UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA

FAET – FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA

CURSO DE ENGENHARIA DA COMPUTAÇÃO

DANIEL DE OLIVEIRA SANTOS

MONITORAÇÃO E CONTROLE ATRAVÉS DE REDE ETHERNET

BRASÍLIA/DF

2º SEMESTRE DE 2007

DANIEL DE OLIVEIRA SANTOS

MONITORAÇÃO E CONTROLE ATRAVÉS DE REDE ETHERNET

Monografia apresentada ao Curso de Engenharia da Computação, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientadora: Prof. Maria Marony

BRASÍLIA/DF

2º SEMESTRE DE 2007

Resumo

Este projeto foi motivado pela observação das condições de segurança e insalubridade, vividas por pessoas que trabalham em salas de equipamentos de telecomunicações ou informática. O objetivo do projeto é amenizar tais condições, propondo uma forma eficaz para o gerenciamento remoto de equipamentos. Para isto, foi desenvolvido um circuito baseado em um microcontrolador capaz de se comunicar por uma rede Ethernet e detectar sinais de contato seco, criando uma ponte entre equipamentos e usuários. Assim, as informações de status dos equipamentos poderão ser visualizadas através de um programa navegador na Internet. Da mesma forma, também será possível executar comandos ou ações sobre tais equipamentos.

Palavras chave: Ethernet, PIC18F4620, ENC28J60, monitoração, controle.

Abstract

This project was motivated by the observation of conditions of security and unsanitary, lived by people who work in rooms of computers and telecommunications equipment. The project's goal is to mitigate such circumstances, offering an effective way for the remote equipment management. For this, a circuit has been developed based on a microcontroller able to communicate by an Ethernet network and detect signs of dry contact, creating a bridge between users and equipment. Thus, the information the status of the equipment can be viewed through a browser program on the Internet. Similarly, it will also be possible to execute commands or actions on such equipment.

Keywords: Ethernet, PIC18F4620, ENC28J60, management.

Agradecimentos

Agradeço primeiramente a Deus pelo dom da vida.
Agradeço a meus pais, Pedro e Dirce, e minha irmã, Flavia, que me incentivaram e me deram todo o apoio sempre que precisei, desde o início do curso.
A meu grande amigo, Juraci, pelos debates, idéias e apoio.
A professora orientadora, Maria Marony, pelo seu apoio e cobrança, sem os quais eu não teria atingido as metas dentro dos prazos.
A minha esposa, Flávia, pela compreensão, carinho e apoio constantes.

Sumário

Índice de figuras

Lista de tabelas

Capítulo 1. Introdução

Este projeto tem como objetivo o desenvolvimento de um circuito que interaja com equipamentos industriais, de telecomunicações ou com outro tipo de equipamento, ao qual ele possa ser adaptado. Ao mesmo tempo, esse circuito será capaz de interagir com o ser humano através de uma página web, onde estarão contidas as informações colhidas dos equipamentos conectados a ele.

O circuito é composto por cinco blocos principais, a saber: sensores, atuadores, processamento e armazenamento, interface de rede e página web.

O bloco de sensores é responsável por detectar sinais de contato seco provenientes de equipamentos ligados ao circuito. Este bloco também é capaz de realizar medida de temperatura ambiente, através de um sensor de temperatura.

Os atuadores são do tipo contato seco, ou seja, relés. Devido a essa característica, eles podem controlar uma infinidade de cargas, desde pequenos consumidores de corrente contínua até dispositivos maiores, com o consumo na ordem de alguns ampères em corrente alternada. Os atuadores são responsáveis por ligar ou desligar equipamentos sob comando do bloco de processamento e armazenamento.

Este bloco de processamento e armazenamento é composto por um microcontrolador e seus circuitos auxiliares. Ele é responsável por receber os sinais dos sensores, comandar os atuadores, controlar a interface de rede, armazenar a página web e atualizar os dados desta página com as informações dos sensores e atuadores.

Para que o usuário possa visualizar as informações dos sensores ou comandar os atuadores, existe uma interface de rede. Esta é composta por um único chip, e seus circuitos auxiliares. De um lado ela se conecta a uma rede Ethernet e de outro ao microcontrolador, usando comunicação SPI.

A página web é responsável por estabelecer um meio visual para o usuário, de forma que ele possa interagir com o circuito.

Enfim, esse circuito é uma forma de controle à distância, utilizando um padrão de comunicação bastante difundido e usual, que são as redes Ethernet e as páginas web.

Na figura 1.1 é visto o diagrama em blocos do circuito. Sendo que a página web está dentro do bloco Processamento e armazenamento.

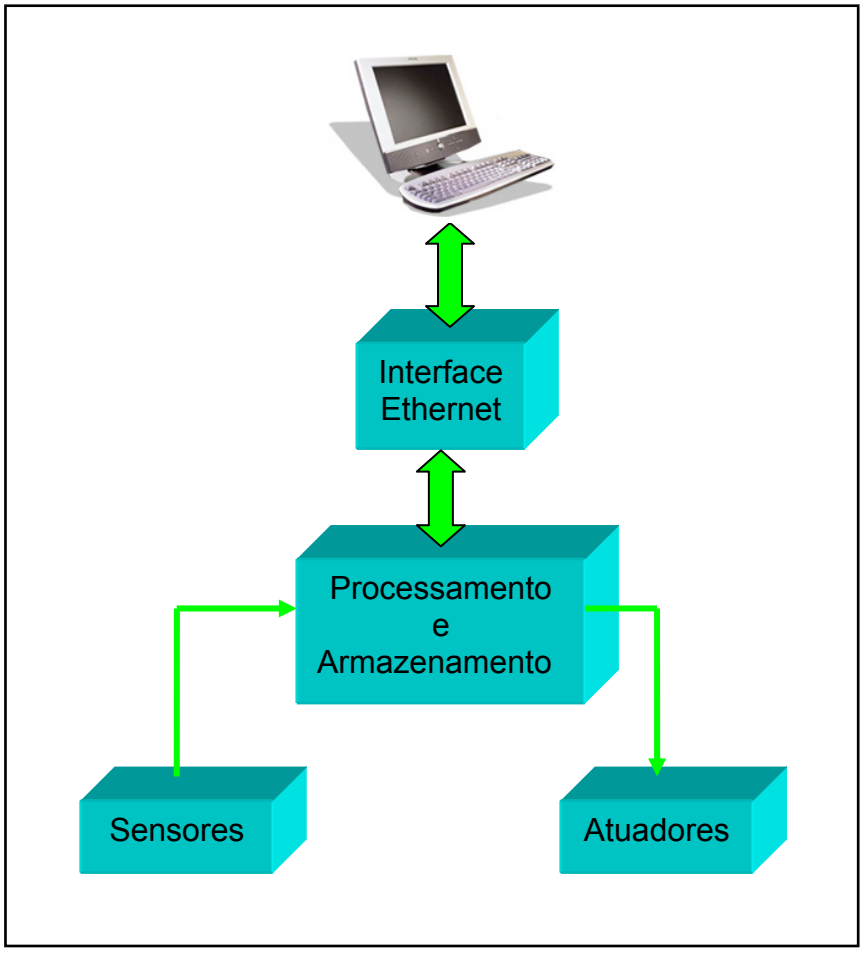


Figura 1.1 – Diagrama de blocos do projeto

1.1 Motivação

Devido a grandes evoluções nas últimas décadas, os equipamentos empregados nas indústrias vêm se tornando capazes de trabalhar de forma autônoma, dispensando a necessidade de haver operadores controlando cada processo. Mesmo assim, ainda é preciso executar manutenção, fazer leitura de informações, acionar e desligar equipamentos.

Sabe-se que ambientes preparados para serem usados como salas de equipamentos costumam conter ar condicionado, ventoinhas, baterias, pontos de alta tensão, dentre outros fatores que podem afetar a saúde ou a segurança física dos funcionários, e por isso podem ser considerados ambientes insalubres.

O ar condicionado, ligado constantemente e regulado para proporcionar baixas temperaturas, pode causar desconforto térmico ou até um enfraquecimento do sistema imunológico, especialmente quando há o choque térmico na transição de ambientes. Ventoinhas, mesmo bem lubrificadas, terminam por causar um ruído constante, que, a longo prazo, pode causar redução da capacidade auditiva, e, dependendo da quantidade de equipamentos, podem forçar os funcionários a falarem em um tom mais elevado de voz para se comunicarem dentro do ambiente. Baterias utilizadas por *nobreaks* costumam utilizar materiais químicos nocivos à saúde humana como metais pesados e ácidos. Os pontos de alta tensão representam um risco de vida constante, uma vez que os funcionários precisam efetuar medidas ou mesmo trabalhar diretamente, ou muito próximos a tais pontos.

A proposta do sistema de monitoração e controle aqui apresentado é proporcionar aos operadores uma forma de trabalho mais confortável e segura, trazendo maior qualidade de vida. É preciso lembrar que o sistema não evita completamente o contato homem-máquina, nem mesmo a necessidade de acesso aos ambientes de maquinário. A intenção é reduzir a freqüência com que estes contatos e acessos ocorrem.

O restante deste trabalho está dividido em cinco capítulos, compreendendo os seguintes assuntos: Hardware, Software, Resultados, Testes e Conclusão. O capítulo 2, Hardware, descreve os principais componentes utilizados no projeto. Serão abordadas, ainda, as necessidades de alimentação, os circuitos de cada bloco e as interligações entre eles, detalhando os tipos de sinais e quantidade de vias, entre outros.

No capítulo 3, Software, é feita a descrição em blocos do programa, detalhando as principais rotinas. Neste mesmo capítulo será feita a descrição da página web e a definição dos endereços IP e MAC do circuito.

O capítulo 4, Testes e Resultados, apresenta os testes que foram realizados, as tentativas que falharam e as alterações e definições que foram feitas com base nos testes. Além disso, serão apresentados os resultados obtidos com o desenvolvimento do projeto.

Enfim, no capítulo 5 é apresentada a Conclusão, onde estarão as considerações finais sobre o projeto e as possíveis linhas de continuidade do mesmo.

1.2 Metodologia

Como métodos de pesquisa para este projeto foram utilizadas as pesquisas bibliográficas, pesquisas na Internet e em periódicos. Tais pesquisas foram importantes tanto para a composição da monografia quanto para a construção do protótipo.

1.3 Objetivo

Este projeto tem por objetivo o desenvolvimento de um circuito de monitoração e controle através de rede Ethernet. Tal circuito pode ser utilizado no ramo das telecomunicações, servindo de sistema de gerenciamento remoto de equipamentos e estações de transmissão.

Capítulo 2. Hardware

Este capítulo apresenta o funcionamento do circuito, as características dos principais componentes utilizados, bem como as necessidades de alimentação de energia elétrica, o detalhamento dos circuitos de sensores e atuadores e a interligação de cada bloco que compõe o hardware, mostrando os tipos de sinais e suas características.

O funcionamento do circuito é descrito a seguir.

Os sensores detectam as alterações dos equipamentos conectados ou da temperatura ambiente e repassam essas informações ao microcontrolador. Este, por sua vez, trata essas informações e as utiliza para atualizar a página web, que é transmitida pela interface de rede. No sentido contrário, o usuário aciona um botão na página web, que está sendo executada no próprio microcontrolador, esse fato faz com que o microcontrolador capture a informação e acione o atuador correspondente ao botão pressionado, acionando a carga que estiver conectada a este atuador.

A seguir é apresentado o microcontrolador, sua pinagem, seu diagrama de blocos e suas características mais importantes para o projeto. Em seguida, é apresentado o controlador de rede Ethernet, sua pinagem e suas características. Na seqüência serão vistos os demais circuitos, seja dos sensores, seja dos atuadores.

2.1 Microcontrolador

O fabricante do microcontrolador escolhido para esse projeto é a Microchip. Sediada em Chandler, Arizona a Microchip é um dos líderes mundiais em fabricação e fornecimento de microcontroladores e semicondutores analógicos. [Microchip, 2007]

O microcontrolador escolhido para esse projeto foi o PIC 18F4620. A escolha foi feita com base em seus recursos, dentre eles, a comunicação através de

interface SPI e a capacidade de memória que servem, respectivamente, para conectá-lo ao controlador Ethernet e para armazenar o programa juntamente com a página web.

Microcontroladores são circuitos integrados programáveis e que possuem, na mesma pastilha (ou chip), todos os componentes necessários à execução de seus processos. Com isso podem ser empregados nas mais diversas aplicações eletrônicas. [SOUZA, 2003]

Estes equipamentos possuem memória interna para o armazenamento do programa que vai controlar todas as suas funções. O programa aí armazenado é a inteligência do microcontrolador, ele deve ser escrito compilado e, então, gravado na memória do microcontrolador. A funcionalidade do microcontrolador será limitada pela inteligência do programa e também pelas características de seus componentes internos. [SOUZA, 2003]

Internamente um microcontrolador possui Unidade Lógica e Aritmética (ULA), memória de programa, memória de dados, portas de entrada e/ou saída, *timers*, contadores, PWMs, conversores analógico-digitais e outros. [SOUZA, 2003] Vale lembrar que a disponibilidade e a quantidade de cada um desses recursos depende da versão do microcontrolador, o que influencia também no seu custo.

Os microcontroladores estão presentes no dia-a-dia das pessoas, mesmo sem serem percebidos. Eles fazem parte de eletrodomésticos como máquinas de lavar, fornos microondas, celulares e brinquedos eletrônicos. Esses são apenas alguns exemplos, pois, às vezes, onde menos se imagina existe um microcontrolador.

Não se deve confundir microcontrolador com microprocessador, são dispositivos distintos. Existem duas diferenças básicas entre eles:

- A ULA dos microprocessadores é muito mais poderosa que a dos microcontroladores.
- Os microprocessadores não possuem todos os recursos agregados na mesma pastilha. Já os microcontroladores, com seus vários recursos agregados, são capazes de trabalhar sem a necessidade de nenhum periférico.

2.1.1 Microcontrolador PIC 18F4620

Neste ponto são apresentadas as principais características do microcontrolador PIC 18F4620, bem como sua pinagem e seu diagrama de blocos.

Principais características:

- Memória de programa com 65536 Bytes
- Memória de dados com 3968 Bytes
- EEPROM de 1024 Bytes
- 20 Interrupções
- 5 Portas de entrada e saída (A, B, C, D e E)
- Comunicação serial SPI, I2C e USART
- 13 entradas para conversor analógico-digital de 10 bits
- Timers
- Entrada de clock para até 40MHz

2.1.2 Pinagem

A pinagem do PIC18F4620 pode ser vista na figura 2.1, onde são apresentados os 40 pinos e suas respectivas funções. Pode-se também, identificar as portas RA, RB, RC, RD e RE e os pinos de alimentação (VDD e VSS). A maioria dos pinos possui mais de uma função. Dependendo da configuração e do programa, eles podem utilizar mais de uma função durante a operação, ou seja, em modo de operação normal, os pinos de entrada e saída podem ser utilizados como entrada e também como saída, sendo que a alteração da funcionalidade é feita pelo programa instalado. isso maximiza o aproveitamento do microcontrolador e permite a utilização em situações críticas, onde se requer muitas entradas e saídas. [SOUZA, 2003] O PIC18F4620 requer uma alimentação de 5 Volts para seu funcionamento normal e a freqüência do clock pode ser de até 40MHz.

