento que é realizado por um microcontrolador *PIC16f877* [15] programado em *linguagem C* [16], e uma porta serial para a conexão com um computador.

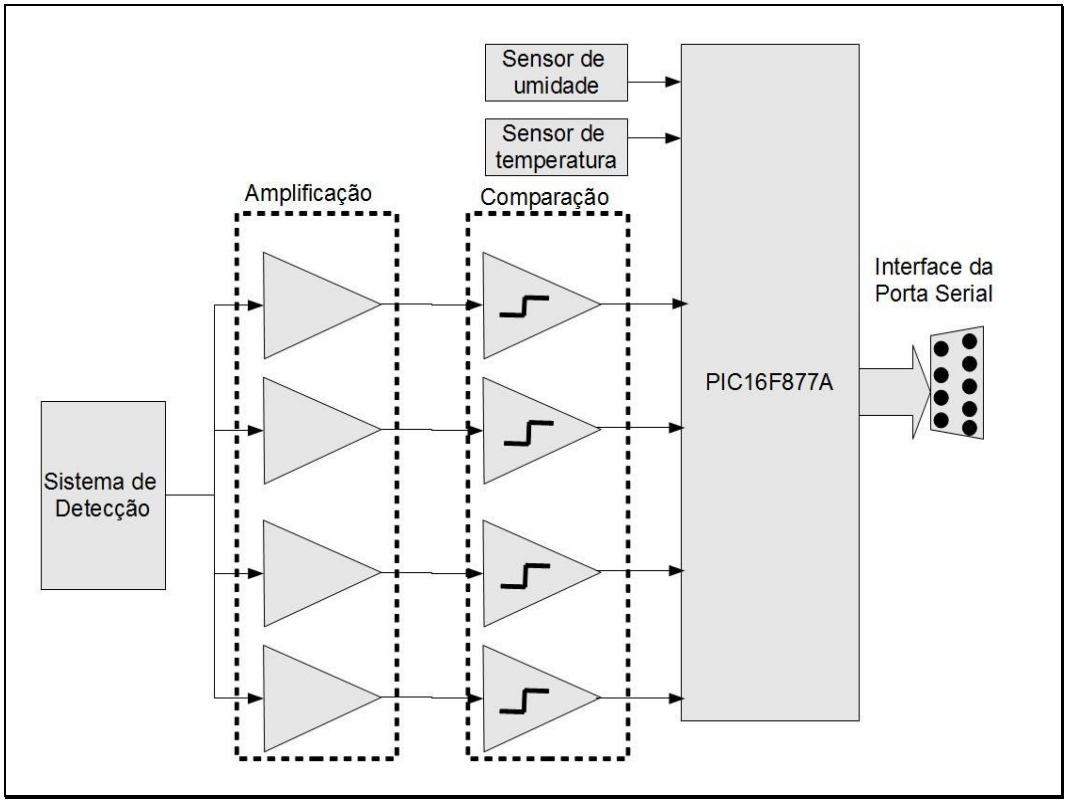


Figura 8. Diagrama de blocos do modulo de processamento.

4 Desenvolvimento do projeto

O dispositivo detector de corrente de fuga em cadeia de isoladores já havia sido testado em linhas de transmissão do sistema CHESF [17], no nordeste brasileiro e com resultados comprovados nesses experimentos. Porém, para mantê-lo competitivo e atualizado a dar suporte a novas tecnologias, o presente trabalho propôs a alteração do modulo de processamento, implementada sobre o hardware do microcontrolador PIC16F877.

A necessidade de alterar o mínimo possível do hardware se baseia na premissa de não comprometer os resultados já alcançados pelos desenvolvedores e pesquisadores que atualmente trabalham no dispositivo atual, grupo ao qual se destina a contribuição do trabalho. Ao fazer modificações em sistemas computacionais, deve-se preservar a funcionalidade ao máximo possível e preservando as características originais de projeto. Para substituir o atual microcontrolador, foi escolhido o microcontrolador PIC18f4550, que tem arquitetura semelhante, mesmo número de pinos, suporta programação em C, mas oferece mais recursos e dará suporte a uma implementação de comunicação sem fio, objetivo traçado pelo trabalho. A figura 9 mostra um diagrama de fluxo de projeto aplicado neste trabalho.

Seguindo um fluxo ordenado de projeto, colhendo informações e orientações dos profissionais que desenvolveram e trabalham na ferramenta, este projeto busca não alterar as funcionalidades do dispositivo, mas sim alterar partes de sua estrutura a fim de prover mais funcionalidades que podem ser implementadas futuramente por pessoas que venham integrar o projeto e também fornecer suas contribuições.

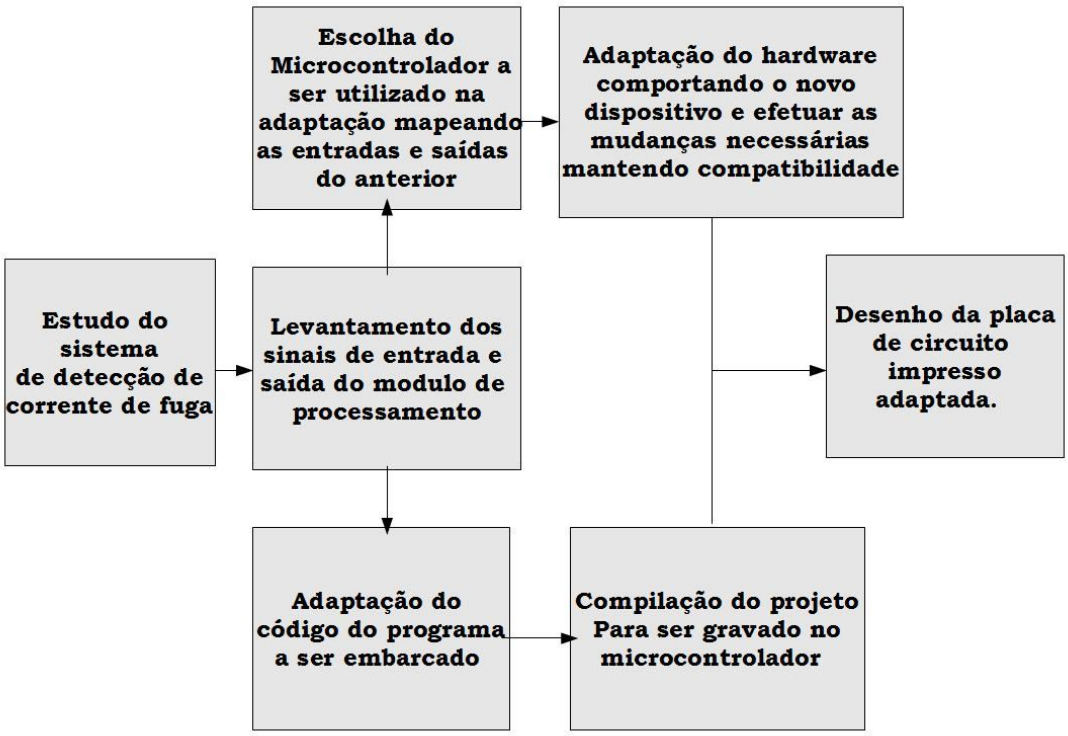


Figura 9. Fluxo de projeto do trabalho.

4.1 Ferramentas utilizadas no projeto

Para desenvolver as mudanças no projeto necessitamos de ferramentas para manusear as partes do projeto: Uma *IDE* (*Integrated Development Environment*) [18] para manipulação do código em linguagem C e sua respectiva integração com o microcontrolador, e uma ferramenta para desenho do circuito elétrico e da placa de circuito impresso para efetuar a troca e adaptação do microcontrolador.