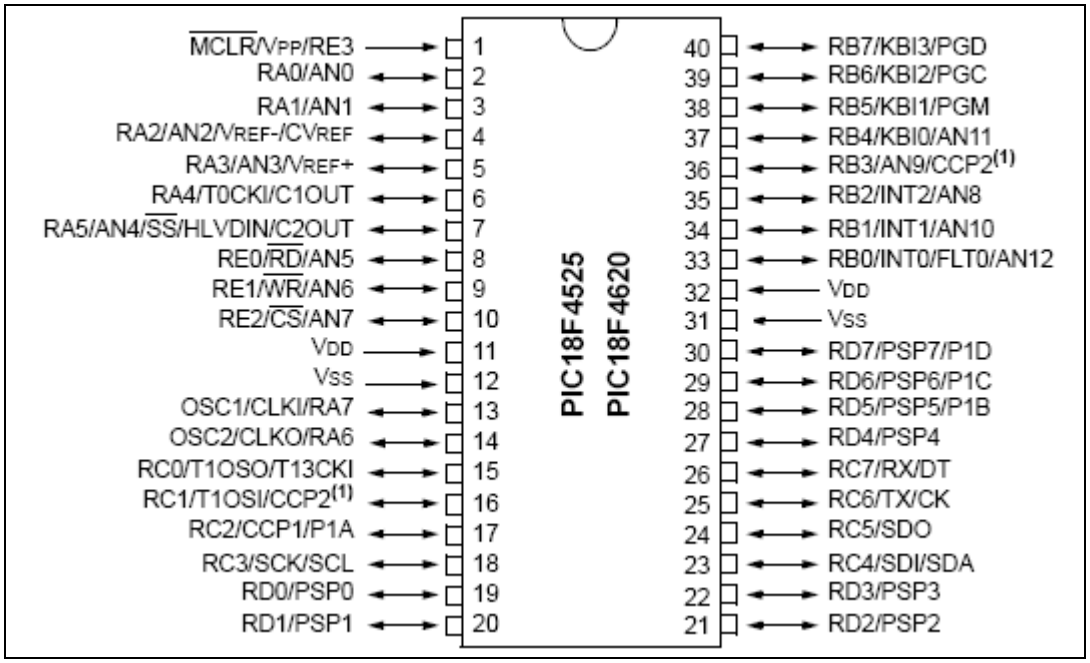


Figura 2.1 – Pinagem do PIC18F5620

2.1.3 Estruturação Interna

Na figura 2.2 é apresentado o diagrama de blocos do PIC 18F4620, mostrando em detalhes todos os periféricos e linhas de comunicação que o compõem.

No diagrama podem ser visualizados todos os componentes do microcontrolador. Na parte superior encontram-se as memórias de programa e de dados, a tabela de ponteiros, o contador de programa e os registradores de endereço. A ULA, próxima ao centro, é ligada ao registrador W e ao barramento de dados, ainda no centro temos o decodificador de instruções e os circuitos responsáveis por osciladores, timers, reset e gravação. [Microchip, 2007]

Os demais periféricos estão na parte inferior, inclusive a interface de comunicação SPI, os conversores A/D e os módulos CCP (Compare, Capture e PWM). Do lado direito estão os PORTs, de PORTA a PORTE, com a indicação das funções de cada pino. Os valores associados aos barramentos são referentes à quantidade de bits dos mesmos, ou seja, o barramento de programa, que interliga a

memória de programa ao registrador de instruções, possui 16 bits, já o barramento de dados possui 8 bits. [Microchip, 2007]

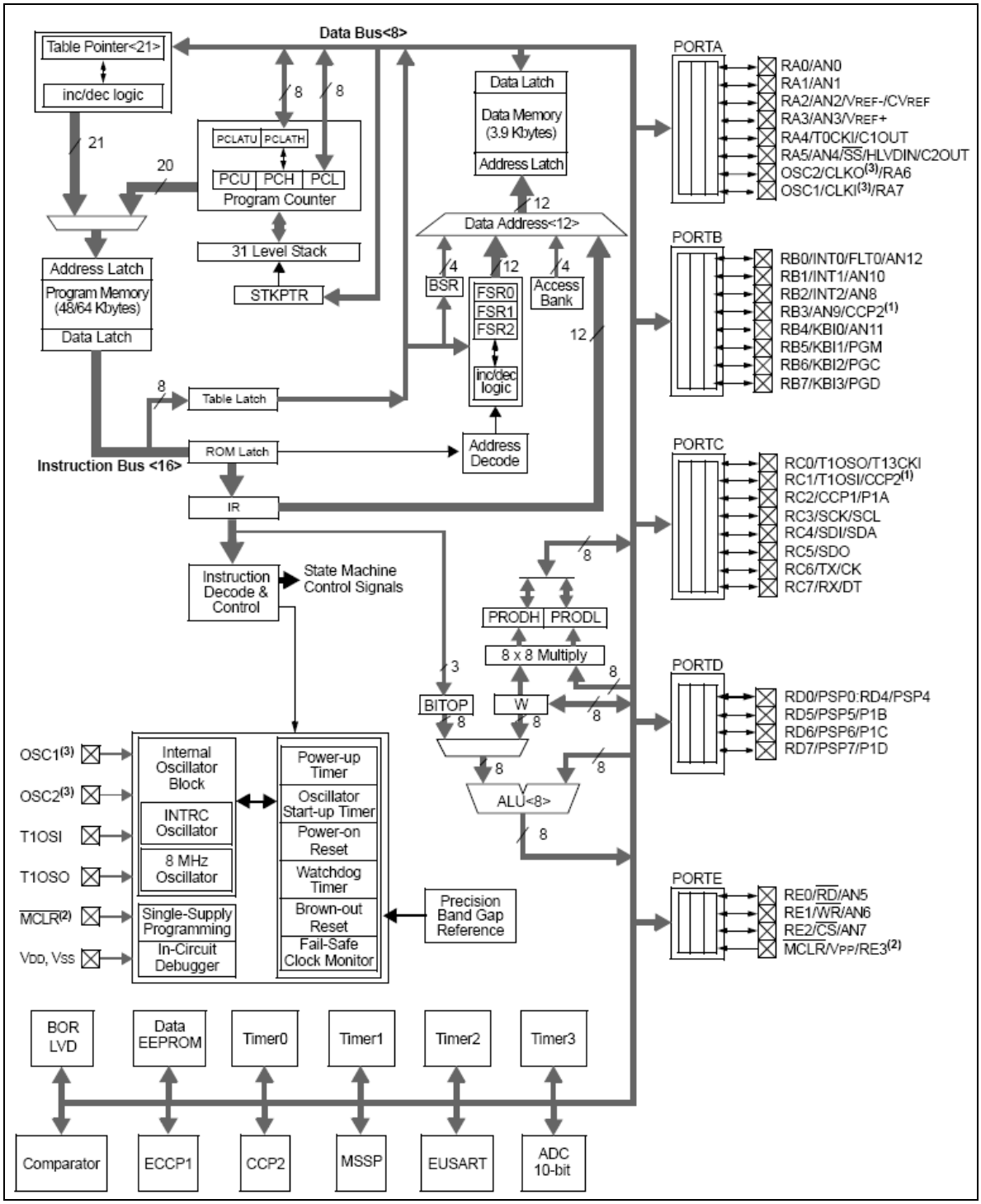


Figura 2.2 – Diagrama de blocos do PIC18F4620 [Microchip, 2007]

2.1.4 Considerações sobre os ciclos de máquina

Na maioria dos modelos da linha PIC o clock interno é equivalente a ¼ do clock externo [SOUZA, 2003]:

$$CK\,\mathrm{int} = \frac{CKext}{4}$$

Assim, se o clock externo for de 4MHz, por exemplo, o clock interno será de 1MHz. Dessa forma, a duração de um ciclo de máquina (CM ou Tcy) será de 1µs. [SOUZA, 2003]

$$Tcy = \frac{1}{CK\,\mathrm{int}}$$

O clock é dividido devido ao funcionamento interno do processador. São necessárias várias operações para que uma instrução seja executada. Essas operações são executadas em cada subciclo de máquina (Q1, Q2, Q3 e Q4), como vistos na figura 2.3. [SOUZA, 2003]

O contador de programa, que aponta para a próxima instrução a ser executada, é incrementado sempre no início de Q1, a instrução será executada durante o período de Q1 a Q4. Isso é suficiente para que a ULA troque informações com a memória de dados e o registrador W, quando necessário. Ao final do período Q4 a próxima instrução é buscada na memória. [SOUZA, 2003]

Todo esse processo também é conhecido como *Pipeline*, e, considerando o clock interno de 4 MHz, quase todas as instruções podem ser executadas em apenas 1µs. Na figura 2.3 é ilustrado esse conceito. [SOUZA, 2003]

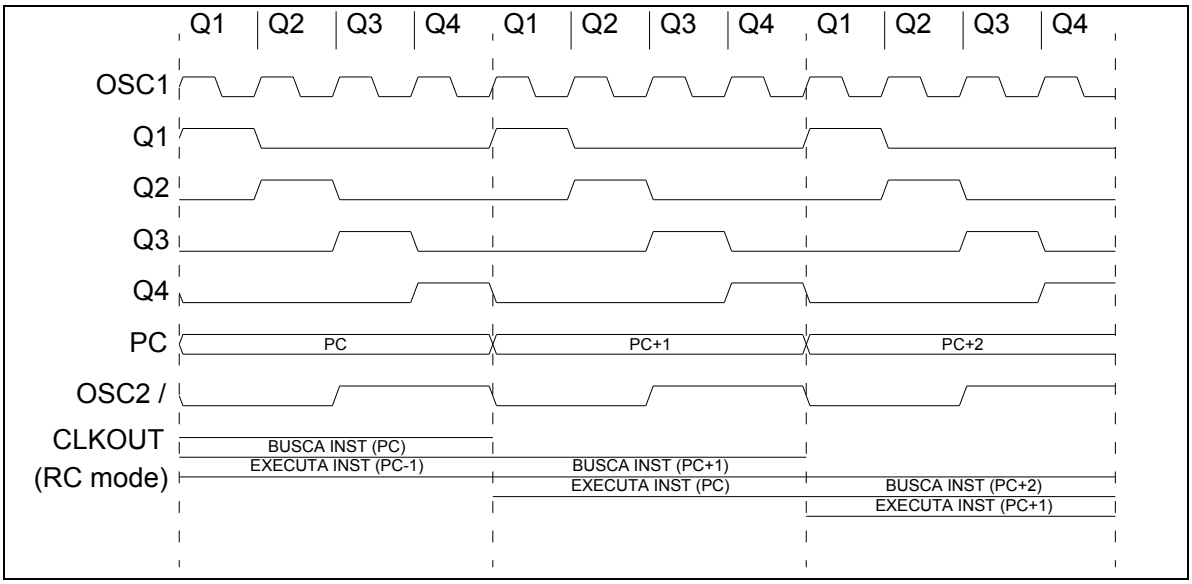


Figura 2.3 – Clock externo/clock interno no PIC 18F4620

2.2 Controlador Ethernet

Estão disponíveis no mercado vários controladores Ethernet, contudo o controlador ENC28J60 foi escolhido para este projeto devido a sua simplicidade e baixo custo. O ENC28J60, ilustrado na figura 2.4, é fabricado pela Microchip, o mesmo fabricante do microcontrolador. Ele foi desenvolvido para servir de interface entre microcontroladores e redes Ethernet, além disso, é encapsulado em um único chip requerendo poucos componentes externos. Traz ainda uma interface SPI para a comunicação com o microcontrolador. [Microchip, 2007]

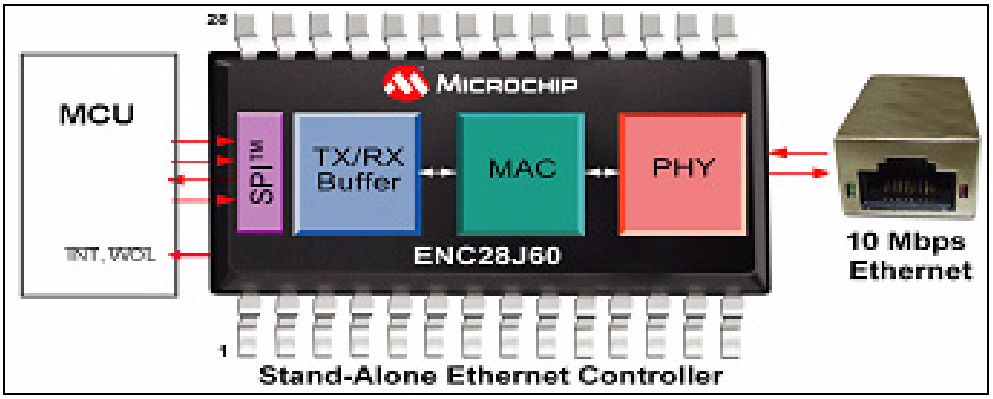


Figura 2.4 – Controlador Ethernet ENC28J60

Este controlador reúne todas as especificações IEEE 802.3 e incorpora esquemas de filtragem de pacotes para limitar os pacotes de entrada. Além disso, possui um módulo DMA interno, que é utilizado para maiores vazões de dados, e executa cálculos de *checksum* do IP com o auxílio do *hardware*. Para a comunicação com o microcontrolador, o ENC28J60 dispõe de dois pinos de interrupção e da interface SPI, que pode atingir uma taxa de transmissão de até 10Mbps. São disponibilizados ainda, dois pinos para a ligação de LEDs, que indicam atividade da rede e do link. [Microchip, 2007]

A ligação deste controlador a uma rede Ethernet é simples e requer poucos componentes no circuito, dentre eles um transformador de pulso.

De acordo com o diagrama de blocos apresentado na figura 2.5, o ENC28J60 é composto por sete blocos principais:

1. Interface SPI. Utilizada para comunicação entre o microcontrolador *host* e o próprio ENC28J60.
2. Registradores de controle. Como o próprio nome diz, esses registradores são utilizados para controlar e monitorar o dispositivo.
3. *Buffer*. Um *buffer* do tipo RAM de duas portas, para os pacotes recebidos e transmitidos.
4. Controlador de acesso ao buffer. Utilizado quando são feitas requisições através do DMA nos blocos de transmissão e recepção.
5. Interface de barramento. Responsável por interpretar dados e comandos recebidos através da interface SPI.
6. Módulo MAC (*Medium Access Control*). Responsável por implementar o padrão IEEE 802.3.
7. Módulo PHY (camada Física). Responsável por codificar e decodificar os dados analógicos presentes na entrada e saída de par trançado.

O circuito integrado conta ainda com outros blocos de suporte, tais como um oscilador, um regulador de tensão, tradutores de níveis para que as portas de entrada suportem 5 Volts e sistema de controle lógico. Todos estes periféricos podem ser vistos no diagrama de blocos a seguir, além deles, temos ainda o filtro de recepção e o controle de fluxo de transmissão. [Microchip, 2007]

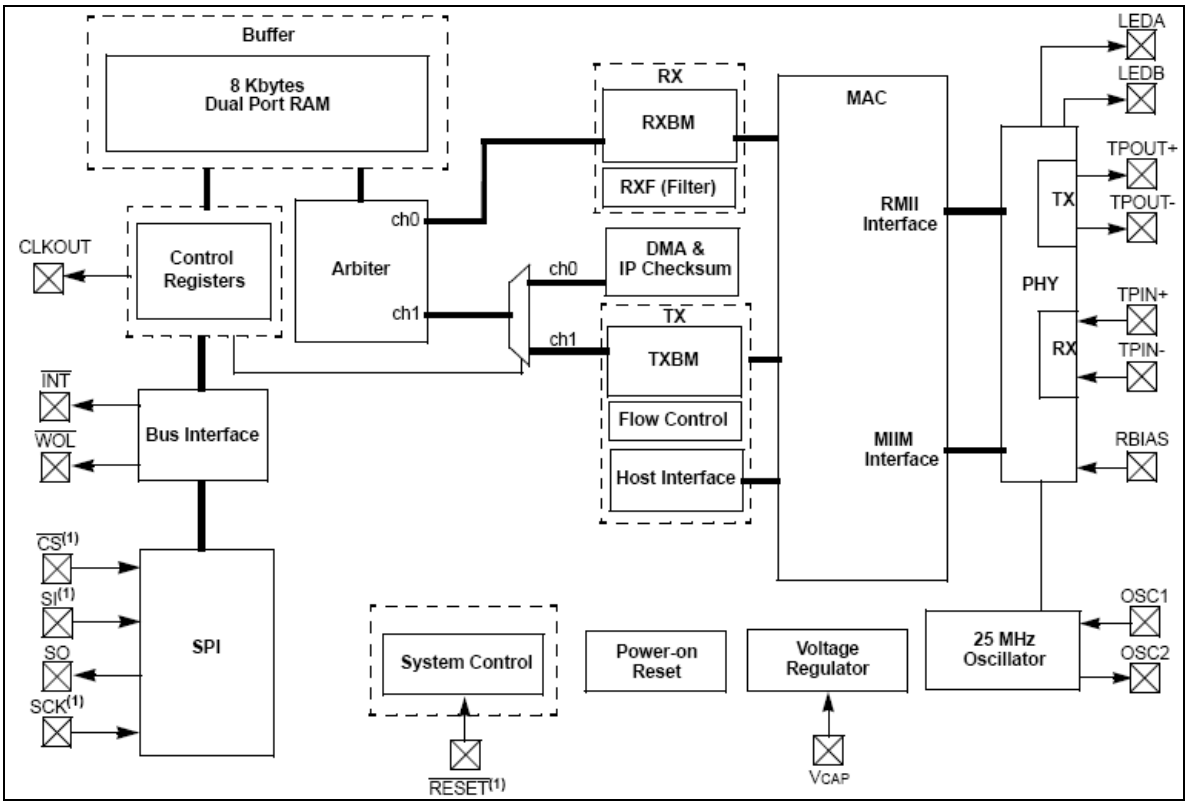


Figura 2.5 - Diagrama de blocos do ENC28J60 [Microchip, 2007]

A tensão de alimentação deste controlador é de 3,3 Volts. Dessa forma, os sinais de saída da sua interface SPI precisam passar por um buffer para que possam apresentar níveis suficientes às entradas do microcontrolador. A pinagem do ENC28J60 é vista na figura 2.6, onde podem ser identificados seus pinos de entrada e saída, tanto da interface SPI quanto da interface Ethernet. Pinos de alimentação, reset e controle também estão identificados. [Microchip, 2007]

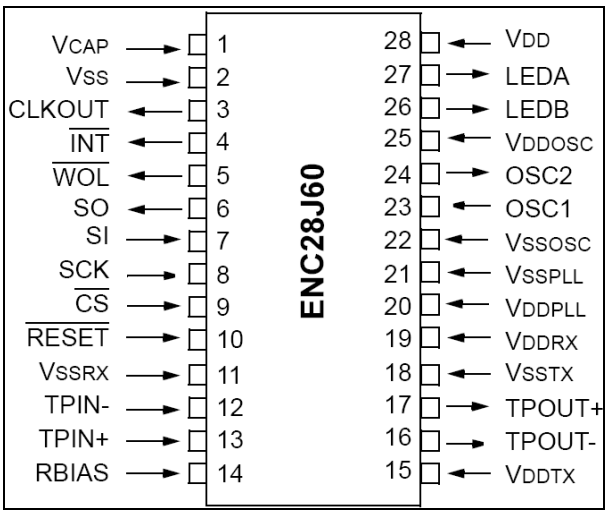


Figura 2.6 - Pinagem do ENC28J60

2.3 Redes Ethernet

“Ethernet é uma rede de barramento em que múltiplos computadores compartilham um meio de transmissão único. Enquanto um computador transmite um quadro para outro, todos os demais computadores devem esperar.” [COMER, 2001]

A Ethernet é um sistema utilizado para interligar computadores e, com isso, compartilhar arquivos e recursos, que podem ser impressoras, scanners ou qualquer outro dispositivo. Tudo isso no mesmo ambiente ou no mesmo prédio, de forma que se use uma mesma rede. [MUSEU DO COMPUTADOR, 2004]

A empresa Xerox criou a primeira impressora laser do mundo em seu centro de pesquisa em Palo Alto, onde foram criados também, alguns dos primeiros computadores pessoais. Dessa forma, houve o desejo que todos os computadores do centro de pesquisa fossem capazes de imprimir nessa impressora. Para isso, foi solicitado a um dos membros do grupo de pesquisa que desenvolvesse o sistema capaz de fazer tal interligação. Seu nome era Robert Metcalfe. [MUSEU DO COMPUTADOR, 2004]

O desafio era construir uma rede rápida para acompanhar o desempenho da nova impressora, além de conectar centenas de computadores no mesmo prédio, isso nunca havia sido feito antes, conectavam-se, no máximo, dois ou três computadores. Não se pensava ainda em grandes redes. [MUSEU DO COMPUTADOR, 2004]

Em 22 de maio de 1973 Metcalfe escreveu um memorando para seu chefe para lhe comunicar o potencial da Ethernet, segundo a imprensa, essa teria sido a data oficial da criação da Ethernet. [MUSEU DO COMPUTADOR, 2004]

Em 1976 Metcalfe e seu assistente publicam uma nota sobre a Ethernet e seu potencial. Em 1979 ele sai da Xerox e começa a promover o uso de computadores e de redes locais. Por fim Metcalfe conseguiu convencer três grandes empresas a trabalharem juntas para estabelecer o padrão Ethernet, essas empresas eram a Digital Equipment, a Intel e a Xerox Corporations. [MUSEU DO COMPUTADOR, 2004]

Originalmente, uma rede ethernet era composta por um cabo coaxial único chamado éter, em que podiam ser conectados vários computadores. O limite de comprimento do cabo coaxial é de 500 metros e, por padrão, é necessário que haja uma distância mínima de 3 metros entre cada par de conexão. O *hardware* é capaz de operar em uma largura de banda de 10Mbps. [COMER, 2001]

Em se tratando da velocidade das redes de computadores, pode-se observar que tal velocidade vem aumentando gradativamente.

Há algum tempo, quando se tratava de velocidade de transmissão em redes de computadores, os “gargalos” eram os próprios computadores, que não possuíam capacidade de processamento suficiente para atingir os 10Mbps, do primeiro padrão aprovado pelo IEEE. Dessa forma a evolução da Ethernet a 100Mbps demorou cerca de 20 anos. [SIQUEIRA, 2004]

Na década de 1990 a velocidade dos computadores aumentou de forma significativa, da mesma forma as redes também evoluíram, ocasionando uma corrida pelo padrão Gigabit. [SIQUEIRA, 2004]

Atualmente o baixo custo e a simplicidade das redes Ethernet são os principais motivos que fazem com que elas sejam usadas em mais de 90% das redes corporativas. Além disso, essa tecnologia vem evoluindo muito rapidamente nos últimos anos, juntamente com as transmissões em fibra óptica, que já operam a 40Gbps e, conforme alguns fabricantes, nos laboratórios já se atingem velocidades de até 6,4Tbps. [SIQUEIRA, 2004]

Com base no processo de evolução das redes Ethernet, podem ser feitas algumas previsões para o futuro dessa tecnologia. No início a Ethernet trabalhava com um cabo coaxial, que proporcionava uma conexão compartilhada em uma largura de faixa de apenas 10Mbps. Em seguida passou a utilizar o par trançado, com a mesma largura de faixa de 10Mbps, mas utilizando comutação e não o compartilhamento. [SIQUEIRA, 2004]

Hoje temos Ethernet comutada, conexões com o computador a 100Mbps e troncos de 1Gbps. Contudo, já existem computadores que podem ser conectados à rede com uma taxa de transmissão de 1Gbps. A previsão para um futuro próximo é que a conexão da maioria dos computadores seja feita a 1Gbps e os troncos atinjam os 10Gbps. [SIQUEIRA, 2004]

De qualquer forma, pode-se perceber que os computadores evoluem e passam a utilizar velocidades mais altas em suas conexões, ao mesmo tempo as redes também evoluem e apresentam maiores taxas de transmissão.

2.4 Adaptadores de rede

Adaptadores de rede ou, comumente chamados placas de rede, são dispositivos necessários para que o computador possa se comunicar com uma rede, através desse dispositivo o computador pode trocar informações com os outros computadores conectados à rede. [TORRES, 1998]

A função do adaptador de rede é basicamente controlar o fluxo de dados, seja transmissão seja recepção, entre o computador e a rede. Cada arquitetura de rede exige um tipo específico de adaptador, assim não se pode utilizar, por exemplo, uma placa Token Ring em uma rede Ethernet. Outra questão a ser observada é o tipo do barramento utilizado pelo adaptador de rede, que para os computadores pode ser dos tipos PCI, ISA, On-board ou outros. [TORRES, 1998]

Contudo, os microcontroladores não costumam possuir barramentos do tipo ISA ou PCI, mas podem implementar comunicações do tipo serial ou paralela. [SOUZA, 2003] E através dessas interfaces podem se comunicar com o mundo exterior, desde que se tenha um adaptador para o tipo de rede a ser utilizada.

Como o microcontrolador escolhido para esse projeto possui uma interface SPI (Serial Peripherall Interface ou Interface Serial para Periféricos) o adaptador de rede deverá apresentar esse mesmo tipo de interface. Dessa forma foi escolhido o ENC28J60, que atende a esses requisitos, além de ser fabricado pela Microchip, mesmo fabricante do microcontrolador.

O ENC28J60 é um adaptador de rede encapsulado em um único chip, foi projetado para ser uma interface de rede Ethernet para microcontroladores que se comuniquem através de interface SPI. [Microchip, 2006]

Esse adaptador incorpora várias funcionalidades e também atende a todas as especificações IEEE 802.3. Ainda assim esse adaptador precisa de alguns

componentes externos para funcionar, tais como transformador e resistores. [Microchip, 2006]

2.5 Sensores

Aqui são detalhados os sensores que compõem o projeto, seus circuitos, componentes e características.

São dois tipos de sensores, um de contato seco e outro de temperatura. O sensor de contato seco é capaz de detectar a abertura ou o fechamento de contato proveniente de equipamentos ou dispositivos como reed-switch ou chaves de contato comuns. O circuito conta com dois sensores deste tipo. Já o sensor de temperatura, é capaz de realizar medidas de temperatura entre 2 e 150 graus Celsius.

O sensor de contato seco é composto por um resistor de pull-up, um capacitor para redução de ruídos provenientes dos contatos e uma porta lógica do tipo AND, usada para adequar o nível de sinal à porta de entrada do microcontrolador. Seu circuito pode ser visto na figura 2.7.

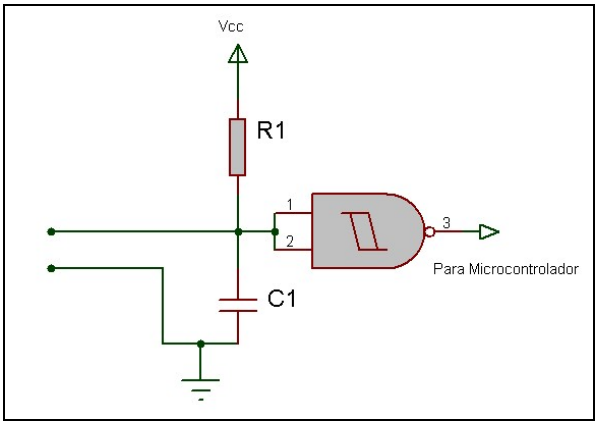


Figura 2.7 - Esquema do sensor de contato seco.

O sensor de temperatura é composto de um circuito integrado LM35, que é capaz de responder a variações de temperatura entre -55 e 150 graus Celsius, de acordo com a montagem. O LM35 não requer muitos componentes externos, na

implementação utilizada, ele não necessita de nenhum outro componente. Este sensor pode receber como alimentação uma tensão entre 4 e 18 Volts, e responde com variações de 10mV por grau Celsius, com precisão de +/- 1 grau.

A saída do sensor de temperatura é ligada diretamente a uma porta analógica do microcontrolador, para que este possa fazer a leitura da tensão e a conversão em graus Celsius. Na Figura 2.8 é mostrado o esquema elétrico. Vale lembrar que o encapsulamento deste sensor é TO-92, ou seja, o mesmo de um transistor comum.

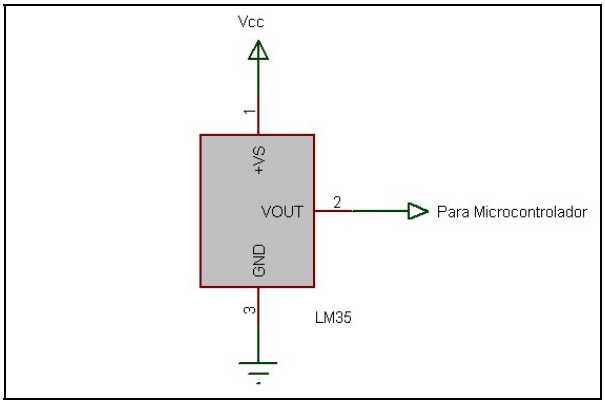


Figura 2.8 – Sensor de temperatura

2.6 Atuadores

Foram desenvolvidos dois tipos de atuadores, um para pequenas cargas ou apenas para ativar um contato a ser detectado por outro equipamento e outro para ativar cargas maiores, como dispositivos elétricos por exemplo.

O atuador tipo 1 mostrado na figura 2.9, é composto de uma porta lógica do tipo AND, que recebe o sinal do microcontrolador e o utiliza para acionar o relé através do resistor R1, que por sua vez, serve para limitar a corrente de saída da porta lógica. O diodo D1 serve para reduzir os ruídos gerados pela passagem de corrente na bobina do relé. O conjunto resistor R2 e LED D2 serve para indicar a atividade do relé, utilizando um dos dois contatos disponíveis. De acordo com a especificação, o relé deste atuador tem capacidade de condução entre seus

contatos de até 1 ampère em corrente contínua ou de até 0,5 ampère em corrente alternada.

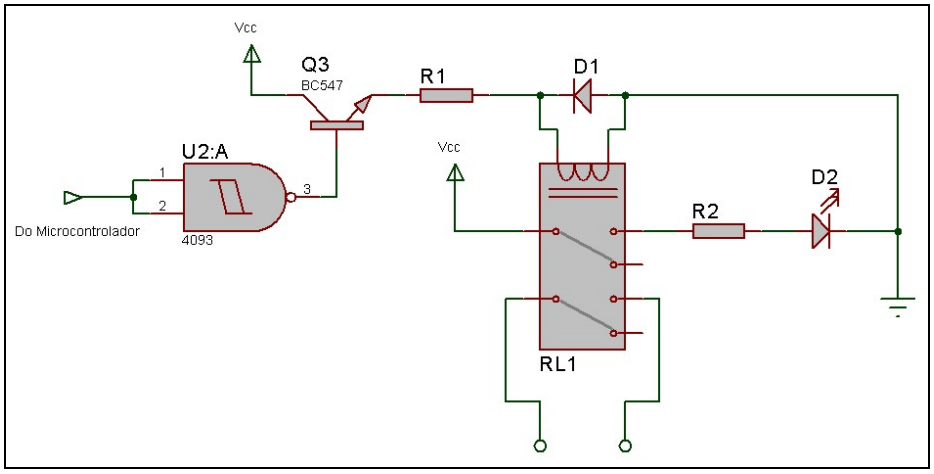


Figura 2.9 – Atuador tipo 1

O atuador tipo 2, apresentado na figura 2.10, é bastante parecido com o atuador tipo 1. A diferença está na utilização de um par de transistores ligados na forma par Darlington, para fornecer corrente suficiente para a ativação do relé. Este relé possui maior capacidade de condução de corrente entre seus contatos, e necessita de uma maior corrente para ser ativado.