4.1.1 *MPLab*® IDE

A *Microchip* [19], empresa fabricante da família de microcontroladores PIC disponibiliza a IDE *MPLab* para a programação e gravação dos códigos nos chips.

A ferramenta oferece uma série de funcionalidades, como geração do código fonte, compilação e teste do chip escolhido pelo projetista, em um ambiente amigável e de fácil compreensão.

4.1.2 CCS C compile®

O *CCS* (*CUSTOM Computer Service* – Sigla utilizada pela empresa fabricante do compilador) *C Compiler* [20] foi o compilador utilizado no trabalho proposto para o desenvolvimento da biblioteca de funções, essa ferramenta possui uma interface bastante amigável, vale destacar ainda que, ele dá suporte a vários *drivers* os quais podem ser incluídos e usados no programa.

- Configuração e inicialização de temporizadores;
- *PWM (Pulse-Width Modulation – Modulação por Largura de Pulso)* [21];
- *I2C (Inter-Intergrated Circuit – Sigla que referencia o barramento serial* Multi-mestre *desenvolvido pela Philips*) [22];
- Leitura e escrita da memória *EEPROM (Electrically-Erasable Programmable Read-Only Memory – Memória somente de leitura programável e apagável eletricamente)* [23];
- Leitura e escrita de pinos de entrada e saída.

Além disso, o compilador fornece funções para conversão de valores. analógicos, para comunicação via *RS232* [24], para comunicação com display de LCD (*Liquid Crystal Display – Display de Cristal Líquido)* [25], além é claro de funções da *linguagem C* no padrão *ANSI*. Além dessas funcionalidades, a capacidade de integração com o *MPLab* é um fator importante na sua escolha para

que seus projetos não necessitem de nenhum tipo de exportação para serem inseridos no projeto de carregamento de código do PIC.

4.1.3 *Proteus®*

Ferramenta disponibilizada pela *Labcenter eletrônics*, O *Proteus* [26] é utilizado para o desenho de circuitos analógicos e digitais, esquemas elétricos e placas de circuitos impressos. Possibilita a modelagem e simulação de circuitos e vêm com vários dispositivos comerciais em seus repositórios.

4.1.4 *TraxMaker®*

O *TraxMaker* [27] é uma ferramenta para a confecção de placas de circuito impresso (PCB), vinculada ao pacote *circuitMaker* 2000. A sua escolha mantém a compatibilidade com a versão anterior da placa de circuito, e também facilita a geração de arquivos para a confecção profissional de placas de circuitos impressos.

4.2 Alterações no Hardware

A proposta inicial do projeto em alterar o microcontrolador *PIC16f877A* não foi ao acaso. Como há a necessidade de novas funcionalidades, dentre elas oferecer o suporte a comunicação sem fio era preciso escolher um microcontrolador que atendesse a alguns requisitos de projeto:

1. Suporte a programação em *C*
2. Ter mesmo número de pinos, mantendo a placa igual.

3. Trabalhar com palavra de dados de mesmo tamanho (oito bits).
4. Prover suporte a comunicação sem fio.
5. Prover suporte a USB

4.2.1 Microcontrolador PIC18F4550

O microcontrolador escolhido para substituir o *PIC16F877A*, foi o *PIC18F4550[28],* mostrado na figura 10 com o encapsulamento *PDIP* 40 pinos, pertencente à família *18F*. Este dispositivo atende as necessidades de projeto e vem se destacando entre os projetistas pelas funcionalidades e por manter várias características da família *16f*.

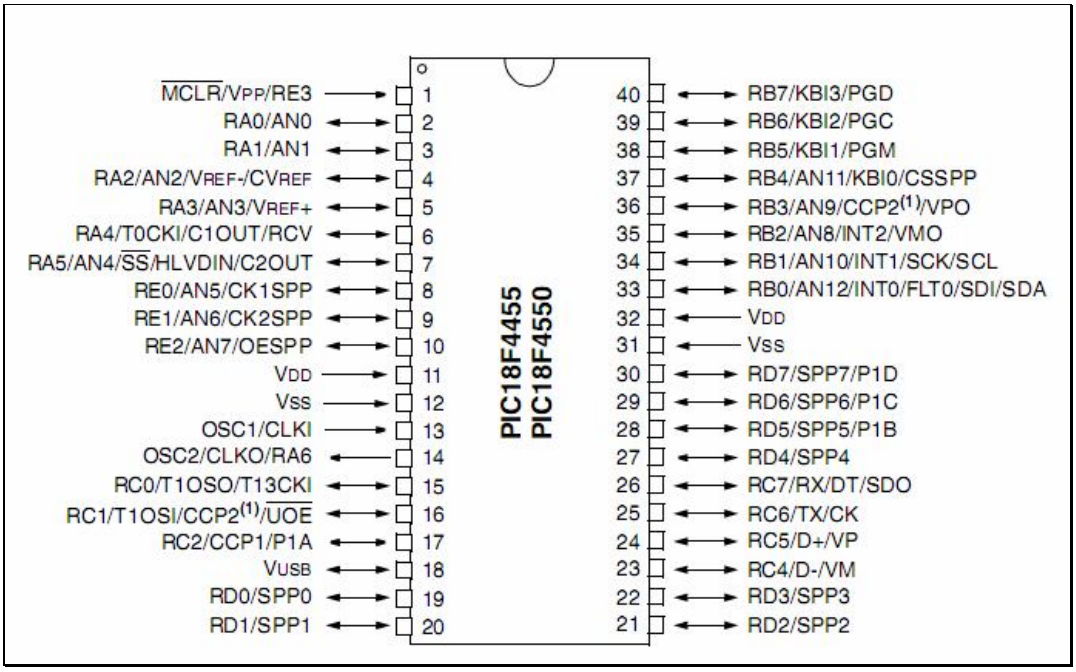


Figura 10. Encapsulamento PDIP de 40 pinos do PIC18F4550.

4.2.2 Alterações no desenho da placa de circuito impresso.

Utilizando o Software *TraxMaker*, foi efetuada a troca entre os microcontroladores e alterações das ligações entre suas entradas e saídas. A Tabela 1 mostra uma tabela com as entradas e saídas do sistema a serem adaptadas.

Tabela 1 - comparativo entre as pinagens anterior e posterior a adaptação.

Entrada/Saída	**Pino 16F877A**	**Pino 18F4550**	**Descrição**
Umidade	Pino 2	Pino 2	Entrada Analógica A0
Temperatura	Pino 3	Pino 3	Entrada Analógica A1
N1	Pino 15	Pino 15	Interrupção Timer 0
N2	Pino 6	Pino 6	Interrupção Timer 0
N3	Pino 33	Pino 33	Interrupção Externa 1
N4	Pino 38	Pino 34	Interrupção Externa 1
Tx	Pino 25	Pino 25	Transmissor (serial)
Rx	Pino 26	Pino 26	Receptor (serial)
Reset	Pino 1	Pino 1	Reinicialização
Oscilador1	Pino 13	Pino 13	Clock externo
Oscilador2	Pino 14	Pino 14	Clock externo
Pico máximo	Pino 4	Pino 4	Entrada Analógica A2
Zerar Pico Máximo	Pino 24	Pino 24	Zerar Pico Máximo

Nesta direção, percebe-se que não só o numero de pinos é igual, como também a compatibilidade dos pinos a serem trocados que praticamente tem as mesmas especificações, e poderiam ser colocadas todas as entradas e saídas nos mesmos pinos. Então com as definições dos sinais de entrada, o qual tem um único pino trocado no esquema elétrico do *Proteus* na Figura 11

e o desenho da placa de circuito impresso adaptado através do *TraxMaker* conforme Figura 12.