O resistor R1 é utilizado para limitar a corrente drenada do par Darlington e, da mesma forma que no atuador 1, o diodo D1 serve para reduzir os ruídos gerados pela passagem de corrente na bobina do relé, enquanto o conjunto resistor R2 e LED D2 serve para indicar a atividade do relé, utilizando um dos dois contatos disponíveis. Este atuador pode acionar cargas que possuam um consumo de corrente menor ou igual a 5 ampères. Esta característica é determinada pela especificação do relé.

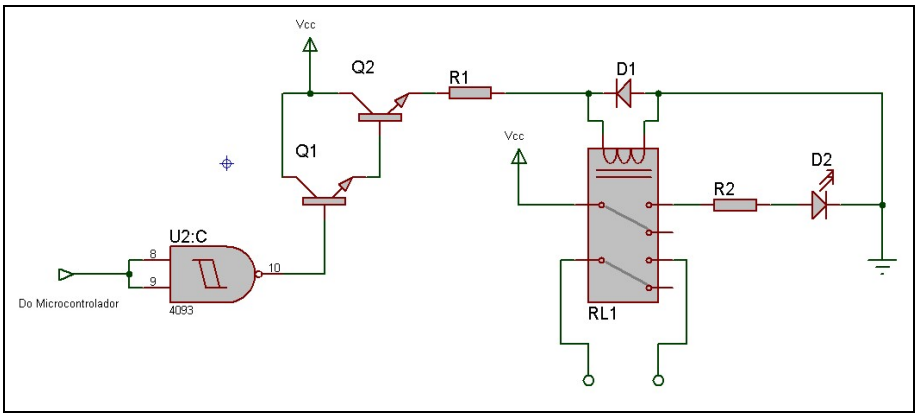


Figura 2.10 – Atuador tipo 2

2.7 Diagrama em blocos do hardware

Neste ponto é apresentado o diagrama de blocos do hardware, mostrando a interligação de todos os blocos já apresentados, ou seja, o microcontrolador, a interface de rede, os sensores e os atuadores. Serão vistos, também, os sinais que trafegam entre estes blocos.

A fonte de alimentação não será detalhada, pois a mesma é uma fonte de computador do tipo ATX. Esta foi escolhida por ter as tensões de 5V e 3,3V, esta segunda utilizada para alimentar o circuito da interface de rede.

Na figura 2.11 os blocos Sensor 1 e Sensor 2 representam os sensores de contato seco, há ainda o sensor de temperatura e os dois blocos de atuadores. Todos estes estão interligados ao microcontrolador, que por sua vez está diretamente ligado à interface de rede.

O sinal que trafega entre os blocos Sensor 1 e Microcontrolador e Sensor 2 e Microcontrolador são do tipo TTL, com 5V de amplitude. Quando este sinal está em nível baixo, ou seja, 0V significa que o sensor não está detectando o fechamento de contato. Já quando este sinal está em nível alto, ou seja, 5V significa que o sensor detectou o fechamento de um contato.

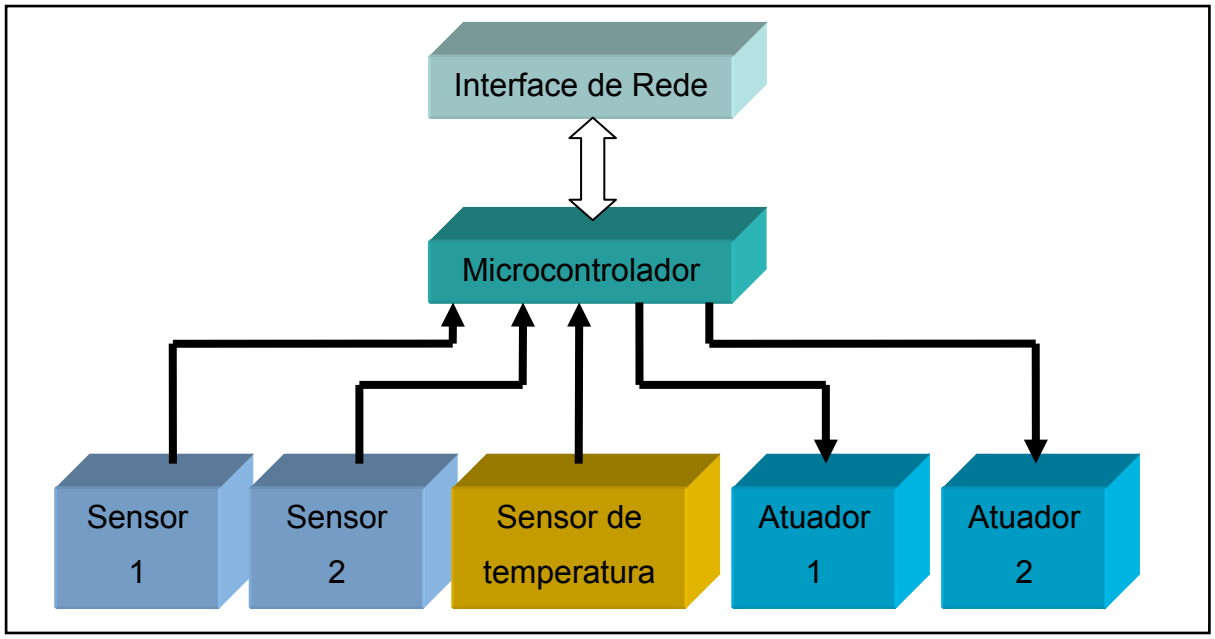


Figura 2.11 – Representação em blocos

O sinal gerado pelo sensor de temperatura e entregue ao microcontrolador é uma tensão analógica, que pode variar de 0 a 5V, de acordo com a temperatura medida pelo sensor, ou seja, 10mV/ºC. Este sinal entra em uma porta analógica do microcontrolador.

Já os sinais que seguem no sentido inverso, ou seja, do microcontrolador para os atuadores, também são sinais do tipo TTL. Quando em nível baixo os atuadores permanecem desativados, quando em nível alto os atuadores acionam seus relés e consequentemente as cargas ligadas a eles.

Entre o microcontrolador e a interface de rede o tráfego de sinais é bidirecional, no sentido do microcontrolador para a interface o nível de tensão é de 5 Volts, mesmo assim, para garantir a integridade das entradas do ENC28J60, são usados resistores nestas interligações. No sentido contrário, como o ENC28J60 é alimentado com 3,3 Volts, suas saídas também apresentam esse mesmo nível, e, para que o microcontrolador possa identificar corretamente os níveis, é utilizado um buffer, que recebe níveis de 0 a 3,3V e repassa o mesmo sinal com níveis de 0 a 5V.

Nessa interligação há um total de cinco vias, sendo três no sentido Microcontrolador – Interface, e duas no sentido Interface – Microcontrolador. As três primeiras são Clock, Dados e Seleção de chip, as outras duas são Interrupção e Dados. Os sinais que trafegam nestas vias são mostrados no capítulo 4 em imagens registradas em osciloscópio.

Capítulo 3. Software

Como se sabe, para que um microcontrolador funcione conforme o desejado é preciso que ele seja programado com as rotinas que implementam suas funcionalidades. A programação do microcontrolador pode ser feita em várias linguagens, Assembler, Basic ou C, por exemplo. O circuito proposto por este projeto depende, em grande parte, do software nele instalado, sem o qual ele não funcionaria. Além do programa do microcontrolador, também há a página web, que serve para que o usuário possa interagir com o circuito.

Neste capítulo serão abordadas as principais características do software instalado no microcontrolador, sua descrição em blocos, a definição dos endereços IP e MAC e a descrição da página web que será armazenada no circuito.

3.1 A utilização de redes de computadores

As redes de computadores são utilizadas tanto para aplicações comerciais quanto para aplicações domésticas. A seguir serão apresentados alguns motivos pelos quais pessoas e empresas estão interessadas em redes de computadores.

3.1.1 Aplicações comerciais

Várias empresas possuem uma quantidade considerável de computadores destinados a diversas aplicações, interligadas ou não. Inicialmente, tais computadores trabalhavam de forma independente, mas, com a necessidade de se correlacionar informações, foi necessário estabelecer uma forma de conexão destas máquinas. Ou seja, fez-se necessário o compartilhamento de recursos, que tem o

objetivo de disponibilizar aos usuários todos os recursos de programas, dados e equipamentos. [TANENBAUM,2003]

Os usuários não são independentes, precisam de informações e serviços oferecidos por um sistema centralizado, tais como troca de mensagens e acesso aos dados e programas. Esse tipo de trabalho cooperativo está presente tanto em empresas quanto em universidades. [SOARES, 1995]

Grande parte das empresas de grande e médio porte são dependentes de informações computadorizadas. Essas informações costumam ser armazenadas em servidores e por isso este modelo recebe o nome de cliente/servidor. Esse modelo é apresentado na figura 3.1. [TANENBAUM,2003]

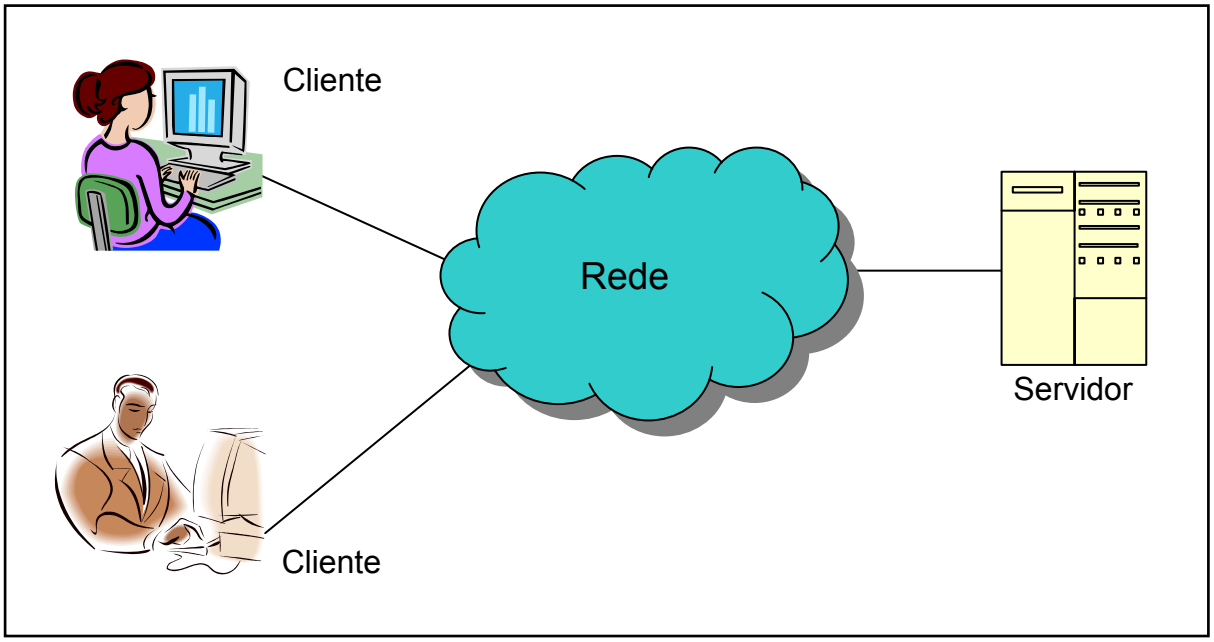


Figura 3.1 – Modelo Cliente/Servidor

A importância do compartilhamento de recursos pode ser menor se comparada à do compartilhamento de informações. Isso porque os dados armazenados e compartilhados podem ser vitais, principalmente para empresas de médio e grande porte. Por exemplo, se os computadores de uma linha de produção parassem de funcionar, a produção seria imediatamente interrompida e a empresa teria prejuízo. [TANENBAUM,2003]

Podem ser identificadas, ainda, outras aplicações que são bastante utilizadas em empresas, tais como o Correio Eletrônico, a Videoconferência, o comércio entre empresas e o comércio direto com consumidores. [TANENBAUM,2003]

O Correio Eletrônico é utilizado para comunicação entre funcionários, substituindo memorandos em papel e outras documentações em geral. A videoconferência é utilizada para fazer reuniões sem a necessidade do deslocamento dos participantes, reduzindo custos com viagens e hospedagens. Os comércios entre empresas e entre empresa e consumidor são bastante parecidos, o comércio entre empresas é feito basicamente para a compra e venda de matéria prima, já entre empresa e consumidor, de uma forma geral, é feita a venda de produtos manufaturados. [TANENBAUM,2003]

3.1.2 Aplicações domésticas

Hoje em dia milhares de residências possuem pelo menos um computador. Esses computadores são utilizados para os mais diversos fins. Mas não foi sempre assim, no início, os computadores eram máquinas destinadas a universidades, grandes empresas e fins militares, tanto pelo seu alto custo quanto pela sua aplicação. Esse quadro mudou e hoje o computador também está presente nas residências. Não era o que pensava o presidente da Digital Equipment Corporation em 1977, Ken Olsen, que dizia não haver razões para se ter um computador em casa. Talvez por esse motivo a empresa não exista mais. [TANENBAUM,2003]

As aplicações domésticas são várias, desde processamento de textos e jogos até transações financeiras. A maior alavanca para o uso dos computadores em casa é a Internet, que proporciona acesso a informações remotas, comunicação entre pessoas, entretenimento e comércio eletrônico, dentre várias outras aplicações. [TANENBAUM,2003]

Um serviço bastante utilizado, principalmente por adolescentes, é a troca de mensagens instantâneas, ou salas de bate-papo, nas quais se pode conversar com pessoas em qualquer lugar do mundo. Existem também os grupos de notícias e fóruns, com discussões sobre uma infinidade de tópicos. Quanto ao entretenimento, existem, dentre outros, os vídeos por demanda, que podem ser acessados em vários servidores desse tipo espalhados na Internet, e os jogos on-line, em que se

reúnem grupos de usuários que compartilham do mesmo ambiente de jogo. [TANENBAUM,2003]

O comércio eletrônico já é uma realidade e vem sendo bastante utilizado. Nessa aplicação, podem ser feitas compras sem a necessidade de sair de casa, consultar catálogos de várias empresas e comparar preços. As instituições financeiras também estão conectadas, tornando possível o pagamento de contas, a administração de contas bancárias e fazer aplicações financeiras. Uma utilização bastante difundida atualmente é o mercado livre eletrônico, onde os usuários podem comprar, vender e leiloar seus produtos, sejam eles novos ou usados. Nesse caso o contato é praticamente de consumidor para consumidor. [TANENBAUM,2003]

Uma outra modalidade de utilização doméstica de computadores é a troca de arquivos entre usuários, em que os participantes se comunicam entre si, enviando e recebendo músicas, vídeos e vários outros tipos de arquivos. Nesse caso é utilizada a comunicação não hierárquica, ou peer-to-peer, que é diferente do modelo cliente/servidor por não haver servidores, a comunicação é feita diretamente de um usuário a outro, como mostra a figura 3.2. [TANENBAUM,2003]

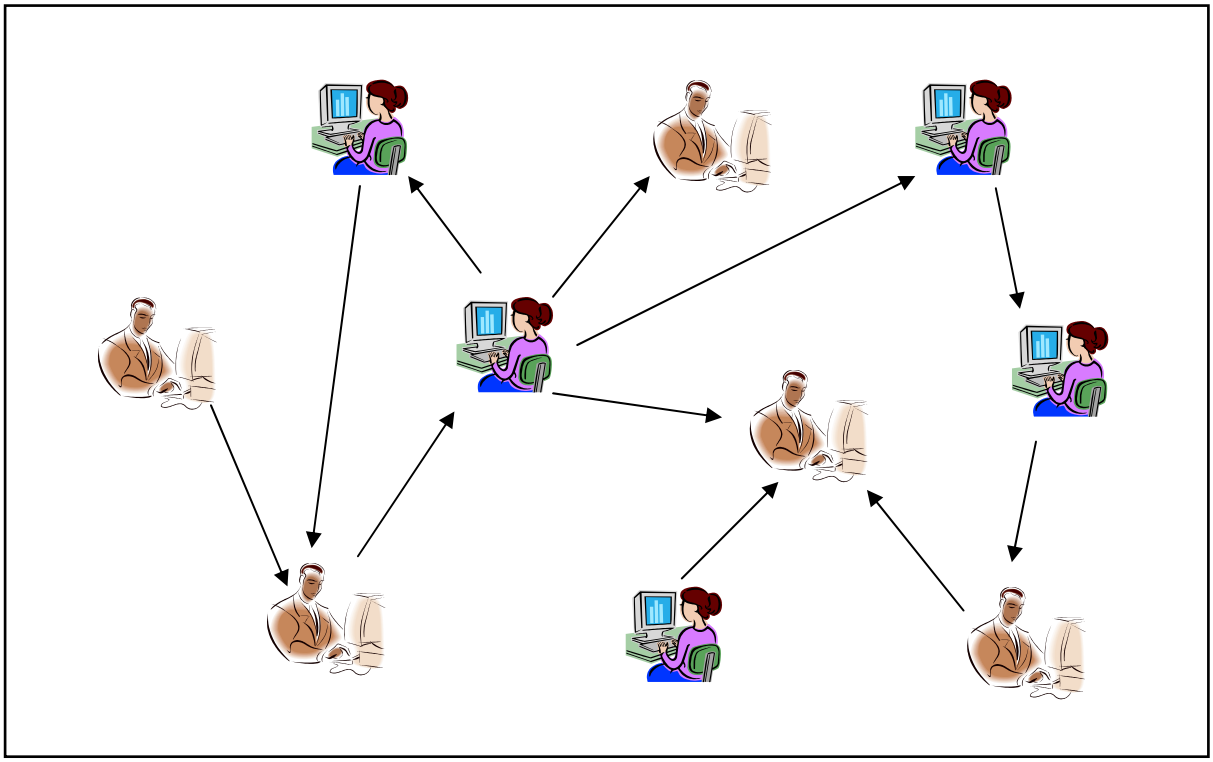


Figura 3.2 – Comunicação não hierárquica

3.2 Protocolos

Conforme Jorge Luis da Silveira, protocolos são conjuntos de regras preestabelecidas e de conhecimento das partes, que disciplinam a comunicação de dados entre dois ou mais dispositivos com a finalidade de garantir que o intercâmbio de informações seja realizado de modo ordenado e sem erros. [SILVEIRA, 1991]

Existem vários protocolos, eles são utilizados de acordo com a aplicação e com as condições da rede. Contudo, de acordo com o foco deste projeto, este tópico apresenta os protocolos TCP e IP.

3.2.1 TCP

O TCP foi criado a partir do UDP, que, apesar de simples e eficiente para tráfego de multimídia e interações entre clientes e servidores, não tem a característica de entrega confiável e em seqüência, necessária à maioria das aplicações da Internet. [TANENBAUM,2003]

O principal objetivo do TCP é oferecer um fluxo de bytes fim-a-fim em uma rede ou inter-rede não confiável. Além disso ele deve se adaptar dinamicamente às propriedades desta rede ou inter-rede e possuir robustez suficiente para enfrentar as várias falhas que podem ocorrer. Como a camada IP não garante a entrega dos datagramas de forma apropriada, o TCP precisa administrar os contadores e retransmitir os datagramas perdidos ou não recebidos, bem como reorganizá-los na seqüência correta. Tudo isso garante a confiabilidade que não é oferecida pelo IP. [TANENBAUM,2003]

O formato do cabeçalho TCP é apresentado na figura 3.3 e seus campos são detalhados a seguir conforme Comer e Tanenbaum.

- Portas de Origem e de Destino: número das portas TCP que identificam os aplicativos nas extremidades.
- Número de seqüência: posição do datagrama no stream de bytes do transmissor.
- Número de Reconhecimento: identifica o número do octeto que a origem espera receber como retorno.
- HLEN: comprimento do cabeçalho do segmento.
- Bits de Código: determinam ou informam a finalidade do conteúdo.
- Janela: quantidade de dados a serem enviados;
- Soma de Verificação: verificação do cabeçalho para aumentar a confiabilidade.
- Ponteiro urgente: Informa a localização dos dados urgentes.
- Opções: espaço para recursos extras.
- Dados: dados a serem transmitidos.

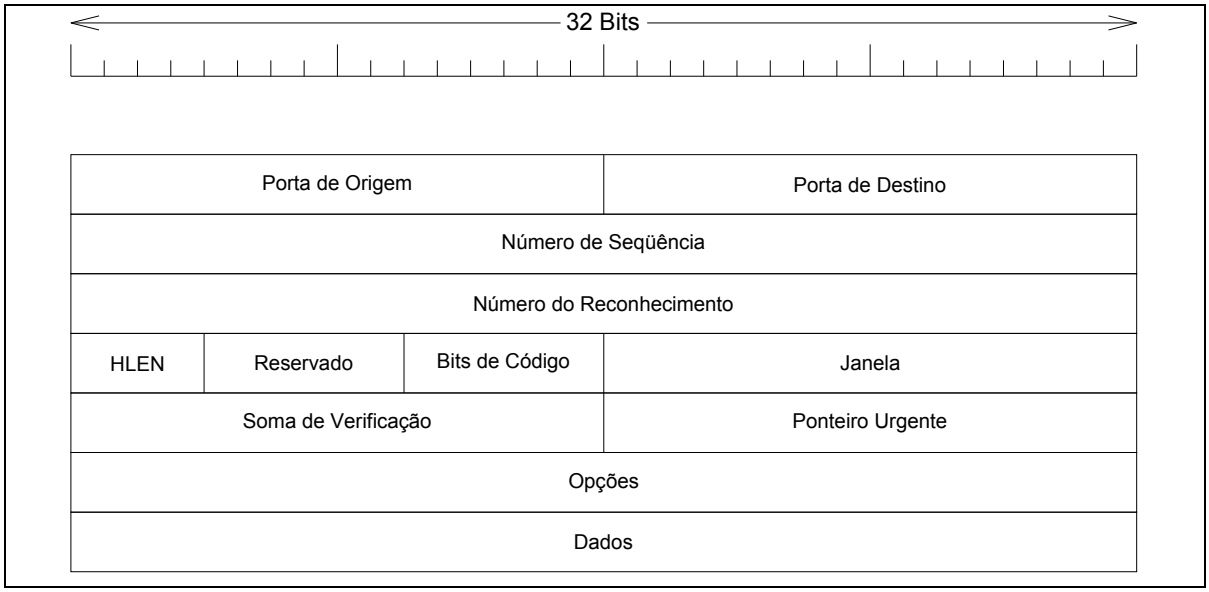


Figura 3.3 – O cabeçalho TCP

3.2.2 IP

O protocolo IP é responsável por manter a Internet unida. Ele foi projetado para promover a interligação de redes. O cabeçalho IP é apresentado na figura 3.4 e cada um de seus componentes é descrito abaixo conforme Tanenbaum.

- Versão: versão do protocolo utilizado.
- IHL: Informa o tamanho do cabeçalho em palavras de 32 bits.
- Tipo de Serviço: distingue as diferentes classes de serviços.
- Comprimento Total: informa o comprimento total do datagrama (cabeçalho e dados)
- Identificação: informa a qual datagrama pertence cada segmento.
- Offset: informa a que ponto do datagrama o fragmento pertence.
- TTL: contador utilizado para limitar a vida útil dos pacotes.
- Protocolo: informa o processo de transporte do datagrama.
- Soma de verificação do cabeçalho: verificação de erros no cabeçalho
- Endereços de Origem e de Destino: Indicam o número da rede e do host
- Opções: Espaço para alterações futuras.

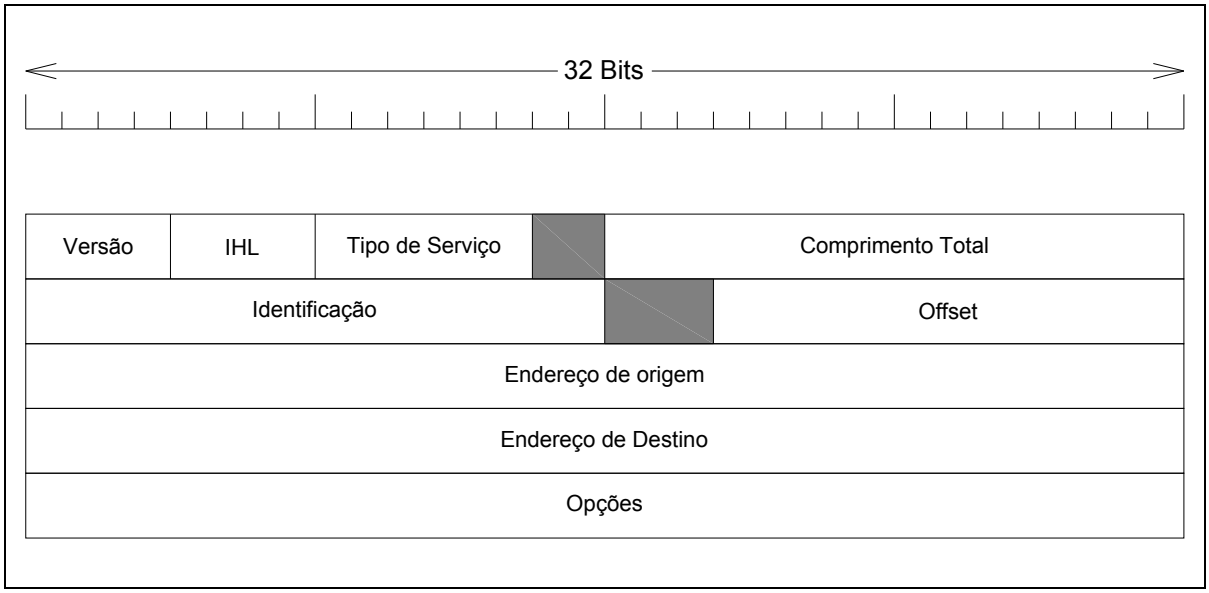


Figura 3.4 – Cabeçalho IP (versão 4)

3.2.2.1 Endereçamento IP

Como as redes de computadores interligam várias máquinas ao mesmo tempo, é necessário que sejam atribuídos endereços para cada máquina, de forma que possam se comunicar devidamente.

O endereço IP é um número inteiro de 32 bits que contém as informações relativas ao host e à rede em que o host está conectado. Cada host recebe um endereço IP distinto, sendo que os endereços das máquinas de uma mesma rede possuem o mesmo prefixo, ou seja, tal prefixo é o identificador da rede. [COMER, 1998]

Os endereços IP foram divididos em cinco classes: A, B, C, D e E. O formato de cada classe define a quantidade de hosts e redes possíveis, bem como se o endereço é usado para multidifusão (multicast) ou se é reservado para uso futuro. Os formatos dessas classes são vistos na figura 3.5. A classe A permite até 128 redes com 16 milhões de hosts cada, com endereços entre 1.0.0.0 e 127.255.255.255, a classe B permite 16.384 redes com 65.536 hosts, com endereços entre 128.0.0.0 e 191.255.255.255 e a classe C permite mais de 2 milhões de redes com 256 hosts cada, os endereços dos hosts desta classe vão de 192.0.0.0 a 223.255.255.255. Já os endereços da classe D, de 224.0.0.0 a 239.255.255.255, são utilizados para multidifusão e a classe E é reservada para uso futuro, onde os endereços vão de 240.0.0.0 a 247.255.255.255. [TANENBAUM,2003]

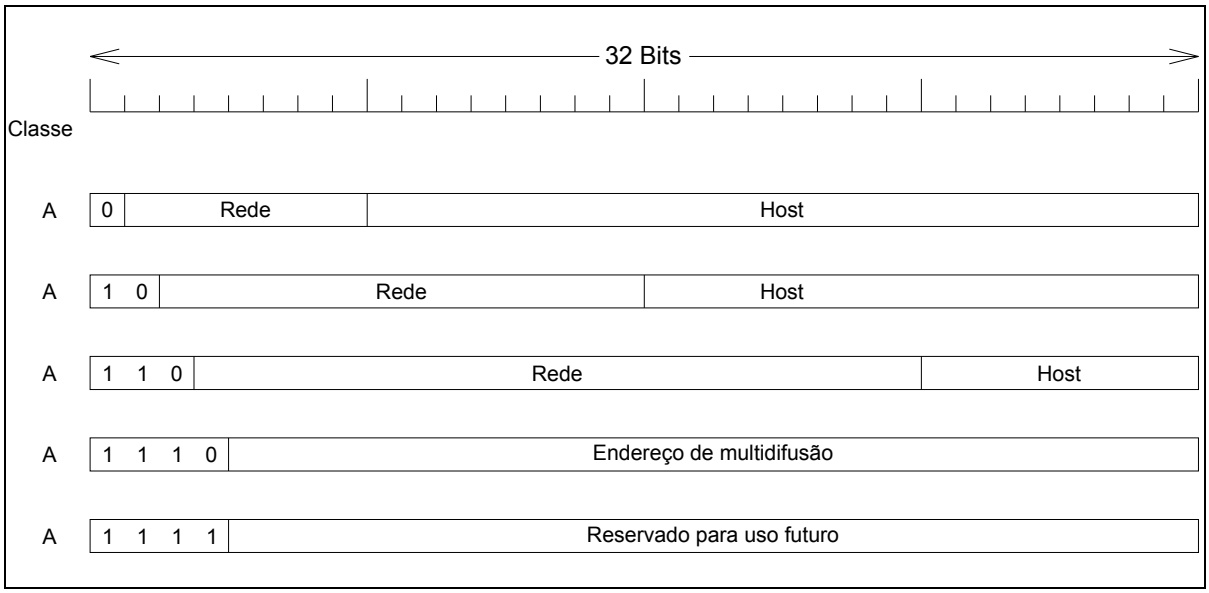


Figura 3.5 – Formato das classes IP

Os endereços IP definem conexões de rede e não a conexão de uma determinada máquina à rede, dessa forma, se uma máquina sai de uma rede e vai para outra, seu endereço deverá ser alterado de forma a se adequar à nova rede. [COMER, 1998] [TANENBAUM,2003]

3.2.2.2 Máscara de Sub-Rede

Todos os hosts de uma rede devem possuir o mesmo prefixo, ou seja, o mesmo endereço de rede. Porém, com o crescimento das redes isto pode se tornar um problema. A questão é que, segundo a regra, cada endereço das classes A, B ou C se refere a uma rede, e não a um conjunto de redes. Dessa forma a solução é dividir uma rede em várias partes para uso interno sem deixar de ser vista, externamente, como uma única rede. Assim surgiram as sub-redes, cada uma das partes em que a rede foi dividida é uma sub-rede. [TANENBAUM,2003]

Uma ilustração da divisão em sub-redes é apresentada na figura 3.6. No roteador principal da rede deverá ser implementada uma máscara de sub-rede, que irá indicar a divisão entre número da rede, sub-rede e host. Tais máscaras podem

ser escritas em notação decimal com pontos ou, de uma forma alternativa, com a indicação da quantidade de bits da máscara de sub-rede. [TANENBAUM,2003]