1. Umidade – Sinal analógico enviado pelo sensor de umidade *HIH3610* [29], está conectado a entrada *AN0*, pino.
   2 em ambos microcontroladores.
2. Temperatura - Sinal analógico enviado pelo sensor de temperatura *LM35* [30], está conectado a entrada *AN1*, pino 3 em ambos microcontroladores.
3. N1 – Nível de corrente de fuga 5mA é conectado no pino 15 em ambos os microcontroladores, correspondente a interrupção de *Timer 0*.
4. N2 – Nível de corrente de fuga 10mA é conectado no pino 6 em ambos os microcontroladores, correspondente a interrupção de *Timer 1*.
5. N3 – Nível de corrente de fuga 20mA é conectado no pino 33 em ambos os microcontroladores, correspondente a interrupção *externa 0*.
6. N4 – Nível de corrente de fuga 40mA, é conectado no pino 38 do *PIC16F8777A* correspondente ao bit *RB5* do *portB*, era monitorado pela interrupção de mudança de estado *(RB4-RB7),* foi inserido no pino 34 do *PIC18F4550* interrupção *externa 1*.

O PIC *18F4550* possui muitos recursos a mais que o *PIC16F877A*, e assim podemos utilizar as interrupções externas que ele oferece, ao todo 3 (*EXT0,EXT1,EXT2*). No projeto anterior, o Nível 4 de corrente (40mA) foi

conectado ao pino *RB5* do *portB*, utilizando a interrupção por mudança de estado no *portB* (*RB4-RB7*), por terem sido usados os dois *timers (timer0, timer1)* para os níveis N1 e N2 e a única interrupção externa que possui (*EXT0*) para o nível N3. Na adaptação, foi aproveitada a entrada *EXT1* do pino 38 para que o nível N4 também fosse coletado como interrupção externa, gerando melhorias na captura do sinal, já que não haverá um tratamento do sinal no *portB* antes de ser armazenado e consequentemente no código em *C* a ser atualizado em função dessas alterações, lembrando que há funções de tratamento de interrupções externas prontas do compilador a ser utilizado.

Essas mudanças tornam o projeto bem flexível, aproveitando a maior parte das configurações do microcontrolador anterior no atual por ter muitas características que igualam ou até melhoram o hardware, possuindo arquitetura similar, trará a reutilização de boa parte do código do programa embarcado no *PIC18F4550* sendo de grande importância ao desenvolvimento do projeto, ressaltando os créditos de autoria do software do módulo de processamento dos desenvolvedores que trabalharam no sistema detector de corrente de fuga.

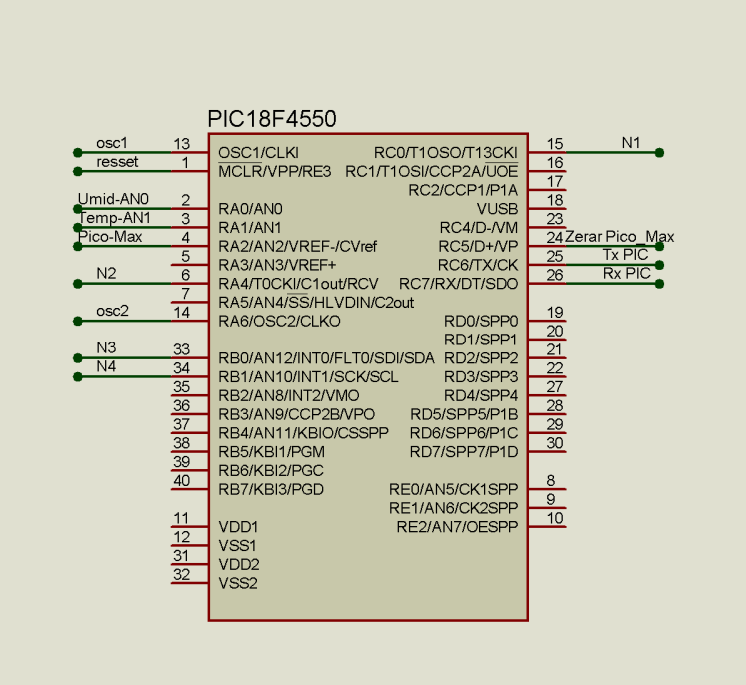


Figura 11. Esquema elétrico das ligações do PIC18F4550.

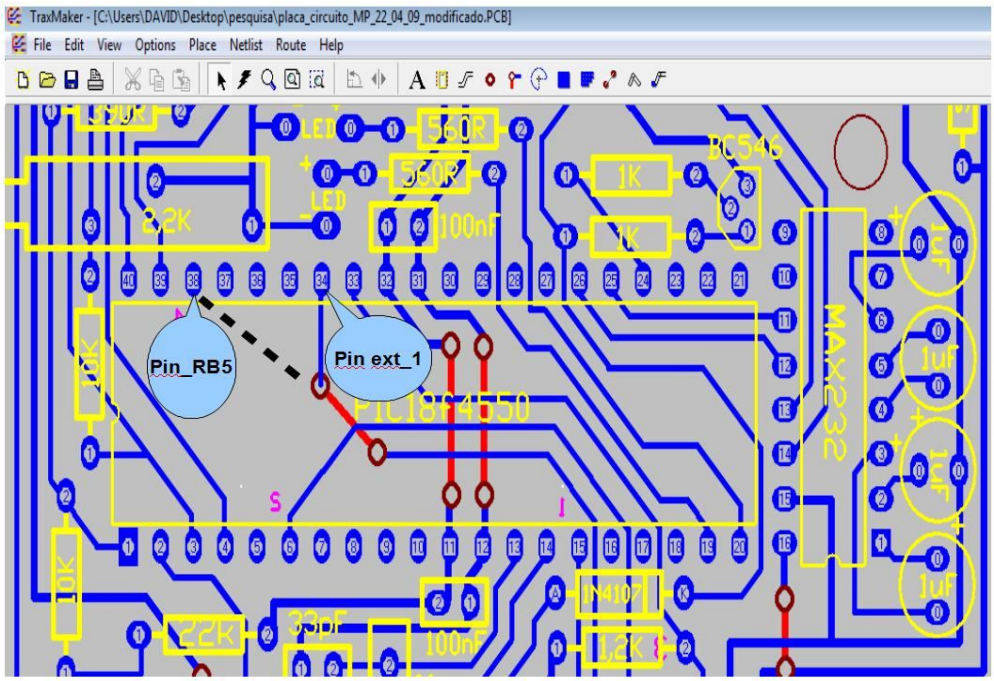


Figura 12. Alteração da pinagem na placa de circuito impresso.

4.3 Alteração do código em linguagem C

Uma das características positivas do módulo de processamento é que sua programação foi desenvolvida em linguagem *C*, que traz como vantagem a abstração do *assembly [31]* do microcontrolador, único para cada chip, para uma linguagem de alto nível e consolidada no mercado. Como foi uma estratégia de projeto positiva, o uso linguagem C continuará na adaptação do software, mantendo a compatibilidade com o sistema anterior para que não haja nenhuma preocupação com a passagem do assembly do *PIC16F877A* para o *PIC18F4550*. Utiliza-se a *IDE CCS C compiler,* com detalhes de sua interface exibida na Figura 13 para a programação do *PIC*, utilizando como compilador o *PCW* para posteriormente importar os arquivos de extensão .c gerados na compilação do projeto pelo *MPLab*.