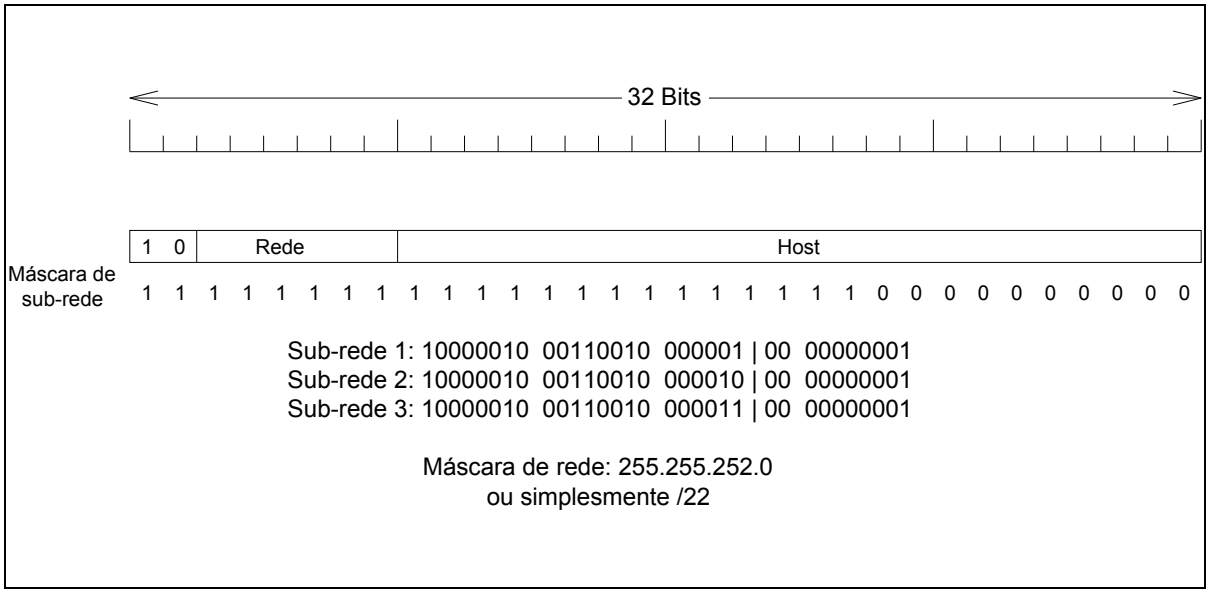


Figura 3.6 – Divisão de uma rede classe B em 64 sub-redes

3.3 ARP

ARP significa Address Resolution Proocol, ou Protocolo de Resolução de Endereço e é utilizado para associar endereços físicos a endereços IP. Fisicamente, as interfaces de rede não trabalham com endereços IP, mas sim com seus endereços físicos, determinados pelo fabricante de cada interface. Uma entidade central distribui faixas de endereços físicos a cada fabricante, assim, o endereço físico de cada interface deverá ser único, evitando possíveis conflitos. O endereço físico de uma interface Ethernet possui 48 bits. [TANENBAUM,2003]

Na figura 3.7 é apresentado o encapsulamento da mensagem ARP no quadro de transmissão da rede a nível físico. O protocolo ARP funciona da seguinte forma: um host precisa enviar um pacote a um outro host que contenha, por exemplo, o endereço 192.31.65.5. O host de origem envia um pacote de difusão (broadcast) perguntando a quem pertence o endereço 192.31.65.5, a mensagem chegará a todas as máquinas da rede, ou sub-rede, e somente a máquina que possuir o

endereço especificado na pergunta responderá à solicitação, enviando como resposta o endereço físico de sua interface Ethernet. Esse protocolo é definido na RFC 826 e a grande maioria das máquinas o executa.

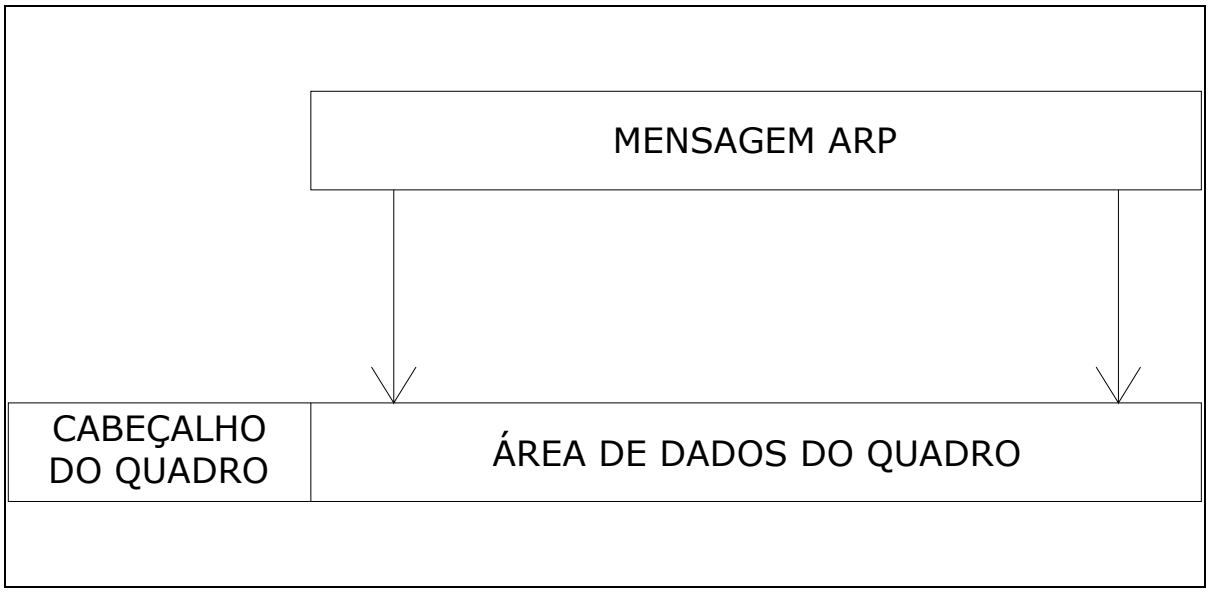


Figura 3.7 – Encapsulamento ARP

Algumas melhorias implementadas ajudam o protocolo ARP a ser ainda mais eficiente, tais como o armazenamento em cachê dos endereços físicos já obtidos, o envio do próprio endereço físico na mensagem de difusão e a difusão do seu mapeamento no momento em que a máquina for reinicializada.

O protocolo RARP funciona de modo análogo ao ARP, contudo, neste caso o endereço conhecido é o endereço físico e a solicitação procura pelo endereço IP.

A figura 3.8 mostra um exemplo do formato da mensagem ARP. Comer faz a descrição de cada campo conforme a lista abaixo.

- Tipo Hardware: Informa o tipo de interface física a receber a resposta. Há um valor específico para Ethernet.
- Tipo Protocolo: Informa o tipo do protocolo de alto nível do remetente. Há um valor específico para IP.
- Operação: Define o tipo de operação ARP, pode ser solicitação ou resposta ARP ou solicitação ou resposta RARP.
- HLEN: Extensão do endereço de hardware, usado em redes arbitrárias que necessitam dessa informação.

- PLEN: Extensão do endereço de protocolo de alto nível, usado em redes arbitrárias que necessitam dessa informação.
- Sender HA: Endereço de hardware do transmissor
- Sender IP: Endereço IP do transmissor
- Target HA: Endereço de hardware de destino, usado em RARP
- Target IP: Endereço IP de destino

<table>
<tr><td colspan="2">TIPO HARDWARE</td><td>TIPO PROTOCOLO</td></tr>
<tr><td>HLEN</td><td>PLEN</td><td>OPERAÇÃO</td></tr>
<tr><td colspan="3">SENDER HA (octetos de 0 a 3)</td></tr>
<tr><td colspan="2">SENDER HA (octetos de 4 e 5)</td><td>SENDER IP (octetos de 0 e 1)</td></tr>
<tr><td colspan="2">SENDER HA (octetos de 2 e 3)</td><td>TARGET HA (octetos de 0 e 1)</td></tr>
<tr><td colspan="3">TARGET HA (octetos de 2 a 5)</td></tr>
<tr><td colspan="3">TARGET IP (octetos de 0 a 3)</td></tr>
</table>

Figura 3.8 – Quadro ARP

3.4 Modelo de Referência OSI da ISO

Com a intenção de elaborar padrões internacionais, foi criada em 1946 a ISO (International Organization for Standardization). Fazem parte da ISO os órgãos de padronização de 89 países, sendo que os representantes do Brasil e dos Estados Unidos são a ABNT e o ANSI, respectivamente. [SOARES, 1995]

Os estudos da ISO com relação à padronização de sistemas de computação, visando sua interconexão, tiveram início em 1977. [TAROUCO, 1986] Foi criado então, o padrão internacional 7498, chamado Open Systems Interconection Reference Model, ou Modelo de Referência para Interconexão de Sistemas Abertos,

cujo objetivo é permitir o desenvolvimento coordenado de padrões para interconexão de sistemas baseado em uma referência comum. [SOARES, 1995]

Esse modelo representava um primeiro passo rumo à padronização internacional dos protocolos empregados em cada camada. Em 1995 o modelo foi revisto e é chamado Modelo de Referência ISO OSI. Atualmente os protocolos do modelo de referência OSI são pouco utilizados, mas o modelo em si ainda é válido e as características descritas para cada camada são muito importantes. Conforme a figura 3.9, o modelo de referência OSI possui 7 camadas: Física, Enlace de dados, Rede, Transporte, Sessão, Apresentação e Aplicação. [TANENBAUM,2003]

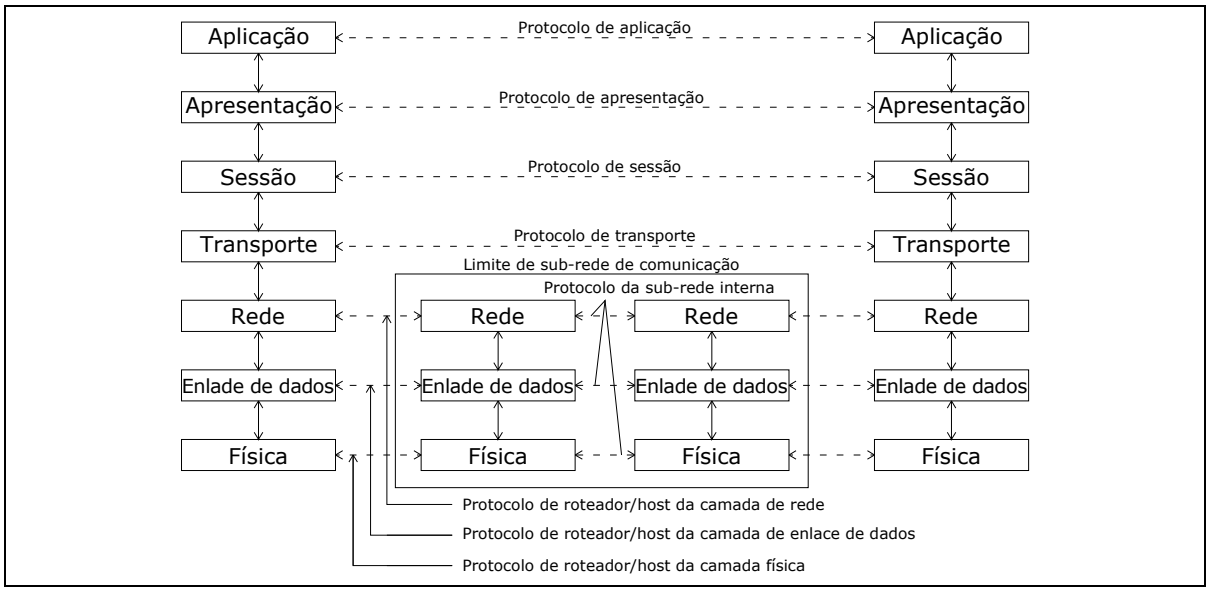


Figura 3.9 – Modelo de referência OSI

A camada Física define a transmissão dos bits pelo canal de comunicação. A camada de Enlace de Dados é responsável por fazer com que o canal de transmissão pareça uma linha livre de erros. A camada de Rede controla a operação da sub-rede, definindo o roteamento dos pacotes da origem até o destino. [TANENBAUM,2003]

Basicamente, a camada de Transporte recebe os dados da camada de Sessão, se for necessário divide-os, repassa-os à camada de Rede e garante que todos os fragmentos serão recebidos corretamente na outra extremidade. A camada de Sessão permite o estabelecimento de sessões entre usuários de diferentes máquinas. A camada de Apresentação trata da sintaxe e da semântica das

informações transmitidas. Por fim, a camada de Aplicação traz vários protocolos necessários aos usuários, como o HTTP, por exemplo. [TANENBAUM,2003]

3.5 Modelo de referência TCP/IP

Este modelo surgiu com a ARPANET e suas dificuldades para operar com seus protocolos nas redes de rádio e satélite. O principal objetivo do projeto era desenvolver habilidade para conectar várias redes de maneira uniforme. Como apresentado na figura 3.10, o modelo de referência TCP/IP possui apenas 4 camadas. [TANENBAUM,2003]

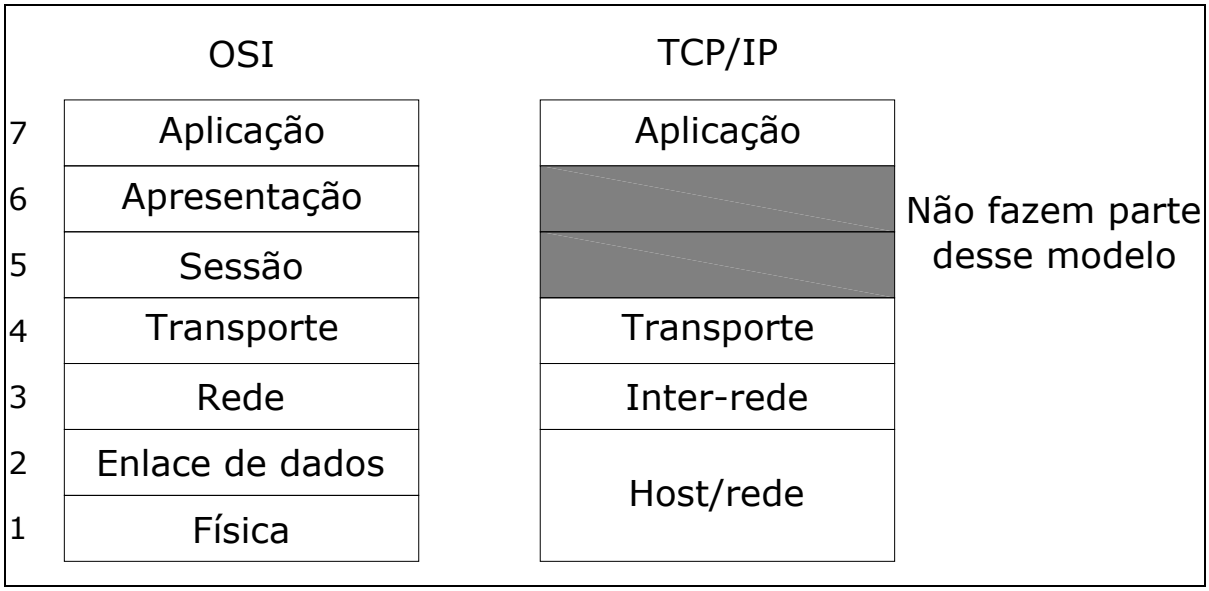


Figura 3.10 – Modelo de referência TCP/IP

A camada Host/Rede tem a função de compatibilizar as tecnologias específicas das interfaces com o protocolo IP. [SOARES, 1995] A camada Inter-redes é responsável pela transferência de dados através de qualquer rede e ainda garante que tais dados chegarão ao destino, mesmo que em ordem diferente. [TANENBAUM,2003]

A camada de Transporte permite a comunicação fim-a-fim entre aplicações. Nessa camada foram definidos os protocolos TCP e UDP. A camada de Aplicação

traz os protocolos de mais alto nível, como TELNET, FTP e HTTP. [TANENBAUM,2003] [SOARES, 1995]