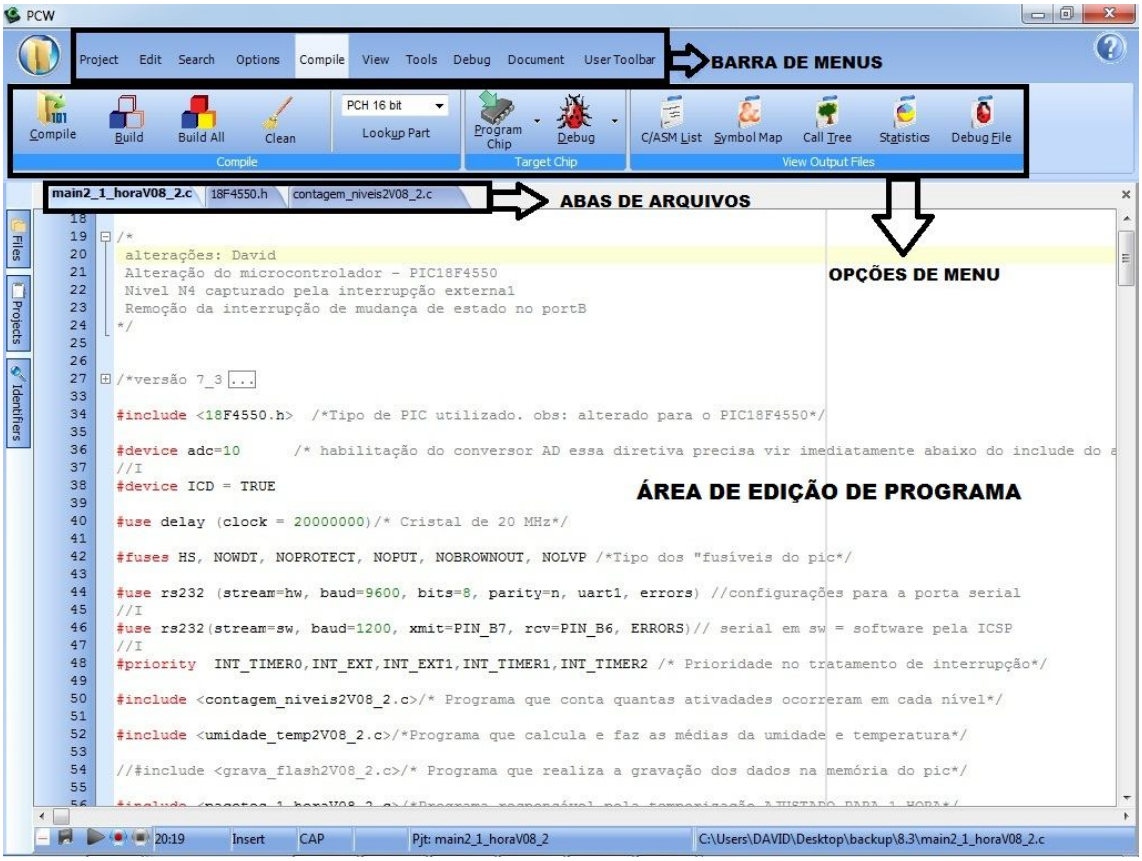


Figura 13. Interface da *IDE CCS C compiler.*

4.3.1 Diretivas de compilação Usadas

Os compiladores de linguagens de programação possuem conjuntos de comandos internos que o auxiliam a traduzir o código para a arquitetura correspondente e o *PCW*, integrante do *CCS* possui inúmeras diretivas que dão suporte a recursos e funcionalidades de cada microcontrolador da família *PIC*.

No projeto, foi preciso utilizar as diretivas compatíveis com o *PIC18F877A* e modificar as que entravam em conflito ou habilitavam novos recursos, seguindo as diretivas básicas de projeto. O *CCS* tem uma função na interface que nos mostra os fusíveis e interrupções válidas para cada microcontrolador da família PIC, no menu *view* conforme mostrado na Figura 14.

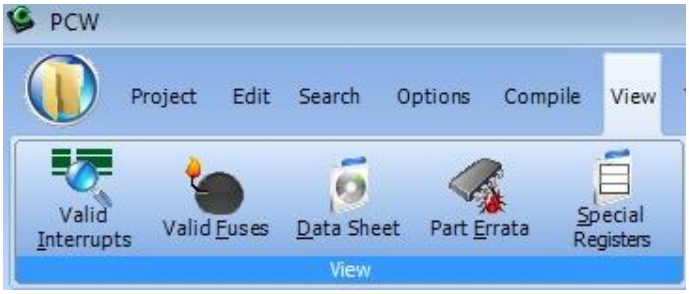


Figura 14. Opções do menu view *CCS C compile.*

Abaixo temos uma descrição das principais diretivas usadas pelo projeto no programa principal *main2_1_horaV08_2.c* e quais precisaram ser adaptadas para comportar o microcontrolador *PIC18F4550*.

- *#include <18F4550.h>* : A diretiva Include, serve para incluir durante a compilação um arquivo auxiliar ou de cabeçalho. Neste caso, foi incluído o arquivo da plataforma adaptada. Essa diretiva também é usada para chamar os arquivos auxiliares do código do programa: *contagem_niveis2V08_2.c, umidade_temp2V08_2.c,grava_flash2V08_2.c, pacotes_1_horaV08_2.c,*

temporizaçãograva_flash3,comunica_com_central1.c,pacotes.c, transmissão_serial_30min.c.

- *#device adc=10* . A diretiva *device* obriga o compilador a considerar o dispositivo citado. Neste caso, define em 10 *bits* a *string* retornada pela função *read_adc()* do conversor *A/D* do *PIC*. Essa mesma diretiva também é usada no projeto para indicar o gravador *ICD* da microchip.

- *#use delay (clock = 20000000)*: Esta diretiva especifica o clock de entrada do microcontrolador. Apesar do PIC18F4550 operar em freqüências de até 48Mhz, o *clock* foi mantido em 20Mhz para aproveitar o circuito do oscilador existente.

- *#fuses HS, NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP*: Os fusíveis do PIC, nada mais são do que as palavras de configuração do microcontrolador. Neste caso, está habilitado o oscilador *HS* e desabilitados o *watchdog*, proteção de código, temporizador de *power-up*,*reset* por queda de tensão e programação em baixa tensão. Essa diretiva manteve-se idêntica ao projeto atual.

- *#priority INT_TIMER0,INT_EXT,INT_EXT1,INT_TIMER1,INT_TIMER2:*

  Esta diretiva define a prioridade no tratamento das interrupções. No projeto, elas são responsáveis pela captura dos eventos dos níveis de corrente de fuga N1, N2, N3 e N4. A única alteração foi a substituição da

interrupção INT_RB(mudança de estado no portB dos pinos RB4-RB7) pela interrupção INT_EXT1 utilizada agora para o monitoramento do nível N4.

- #use rs232 (stream=hw, baud=9600, bits=8, parity=n, uart1, errors) e
  #use rs232(stream=sw, baud=1200, xmit=PIN_B7, rcv=PIN_B6, ERRORS:
  A diretiva use rs232 ativa o suporte a comunicação serial, onde têm as opções de configuração: *stream* (associa a interface rs232a hardware ou software), *baud* (velocidade de comunicação serial), bits (número de bits de dados).

4.3.2 Alteração no código fonte

A Concepção do código fonte do programa que controla o microcontrolador segue um modelo de divisão por módulos, onde cada parte tem um arquivo.c correspondente. A Figura 15 exemplifica como está organizada a hierarquia do projeto.