3.6 Pilha TCP/IP

A Pilha TCP/IP da Microchip é um conjunto de programas que provê serviços para aplicações baseadas no padrão TCP/IP. Esta pilha é implementada em um formato modular, trazendo um alto grau de abstração de camadas. As facilidades implementadas por esta pilha fazem com que o programador não necessite de grandes conhecimentos em TCP/IP. [RAJBHARTI, 2002]

Muitas implementações de pilhas TCP/IP seguem a arquitetura do modelo de referência TCP/IP, apresentado na figura 3.11. Nesse modelo o software é dividido em camadas e cada camada acessa serviços de camadas mais inferiores. Por especificação, várias camadas TCP/IP permanecem sempre ativas, de forma a trabalhar, não somente quando são requisitadas, mas sempre que ocorrerem eventos como limite de tempo esgotado ou chegada de novos pacotes. [RAJBHARTI, 2002]

Um sistema com boa capacidade de memória de programa, memória de dados e processamento implementa facilmente essa arquitetura. Um sistema multitarefa facilita ainda mais essa implementação, mas quando se trata de um microcontrolador, onde memória de programa e memória RAM são escassas, essa tarefa se torna um pouco mais difícil. Sem um sistema multitarefa é preciso dar um pouco mais de atenção para que a pilha TCP/IP seja independente do programa principal. [RAJBHARTI, 2002]

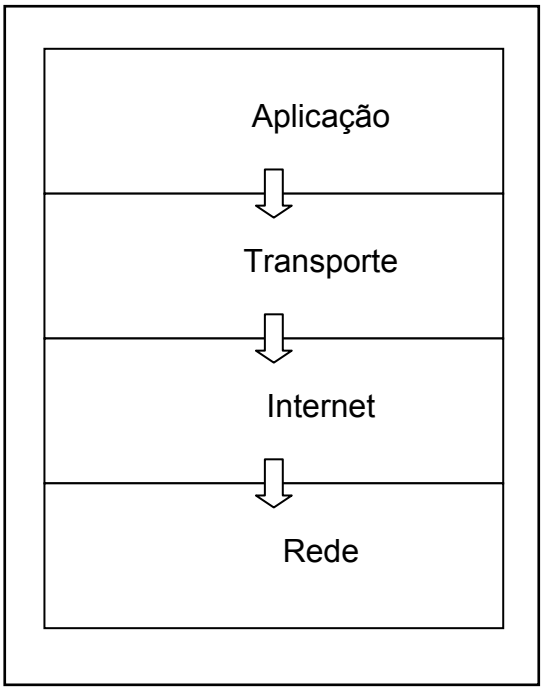


Figura 3.11 – Modelo de referência TCP/IP

Esta pilha é escrita em linguagem C de programação e pode ser compilada pelo compilador C18 da microchip ou pelo PICC 18 da Hi-Tech. Além disso ela foi projetada para trabalhar com toda a família de microcontroladores PIC18 da Microchip. Vale lembrar que, em se tratando de aplicação Ethernet, quanto mais memória Ram, memória de programa e capacidade de processamento o microcontrolador possuir, mais fácil será a implementação e utilização. [RAJBHARTI, 2002]

Assim como o modelo de referência TCP/IP, a pilha TCP/IP da Microchip é dividida em múltiplas camadas. O modelo dessa pilha TCP/IP é apresentado na figura 3.12. Os códigos que implementam cada camada ficam em arquivos fonte independentes. Um ponto que difere do modelo de referência é que, na pilha em questão, várias camadas têm acesso direto a outras camadas, que não precisam estar diretamente abaixo delas. A decisão de saltar ou não camadas adjacentes é feita de acordo com o montante de overhead ou com a necessidade de processamento inteligente de um serviço antes de ser passado adiante. Outro ponto importante de diferença é a existência de dois módulos, um usado para gerenciar as operações da pilha e de todos os seus módulos e outro para gerenciar a camada de endereçamento ARP. [RAJBHARTI, 2002]

Para se manter independente da aplicação principal e permanecer ativa, executando algumas operações temporizadas de forma assíncrona, a pilha TCP/IP da Microchip utiliza a técnica de cooperação multitarefa, em que cada tarefa executa seu trabalho e retorna seu controle, então a próxima tarefa poderá executar, por sua vez, o seu trabalho. Os dois módulos citados anteriormente são tarefas cooperativas. [RAJBHARTI, 2002] A figura 3.12 apresenta uma comparação entre o modelo de referência TCP/IP e a pilha TCP/IP da Microchip.

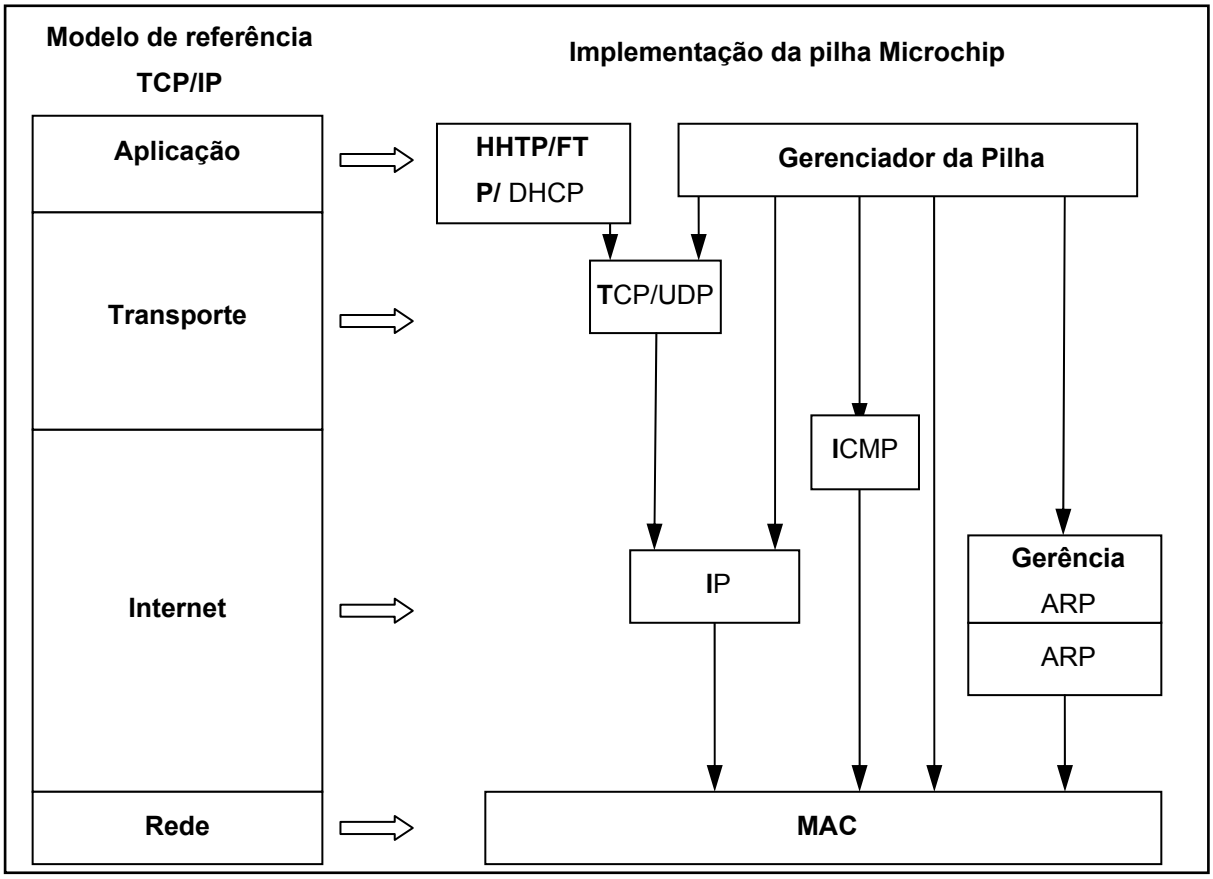


Figura 3.12 – Comparativo entre modelo de referência e pilha Microchip

3.7 Protocolo de Controle da Transmissão (TCP)

Na arquitetura da pilha TCP/IP da Microchip a camada TCP é uma camada ativa. Ela recebe os pacotes TCP e responde ao host remoto. O módulo TCP é

implementado como uma tarefa cooperativa, executando operações automáticas sem a necessidade do conhecimento da aplicação principal. [RAJBHARTI, 2002]

Como apresentado na figura 3.13, esta camada utiliza o trecho de código para implementar o serviço de sockets, permitindo de 2 a 253 sockets, onde esse número é limitado apenas pelo compilador utilizado e pela quantidade de memória. Contudo, quanto mais sockets são utilizados, mais lento o serviço pode ficar, visto que cada socket consome cerca de 36 bytes e aumenta o tempo total de processamento de pacotes TCP. [RAJBHARTI, 2002]

```
void TCPInit(void)
{
 TCP_SOCKET s;
 SOCKET_INFO* ps;

 // Initialize all sockets.
 for(s = 0; s < MAX_SOCKETS; s++)
 {
        ps = &TCB[s];

        ps->smState          = TCP_CLOSED;
        ps->Flags.bServer    = FALSE;
        ps->Flags.bIsPutReady  = TRUE;
        ps->Flags.bFirstRead   = TRUE;
        ps->Flags.bIsTxInProgress = FALSE;
        ps->Flags.bIsGetReady  = FALSE;
        if(ps->TxBuffer != INVALID_BUFFER)
        {
                MACDiscardTx(ps->TxBuffer);
                ps->TxBuffer     = INVALID_BUFFER;
        }
        ps->TimeOut          = TCP_START_TIMEOUT_VAL;
        ps->TxCount                          = 0;
 }
}
```

Figura 3.13 – Trecho de código da implementação de sockets

Esse modelo determina que, quando um pacote é recebido e uma tarefa precisa receber tal pacote, é necessário que a todo o processo de recepção seja

realizado no mesmo período de execução da tarefa, sendo impossível ler parte do pacote em um período de execução e o restante em outro. [RAJBHARTI, 2002]

De acordo com as especificações TCP, cada segmento TCP contém um checksum para verificar todo o pacote, inclusive a área de dados. Para reduzir a necessidade de memória RAM, a camada TCP determina que o armazenamento e o cálculo de checksum sejam feitos no controlador de interface de rede, utilizando, para isso a memória SRAM desse dispositivo. O trecho de código que faz a criação do cabeçalho TCP é apresentada na figura 3.14. [RAJBHARTI, 2002]

```
// TCP Header
typedef struct _TCP_HEADER
{
 WORD   SourcePort;
 WORD   DestPort;
 DWORD  SeqNumber;
 DWORD  AckNumber;

 struct   {
          unsigned char Reserved3     : 4;
          unsigned char Val           : 4;
 } DataOffset;

 union
 {
          struct              {
                    unsigned char flagFIN    : 1;
                    unsigned char flagSYN    : 1;
                    unsigned char flagRST    : 1;
                    unsigned char flagPSH    : 1;
                    unsigned char flagACK    : 1;
                    unsigned char flagURG    : 1;
                    unsigned char Reserved2  : 2;
          } bits;
          BYTE byte;
 } Flags;

 WORD   Window;
 WORD   Checksum;
 WORD   UrgentPointer;
```

Figura 3.14 – Implementação do Cabeçalho TCP

Alem disso, de acordo com a RFC793, a camada TCP implementa retry automático, ou seja, reenvio de informações caso o recebimento não seja confirmado. Essa opção pode ser habilitada ou não, se estiver habilitada, cada buffer de transmissão fica reservado até que seja recebido uma confirmação de recebimento (ACK), mas isso reduz consideravelmente a vazão de dados. [RAJBHARTI, 2002]

3.7.1 Gerenciamento da pilha

Como já foi dito, a pilha TCP/IP é um conjunto de módulos, e, qualquer aplicação precisa fazer as chamadas a estes módulos nos momentos corretos. Para que não seja necessário fazer esse controle no programa principal, é usado um módulo especial na camada de aplicação, o gerenciador da pilha. O gerenciador da pilha também é implementado como uma tarefa cooperativa, quando este recebe seu tempo de processamento, ele questiona a camada MAC se há pacotes de dados válidos. Quando um pacote é recebido, o gerenciador da pilha o decodifica e o encaminha para o módulo apropriado para que os dados sejam tratados. Para o gerenciador da pilha funcionar corretamente, ele precisa ser inicializado pelo programa principal, essa inicialização é apresentada no trecho de código na figura 3.15. [RAJBHARTI, 2002]

```
void StackInit(void)
{
    smStack                 = SM_STACK_IDLE;

#if defined(STACK_USE_IP_GLEANING) || defined(STACK_USE_DHCP)
    /*
     * If DHCP or IP Gleaning is enabled,
     * startup in Config Mode.
     */
    AppConfig.Flags.bInConfigMode = TRUE;

#endif

    MACInit();

    ARPInit();

#if defined(STACK_USE_UDP)
    UDPInit();
#endif

#if defined(STACK_USE_TCP)
    TCPInit();
#endif

}
```

Figura 3.15 – Inicialização da pilha

3.8 O Servidor http

O servidor HTTP é, na verdade, um mini-servidor projetado para sistemas embarcados, também é uma aplicação desenvolvida como tarefa cooperativa. Conforme Rajbharti, este servidor possui as seguintes características:

- Suporta múltiplas conexões HTTP

- Trabalha com um modelo simplificado de dados
- Suporte a páginas Web gravadas na memória de programa do microcontrolador
- Suporte ao método GET
- Suporte a CGI (Common Gateway Interface) para executar funções solicitadas pelo browser remoto
- Suporte a geração dinâmica de conteúdo da página Web

A página padrão para este servidor é o arquivo “index.htm”, dessa forma, se algum usuário se conectar ao servidor através do seu endereço IP ou somente do seu nome de domínio, o servidor irá fornecer o arquivo “index.htm”. [RAJBHARTI, 2002]

Os nomes das páginas Web não podem conter os seguintes caracteres: ‘, ”, <, >, #, %, [,], {, }, |, \, ^ e ~. Caso uma página possua um destes caracteres, ela ficará inacessível e não haverá nenhuma informação sobre o erro. [RAJBHARTI, 2002]

O servidor HTTP pode alterar as páginas Web dinamicamente, substituindo informações em tempo real, tais como informações de status das entradas ou saídas do microcontrolador. Para incorporar as informações em tempo real são utilizados os arquivos CGI, que devem conter informações do tipo %XX, onde o caractere % funciona como um controle e a porção XX como um identificador de variáveis. A utilização desses arquivos CGI proporciona a visualização de informações em formulários de páginas Web. O trecho de código a seguir, apresentado na figura 3.16 é responsável por identificar a utilização de conteúdo dinâmico na página Web fazendo uma pesquisa. [RAJBHARTI, 2002]

```
if(ph->bProcess)
{
   while(TCPIsPutReady(ph->socket))
   {
      lbTransmit = FALSE;

      if(ph->smHTTPGet != SM_HTTP_GET_VAR)
      {
         c = MPFSGet();
         if(MPFSIsEOF())
         {
            MPFSGetEnd();
            TCPFlush(ph->socket);
            return TRUE;
         }
      }
                  switch(ph->smHTTPGet)
                  {
                  case SM_HTTP_GET_READ:
                     if ( c == HTTP_VAR_ESC_CHAR )
                        ph->smHTTPGet = SM_HTTP_GET_DLE;
                     else
                        lbTransmit = TRUE;
                     break;
                  case SM_HTTP_GET_DLE:
                     if ( c == HTTP_VAR_ESC_CHAR )
                     {
                        lbTransmit = TRUE;
                        ph->smHTTPGet = SM_HTTP_GET_READ;
                     }
                     else
                     {
                        HexNumber.v[1] = c;
                        ph->smHTTPGet = SM_HTTP_GET_HANDLE;
                     }
                     break;
                  case SM_HTTP_GET_HANDLE:
                     HexNumber.v[0] = c;
```