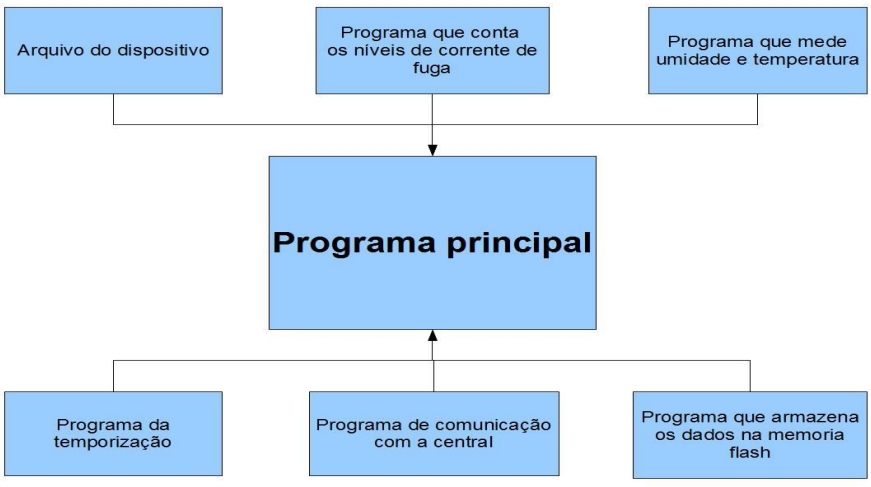


Figura 15. Organização modular do código fonte do módulo de processamento.

As modificações necessárias no software se restringem a alterar trechos de código que foram atingidos pela alteração do hardware: A troca do próprio microcontrolador, o programa principal e o que conta os níveis de corrente (Alteração do nível *N4* para responder a interrupção *EXT1*), e variáveis que tenham definições diferentes do *PIC18f4550*.

No programa principal, além de alterar as diretivas de configuração, foram necessárias algumas alterações de código, listadas a seguir:

1. O Parâmetro da função que escolhe os pinos de entradas analógicas *setup_adc_ports(AN0_AN1_AN2_AN3_AN4)* foi convertido para *setup_adc_ports(AN0_TO_AN4)*. Embora não alteradas no hardware, são especificadas de forma diferente no arquivo *pic18f4550.h.*
2. O parâmetro da função *enable_interrupts(int_rb)* foi substituído por *enable_interrupts(int_ext1)* pela troca da entrada *RB5* pela entrada *EXT1* para a contagem do nível N4.
3. Modificação de *EXT_INT_EDGE(H_TO_L) para EXT_INT_EDGE(0,H_TO_L) mais EXT_INT_EDGE(1,H_TO_L),* que seleciona a borda de sensibilidades da interrupção externa *(H_TO_L* borda de descida do sinal e *L_TO_H* na subida do sinal). Como o *PIC16F877A* só possui uma interrupção externa (*EXT0*), o argumento não precisa identificar qual a interrupção configurada. Como utilizamos no *PIC18F4550* duas interrupções externas (*EXT0* e *EXT1*), devemos inserir no código essa função duas vezes identificando não só a borda de sensibilidade, como também qual a interrupção que ela configura (0 = *EXT0*, 1 = *EXT1*).

No programa que efetua a contagem dos níveis de corrente, foi alterada a rotina de tratamento de interrupção que contava o nível N4, que era incrementado com uma função de tratamento da interrupção int_rb, sendo substituída por uma função que incrementava a variável n4 a cada interrupção in_ext1. A adaptação é mostrada nos trechos de código a seguir.

```
#int_rb //N4 incrementado pela interrupção de mudança de estado no portB
void estado_n4 ()  //função para tratamento da interrupção por mudança de
      estado no portB
            {
                  teste_n4 =1;
                  if(!input(pin_b5)&& teste_n4 ==1)
                  n4++;
                  teste_n4 = 0;
            }
```

```
#int_ext1 // N4 incrementado pela interrupção externa 1
void externa_n4 ()  //função para tratamento da interrupção externa1
{
    n4++;
}
```

Como o código fonte é organizado de forma modular, as alterações afetam apenas os programas que tiveram variáveis afetadas com a mudança de hardware,

mostrando que o sistema é robusto e pode ter suas partes adaptadas de forma independente.

Com as alterações feitas, resta apenas utilizar a opção *compiler* no menu de mesmo nome, conforme mostrado na Figura 16 até que não haja mais erros de sintaxe, semântica ou conflito de variáveis comuns no desenvolvimento de *software*. A Figura 17 mostra a saída do compilador após o processo.

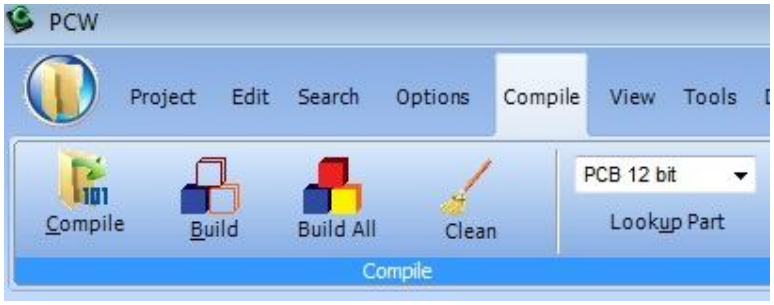


Figura 16. Opção de compilar no *CCS C compile.*

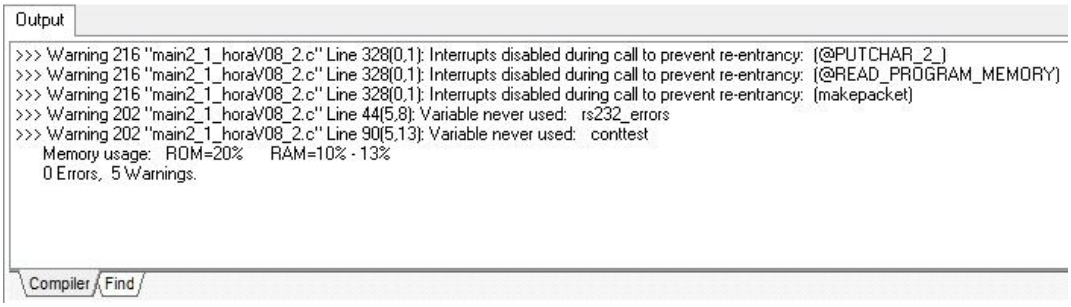


Figura 17. Mensagem de saída do compilador.

4.3.3 Integração do projeto utilizando o *MPLab*

Após as modificações no código do programa principal e dos auxiliares, inserção do arquivo de cabeçalho *PIC18F4550* através da diretiva *include* houve a compilação do projeto como um todo no *MPLab*. Após isso, deve-se escolher

um gravador adequado e carregar os arquivos compilados no Microcontrolador para posteriormente ser integrado a placa de circuito impresso (*PCB*). O processo é feito vinculando-se o *CCS C compile* ao *MPLab* para reconhecer o compilador *PCW* e em seguida com todos os programas principais e o *header* do microcontrolador *PIC18F4550*. Após esse procedimento, no *MPLab* selecionar no menu project a opção *Build all* para compilar todos os arquivos do projeto. Se nenhuma mensagem de erro for exibida, toda a compilação foi efetuada com sucesso.

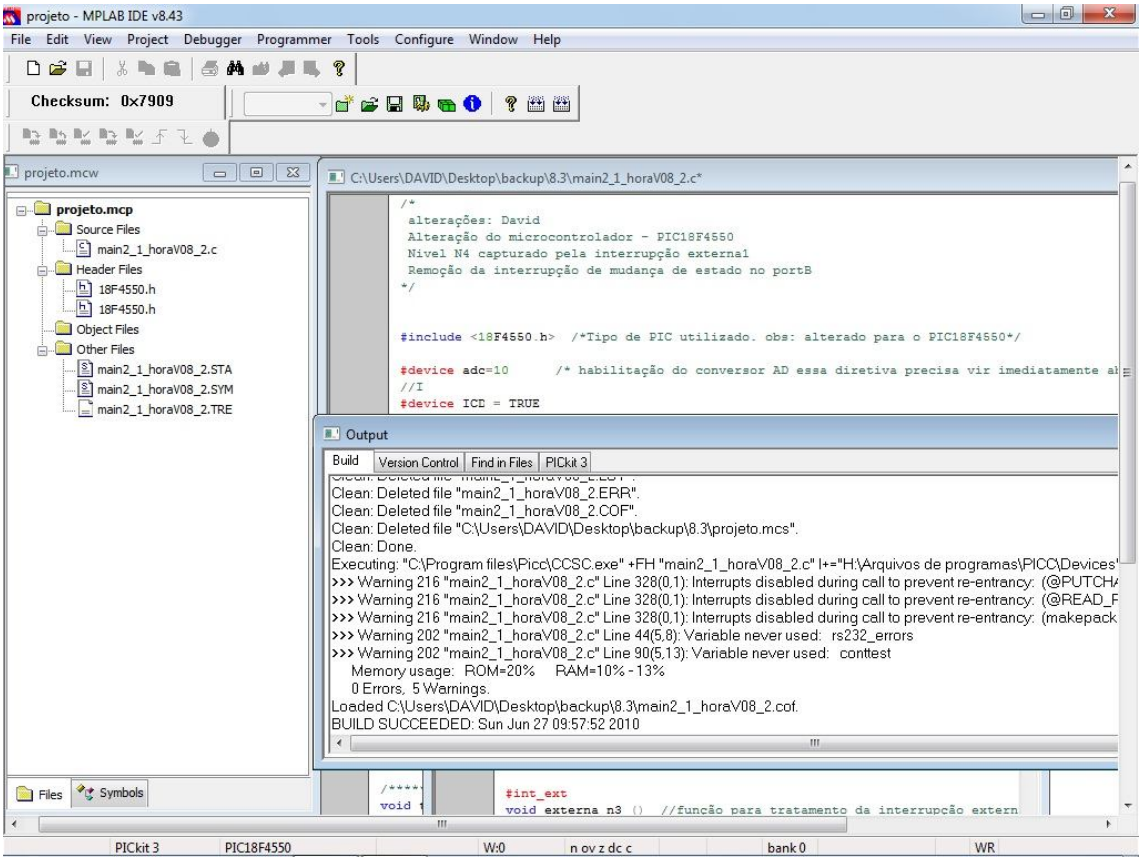


Figura 18. Compilação do projeto no *MPLab.*

5 Conclusões e trabalhos futuros

Dentro das perspectivas de projetos de automação, o uso de microcontroladores é uma possibilidade prática, eficiente e com muitas opções de aplicação. Neste projeto, o uso do microcontrolador *PIC18F4550* se mostrou muito promissor e dará suporte a atualizações futuras no detector de corrente de fuga de isoladores de linhas de transmissão elétrica, como por exemplo, a implementação de comunicação sem fio, aumento no número de níveis de corrente detectadas, entre outros.

Há também uma perspectiva quanto à melhoria do próprio programa, que com a oferta de recursos ampliada pela alteração do microcontrolador, poderá explorar ainda mais recursos do hardware, como utilizar a comunicação *USB* para a transferência de dados, algo que o novo microcontrolador oferece embutido. Como seu código está disposto de forma modular, estas adaptações podem ser feitas em paralelo sempre mantendo a hierarquia de acordo com o programa principal.

Para facilitar o entendimento do programa, podem-se utilizar ferramentas de documentação de código para controle de atualizações e manter um nível de organização semelhante à área de projetos de software, sempre embasada em metodologias para controle do processo de desenvolvimento.

Bibliografia

[1] MICROCONTROLADOR. Disponível em http://pt.wikipedia.org/wiki/microcontrolador. Acesso em 20 de março de 2010.

[2] SMITH, Kenneth C. Microelectronic Circuits, Edição 5. Editora Oxford University Press. Nova Yorque, 2004. 394 p.

[3] STALLINGS, W. Arquitetura e Organização de Computadores,Edição 5.Editora Pearson Addison Wesley. 2005.

[4] MELLO, C. A. B. Notas de aula de Teoria da Computação, UPE. 2007

[5] CLOCK. Disponível em http://pt.wikipedia.org/wiki/Clock. Acesso em 23 de março de 2010.

[6] PATTERSON, David A. John L. Hennessy. Organização e Projeto de Computadores. Edição 3. Editora CAMPUS. 2003. 512 p

[7] RISC. Disponível em http://pt.wikipedia.org/wiki/RISC. Acesso em 25 de março de 2010.

[8] MICROCONTROLADORES PIC. Disponível em http://www.microchip.com. Acesso em 21 de março de 2010.

[9] TANENBAUM, Andrew S. Organização estruturada de computadores. 5ª ed. São Paulo: Pearson Prentice Hall, 2007.

[10] TURING, A. M. COMPUTING MACHINERY AND INTELLIGENCE, Disponível em http://www.fbln.pro.br/htm_indexes/download.htm. Acesso 20 de maio de 2010.

[11] MICROCÓDIGO. Disponível em http://pt.wikilingue.com/es/Microcodigo

Acessoe em 13 de maio de 2010.

[12] TORRES, Gabriel. Hardware: Curso completo. Edição 4. Editora Axcel Books. 2001. 1.440 p.

[13] EPROM. Disponível em http://pt.wikipedia.org/wiki/EPROM. Acesso em 23 de março de 2010.

[14] OLIVEIRA, Sergio Campello. Sistemas de detecção óptica de descargas parciais em cadeias de isoladores de linhas de transmissão de alta tensão. 2008. 118 f. Tese de doutorado em engenharia elétrica, departamento de Engenharia Elétrica – UFPE, Recife-PE.

[15] Data Sheet do microcontrolador PIC16F877A disponível em

ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf. Acessado 1º de maio de 2010.

[16] SCHILDT, Herbert. C Completo e Total. 3 ed. Brasil: Makron Books, 1997. 827 p.

[17] CHESF, Disponível em http://www.chesf.gov.br/. Acesso em 23 de março de 2010.

[18] IDE. Integrated development environment, Ambiente de desenvolvimento Integrado para programação estruturada. Disponível em http://en.wikipedia.org/wiki/Integrated_development_environment. Acesso em 25 de maio de 2010.

[19] MPLab. Disponível em http://www.microchip.com.

Acesso em 05 de maio de 2010.

[20] CCS Inc. Home. Site Oficial do compilador CCS. Disponível em http://www.ccsinfo.com/. Acessado em 01 de junho de 2010.

[21] APOSTILA PWM. Disponível em www.eletronica.org. Acesso em 13 de maio de 2010.

[22] SOUZA, David José de. Desbravando o PIC. Edição 12. Editora Érica. 2001. 272 p.

[23] IDOETA, Ivan Valeije. Elementos da eletrônica digital. Edição 34. Editora Érica. 2001. 552 p.

[24] FERREIRA, Fabio. Microcontroladores PIC Programação em C. 6ª ed. São Paulo: Editora Érica. 358 p.

[25] TOCCI, Ronald J. Sistemas Digitais: Principios e Aplicações. Edição 8. Editora Prentice-hall. 2003. 768 p.

[26] PROTEUS. Disponível em http://www.labcenter.co.uk Acesso em 5 de junho de 2010.

[27] OLIVEIRA, S. C. Tutorial *Trax Maker,* Disponível em http://www.poli.br/~marcilio/Eletronica/TutorialTraxMaker.doc. Acesso em 10 de maio de 2010.

[28] Datasheet do microcontrolador PIC18F4550 disponível em http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf. Acesso em 1 de maio de 2010.

[29] Datasheet do sensor de umidade HIH3610 Series. Disponivel em http://www.datasheetcatalog.org/datasheets2/76/76269_1.pdf. Acesso em 3 de junho de 2010.

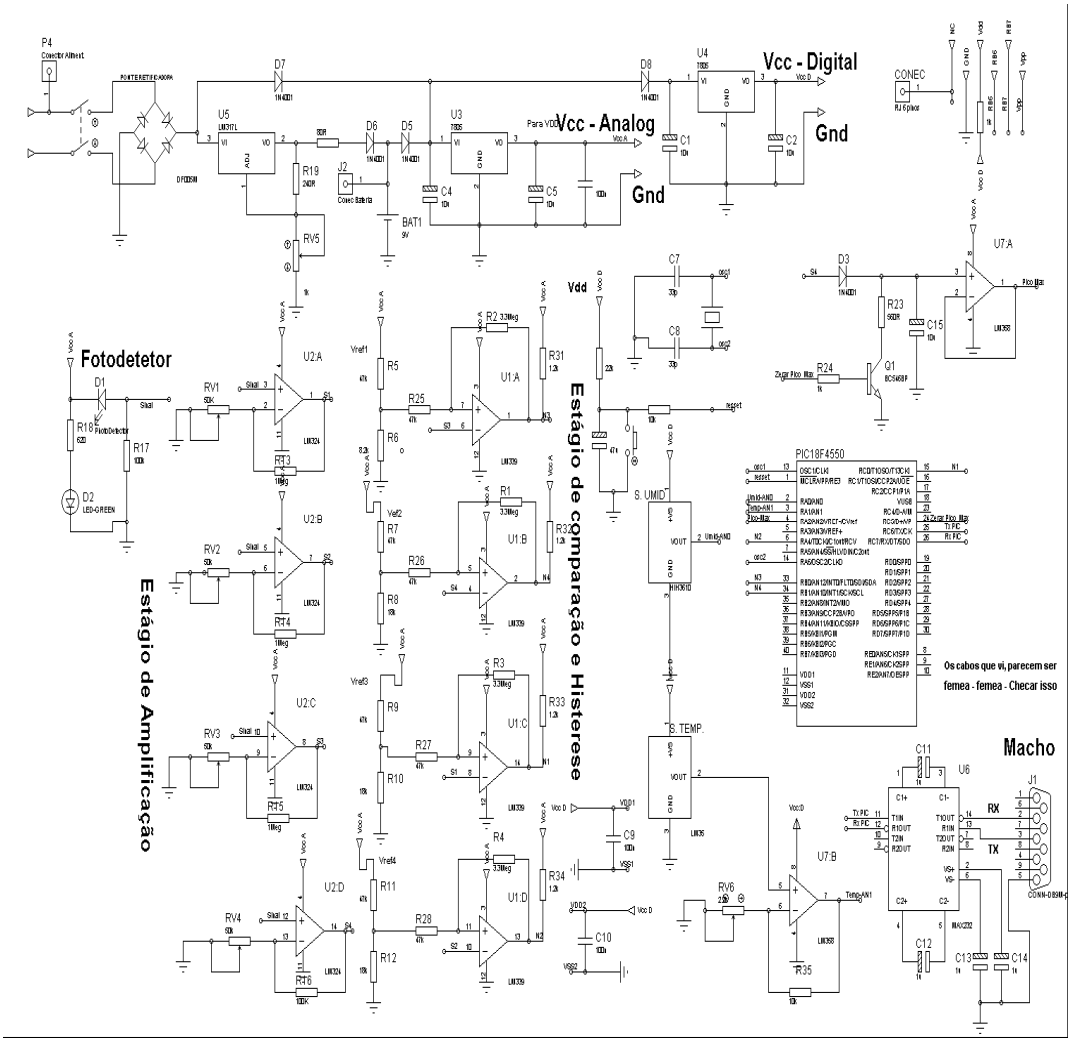
[30] Datasheet do sensor de temperatura LM35 Precision Centigrade Temperature Sensors. Disponível em ‘ http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/DS005516.PDF. Acesso em 30 de maio de 2010.

[31] ASSEMBLY. Linguagem de montagem. Disponível em http://pt.wikipedia.org/wiki/Linguagem_de_montagem

Acesso 4 de junho de 2010

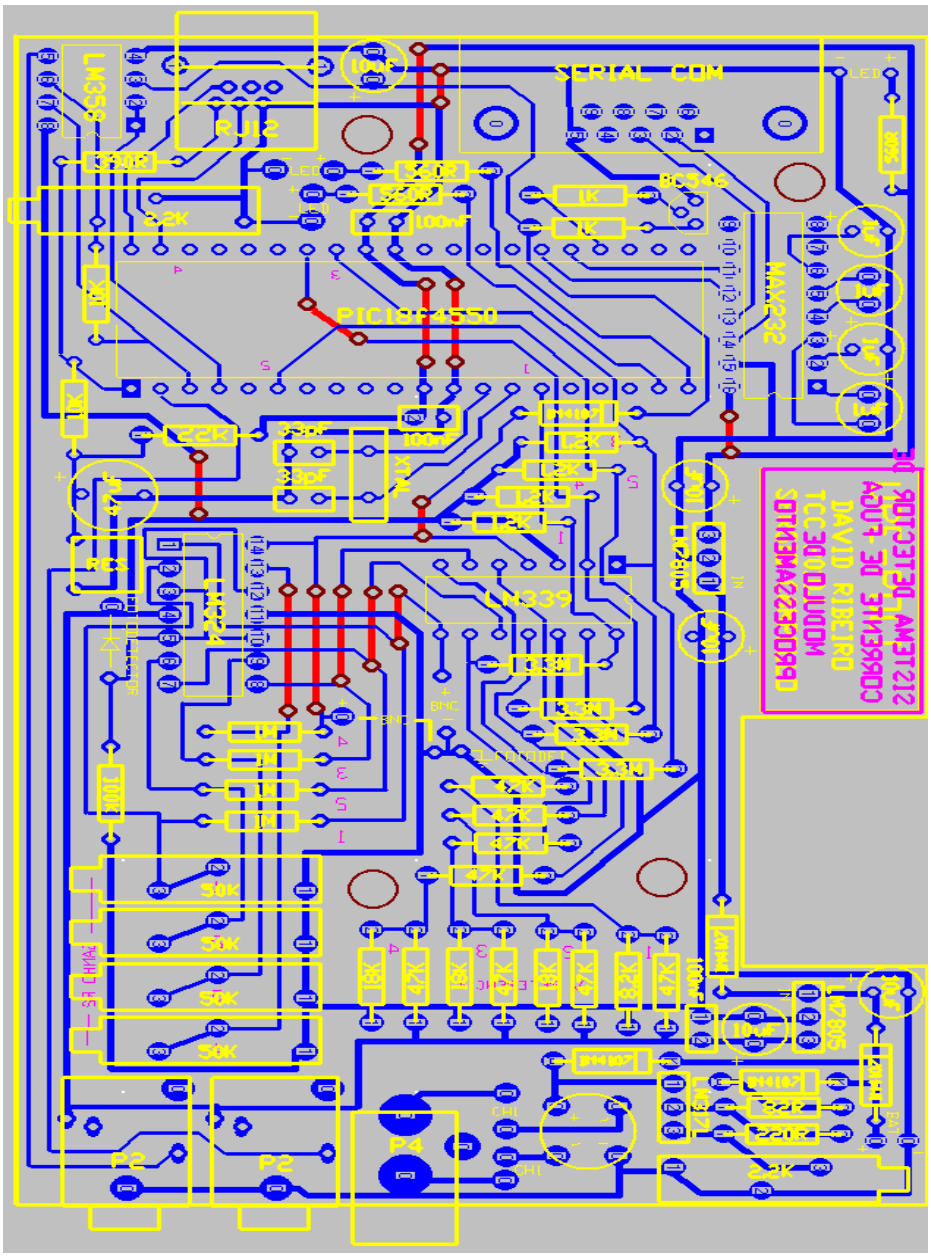
Apêndice A

Circuito elétrico adaptado do módulo de processamento.



Apêndice B

Desenho da placa de circuito impresso adaptada.



Apêndice C

Código fonte dos programas principal e contagem de níveis adaptados.

```
/*
 Alterações: David Edson Ribeiro
 Alteração do microcontrolador - PIC18F4550
 Nivel N4 capturado pela interrupção externa1
 Remoção da interrupção de mudança de estado no portB
*/

#include <18F4550.h>  /*Tipo de PIC utilizado. obs: alterado para o PIC18F4550*/

#device adc=10      /* habilitação do conversor AD essa diretiva precisa vir imediatamente abaixo do include do arquivo do processador*/

#device ICD = TRUE

#use delay (clock = 20000000)/* Cristal de 20 MHz*/

#fuses HS, NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP /*Tipo dos "fusíveis do pic*/

#use rs232 (stream=hw, baud=9600, bits=8, parity=n, uart1, errors) //configurações para a porta serial
//I
#use rs232(stream=sw, baud=1200, xmit=PIN_B7, rcv=PIN_B6, ERRORS)// serial em sw = software pela ICSP
//I
#priority  INT_TIMER0,INT_EXT,INT_EXT1,INT_TIMER1,INT_TIMER2 /* Prioridade no tratamento de interrupção, alteração para incluir INT_EXT1*/

#include <contagem_niveis2V08_2.c>/* Programa que conta quantas atividades ocorreram em cada nível*/

#include <umidade_temp2V08_2.c>/*Programa que calcula e faz as médias da umidade e temperatura*/

//#include <grava_flash2V08_2.c>/* Programa que realiza a gravação dos dados na memória do pic*/

#include <pacotes_1_horaV08_2.c>/*Programa responsável pela temporização AJUSTADO PARA 1 HORA*/

#include <grava_flash3.c>
#include <comunica_com_central1.c>

//#include <pacotes.c>/*Programa responsável pela temporização*/
```

```
//#include <transmissão_serial_30min.c> /*Programa com funções para a comunicação serial*/

/* Inicialização das variáveis principais*/

void main () /* declaração da variável principal*/

{
/*******************Declaração das variáveis de transmissão*******************/

int32 n1_temp=0;
int32 n2_temp=0;
int32 n3_temp=0;
int32 n4_temp=0;
int16 temperatura_temp=0;
int16 umidade_temp=0;
int16 pico_max_temp=0;

int8 a;

//int8 contagem_tentativas_sincronizacao=0;
//int8 momento_tentativa_sincronizacao=0;

//I
int conttest=0;
enable_interrupts(int_rda);
enable_interrupts(global);
verificaponte();
//I

/*****************************************************************************/

port_b_pullups(true);//Estabilidade da porta b

/*-----------------------------------------------------------------------------*/
   //umidade = 0;/* Inicialização da variável umidade com zero*/
   //temperatura = 0;/* Inicialização da variável temperatura com zero*/
//   address = buscar_endereco_base();/* Inicialização da variável address com a função busca
endereço base*/
  // first_idle = address;/* inicialização da variável first_idle com a variávell address*/
/*-----------------------------------------------------------------------------*/

    /*configura conversor AD*/
   setup_adc(ADC_CLOCK_DIV_32);   /*configura conversor AD*/
   setup_adc_ports(AN0_TO_AN4);   /*escolhe os pinos de entradas analógicas            no 18f4550*/


  /*Habilitação da interrupção global*/
   enable_interrupts(global);

    /* Habilitando as configurações da interrupção externa*/
   EXT_INT_EDGE(0,H_TO_L);
   enable_interrupts(int_ext);//alterado para comportar a externa1 PIC18F4550

   /*Habilitação as configurações da interrupção externa 1 */
   EXT_INT_EDGE(1,H_TO_L);
   enable_interrupts(int_ext1);//alterado para comportar a externa 1 PIC18F4550

   /* Habilitando as configurações de timer 0*/
   Setup_timer_0(RTCC_EXT_H_TO_L | RTCC_DIV_1);
```

```
   set_timer0(0); /*inicializa o timer com 0*/
   enable_interrupts(int_timer0);/* habilitar interrupção de timer 0*/

 /* Habilitando as configurações do timer1*/
   Setup_timer_1(T1_EXTERNAL_SYNC |T1_CLK_OUT );/*timer1 incrementado pelo  clock interno e
com prescaler 1:1*/
   set_timer1(0); /*inicializa o timer com 0*/
   enable_interrupts(int_timer1);/*Habilitação do timer 1*/

  /*Habilitando as funções do timer2*/
   //setup_timer_2(T2_DIV_BY_16,250,15);/* a base de tempo está em 12ms*/
   setup_timer_2(T2_DIV_BY_4,5,5); // para teste
   enable_interrupts(int_timer2);    /*habilita a interrupção de timer2*/
   set_timer2(0);
```

Programa de contagem de níveis de corrente de fuga

```
/*Alterado por David
  Nivel N4 passa a
  ser capturado pela
  interrupção externa1
  do pic18f4550
*/
//int32 n2 =0;
int32 n3 =0;
int32 n4 =0;/* variáveis usadas para tratamento de interrupção
int16 estouro_timer0=0; // conta a quantidade de estouros de timer 0(nível3)
int8 estouro_timer1=0; // indica se o timer 1 estourou (nível4)
//short int aceso1,aceso2,aceso3,aceso4;
//short int teste_n4 =0;

#int_ext1 //função alterada para comportar alteração do sinal N4 para a externa 1
void externa_n4 ()  //função para tratamento da interrupção externa1
{

      n4++;

}

#int_ext
void externa_n3 ()  //função para tratamento da interrupção externa

{

      n3++;
}


#int_timer0
void T0n2 () // função para contagem dos picos de N2

{
   if(estouro_timer0 !=1023)
      estouro_timer0++;
}

#int_timer1
void T1n1 () //função para contagem dos picos de N1
{
```

```
      //flag_n1 = 1;
      if(estouro_timer1 !=3)
        estouro_timer1++;
}


int32 obter_n1(){
   int32 n1;

   n1=((int32)estouro_timer1*65536) + get_timer1();

   return n1;
}

 int32 obter_n2(){
   int32 n2;
   n2 =((int32)estouro_timer0*256) + get_timer0() ;

   return n2;
}

void zerar_n3(){
   n3 = 0;
}

void zerar_n4(){
   n4 = 0;
}

void zerar_niveis(){
   set_timer0(0);
   set_timer1(0);
   n3=0;
   n4=0;
   estouro_timer0=0;
   estouro_timer1=0;

}
```