Figura 3.16 – Trecho da implementação de páginas dinâmicas

3.9 Sistema de arquivos MPFS

Como a memória para armazenamento da página web é bastante limitada, é necessário que seja utilizado um sistema de arquivos simplificado, que ocupe pouco espaço além dos dados. Por isso é utilizado o MPFS (Microchip File Sistem), que pode ser gravado na memória de programa do microcontrolador. O trecho que cria a estrutura para armazenar esse tipo de arquivo na memória de programa é visto na figura 3.17. [RAJBHARTI, 2002]

```
#ifdef MPFS_USE_PGRM
typedef struct  _MPFS_ENTRY
{
   BYTE Flag;
   MPFS Address;
   BYTE Name[MAX_FILE_NAME_LEN];
} MPFS_ENTRY;
```

Figura 3.17 – Criação da estrutura MPFS

O MPFS obedece a um formato especial de armazenamento de múltiplos arquivos, criando um único arquivo compacto. Esse formato pode ser visto na Figura 3.18. [RAJBHARTI, 2002]

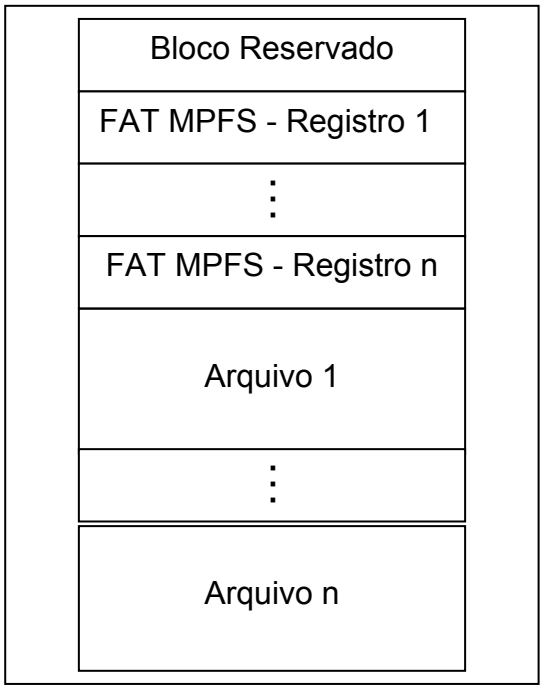


Figura 3.18 – Formato de armazenamento MPFS

O bloco reservado pode ser usado pelo programa principal para armazenar algumas informações sobre valores de configuração. O MPFS inicia com o armazenamento de uma ou mais tabelas de alocação de arquivo (FAT), seguido de um ou mais arquivos de dados. Cada FAT descreve o nome do arquivo, sua localização e seu status. O formato do bloco FAT é apresentado na figura 3.19. [RAJBHARTI, 2002]

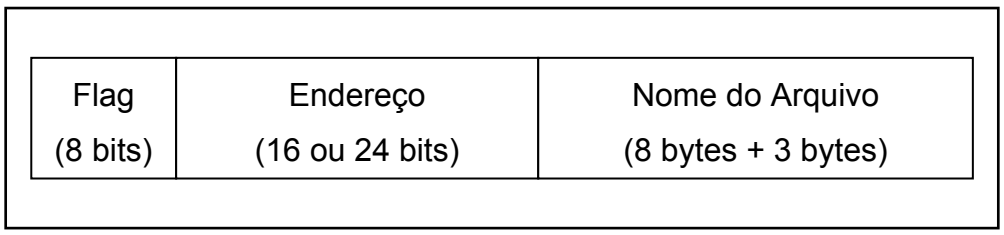


Figura 3.19 – Formato do registro FAT MPFS

O Flag indica se o registro está em uso, se foi apagado ou se está no final da tabela FAT. Cada registro FAT pode conter um campo de endereçamento de 16 ou 24 bits, que é determinado de acordo com a memória a ser utilizada, bem como seu espaço disponível. O campo ‘Nome do arquivo’ contém o nome do arquivo em

questão. Este campo obedece ao padrão de 8 bytes para o nome do arquivo e 3 bytes para a extensão. [RAJBHARTI, 2002]

O endereço armazenado em cada registro FAT aponta para um bloco de dados que contém o arquivo correspondente. Conforme visto na figura 3.20, o bloco de dados é composto de três partes, Dados, EOF e Confirmação de fim. O campo Dados é de tamanho variável e contém os dados correspondentes ao arquivo. Os dois campos seguintes compõem o terminador, o EOF (End Of File) seguido por FFFFh ou FFFFFFh indicam o final do arquivo. [RAJBHARTI, 2002]

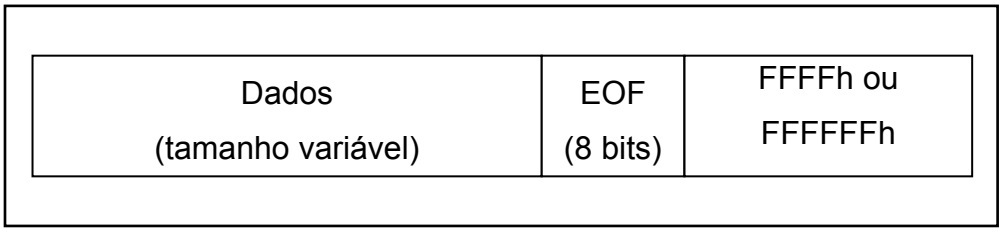


Figura 3.20 – Formato do bloco de dados MPFS

Para criar as imagens de arquivo MPFS é utilizado um aplicativo que agrupa um conjunto de arquivos em um único arquivo MPFS, assim, antes de criar o arquivo, é preciso criar todo o conteúdo da página Web que será gravada no microcontrolador e armazena-lo em uma única pasta de arquivos. Depois disso basta executar o programa para obter a imagem MPFS. [RAJBHARTI, 2002]

3.10 Módulo principal

Este módulo é responsável por controlar o funcionamento dos sensores e atuadores, bem como da atualização da página Web. Neste ponto serão apresentados os principais pontos do programa, suas características e seu funcionamento.

É neste módulo que se encontram o loop principal, as chamadas de inicialização das funções e as definições dos endereços dos arquivos CGI, bem como toda a configuração do microcontrolador. Em primeiro lugar, são definidas

variáveis que serão utilizadas por outras funções, tais como as variáveis do endereçamento IP, apresentadas na figura 3.21.

```
APP_CONFIG AppConfig =
{
    {MY_DEFAULT_IP_ADDR_BYTE1, MY_DEFAULT_IP_ADDR_BYTE2,
MY_DEFAULT_IP_ADDR_BYTE3, MY_DEFAULT_IP_ADDR_BYTE4},
    {MY_DEFAULT_MAC_BYTE1, MY_DEFAULT_MAC_BYTE2,
MY_DEFAULT_MAC_BYTE3, MY_DEFAULT_MAC_BYTE4, MY_DEFAULT_MAC_BYTE5,
MY_DEFAULT_MAC_BYTE6},
    {MY_DEFAULT_MASK_BYTE1, MY_DEFAULT_MASK_BYTE2,
MY_DEFAULT_MASK_BYTE3, MY_DEFAULT_MASK_BYTE4},
    {MY_DEFAULT_GATE_BYTE1, MY_DEFAULT_GATE_BYTE2,
MY_DEFAULT_GATE_BYTE3, MY_DEFAULT_GATE_BYTE4},
    {MY_DEFAULT_DNS_BYTE1, MY_DEFAULT_DNS_BYTE2,
MY_DEFAULT_DNS_BYTE3, MY_DEFAULT_DNS_BYTE4},
    {0b00000001},  // Flags, enable DHCP
};
```

Figura 3.21 – Variáveis do endereçamento IP

Outro trecho muito importante é apresentado na figura 3.22, onde é feita a configuração do microcontrolador. Essa configuração é feita no processo de gravação do dispositivo, para ser alterada é necessário regravar o microcontrolador com as configurações desejadas. Cada item tem seu significado descrito a seguir.

- OSC=HS: oscilador do tipo HS (Hi Speed ou Alta Velocidade), suporta freqüências entre 4 e 25MHz.
- WDT=OFF: Desliga o Watch Dog Timer
- MCLRE=ON: Define o pino 1 como entrada de sinal de reset.
- PBADEN=OFF: Define as portas de entrada como digitais, dentre outros.
- LVP=OFF: Desliga a gravação por baixa tensão.
- XINSY=OFF: Desativa o conjunto de instruções extendidas

```
#elif defined(__18F4620)
        // PICDEM.net board
        #pragma config OSC=HS, WDT=OFF, MCLRE=ON, PBADEN=OFF, LVP=OFF,
XINST=OFF
```

Figura 3.22 – Configuração do microcontrolador

Como todas as aplicações utilizadas precisam ser inicializadas, uma parte do código é destinada a esse processo, isso é visto na figura 3.23. Neste posto são inicializadas as configurações de portas do microcontrolador, as funções relativas ao gerenciamento da pilha, a aplicação de arquivos MPFS e o servidor HTTP.

```
InitializeBoard();

TickInit();

MPFSInit();

memcpypgm2ram(AppConfig.NetBIOSName,(ROM void*) MY_DEFAULT_HOST_NAME, 16);
 FormatNetBIOSName(AppConfig.NetBIOSName);
InitAppConfig();

if(BUTTON0_IO == 0)
  {
    SetConfig();
  }

StackInit();

#if defined(STACK_USE_HTTP_SERVER)
  HTTPInit();
```

Figura 3.23 – Inicialização das aplicações

Com todas as aplicações inicializadas, é hora de entrar no loop principal. Esse loop é feito com a função “while(1)”, que faz com que o loop seja infinito. Nesse loop estão os comandos que fazem o LED piscar, indicando atividade, a chamada ao gerenciador da pilha, para que seja verificada a comunicação através da rede, o servidor http, que atualiza a página Web e a disponibiliza e as funções executadas pelo microcontrolador, que são a verificação dos sensores e o controle dos

atuadores. Trechos desse código podem ser vistos na figura 3.24. As aplicações específicas de controle dos sensores e atuadores são realizadas em uma função distinta, nesse ponto ela é apenas chamada.

Dessa forma, o loop principal divide o tempo de processamento entre as várias aplicações necessárias, chamando-as uma a uma e, como cada função é preparada para trabalhar em conjunto com várias outras em regime cooperativo, não é necessário executar outras ações além das chamadas.

```
while(1)   {
    if ( TickGetDiff(TickGet(), t) >= TICK_SECOND/2 )
    {
       t = TickGet();
       LED0_IO ^= 1;
    }
    StackTask();
#if defined(STACK_USE_HTTP_SERVER)
    HTTPServer();
#endif
    ProcessIO();
```

Figura 3.24 – Loop principal

A função ProcessIO define o funcionamento do microcontrolador em relação aos sensores, atuadores e comandos através de CGI. Alguns trechos dessa função podem ser vistos na figura 3.25, tais como o cálculo de temperatura, os endereçamentos e comandos CGI.

```
float Temperature;
Temperature = ((float)(ADC1BUF0)*(3.3/1024.)-0.500)*100.;
sprintf(AN1String, "%3.1f°C", Temperature);
itoa((unsigned)ADC1BUF0, AN0String);
#define VAR_ANAIN_AN0       (0x02)
#define VAR_ANAIN_AN1       (0x03)
#define SENSOR0             (0x04)
#define SENSOR1             (0x0D)
#define RELE1               (0x0)
#define RELE2               (0x1)
```

Figura 3.25 – Definição de variáveis e comandos

3.11 Página Web

Outro ponto importante do projeto é a página Web. É através dela que o usuário poderá interagir com o dispositivo, tanto para visualizar as entradas quanto para disparar os acionadores. Esta página fica armazenada na memória de programa do microcontrolador e pode ser acessada como se estivesse em um servidor comum. Como visto nas figuras 3.26, 3.27 e 3.28, a página Web é composta por um cabeçalho, um campo para texto, os botões para controle dos acionadores e a área de status.

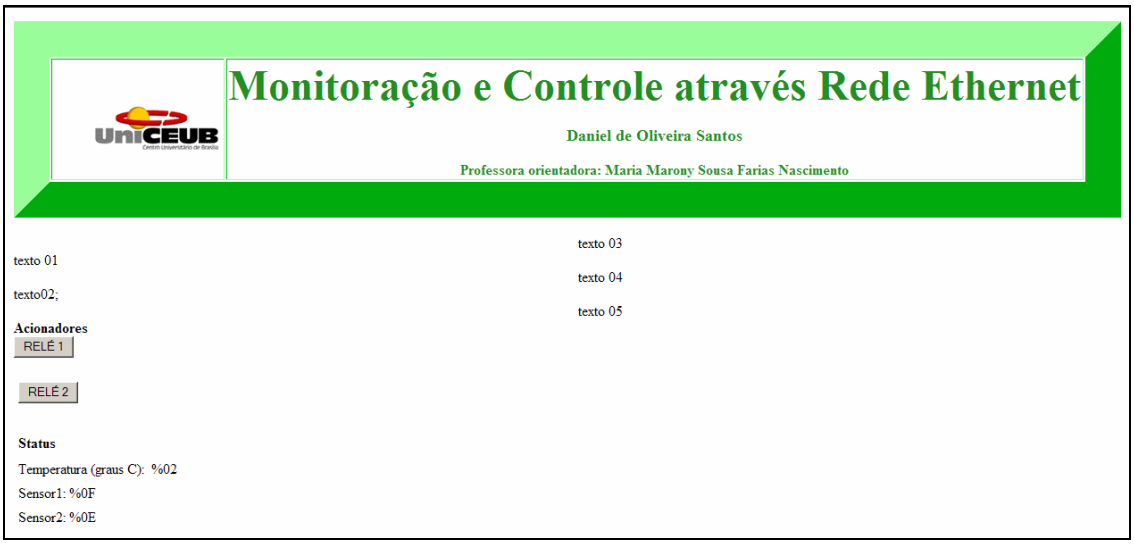


Figura 3.26 –Página Web

O cabeçalho contém o título do projeto, o nome do autor e o nome da professora orientadora. O campo de texto fica reservado para utilizações posteriores como descrição do funcionamento. O campo dos acionadores traz dois botões, que servem para ativar e desativar cada um dos acionadores. Já o campo status apresenta as informações dos sensores de contato seco e de temperatura.

Figura 3.27 – Detalhe do cabeçalho da página Web

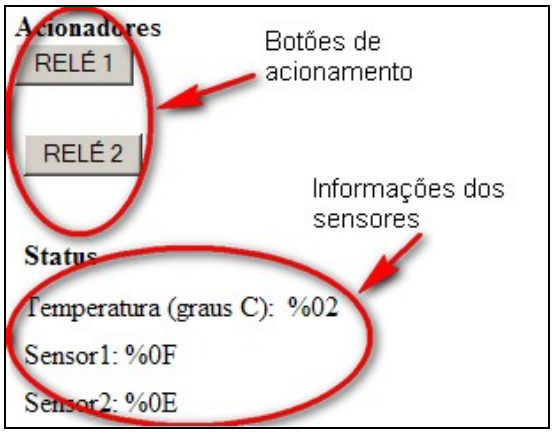


Figura 3.28 – Detalhe de outros trechos da página Web

Na figura 3.29 é apresentado um trecho do código fonte da página Web, esse é o ponto onde são definidos os botões e a área de status. A programação foi feita em HTML e utiliza a facilidade dos arquivos CGI para fazer a atualização das informações e o envio de comando através dos botões.

```
<table cellpadding="3">
<form>
<tr>
<p align="center" color="#99FF99">
<input type="button"value="RELÉ 1"
onclick="GetServerFile('0?1=LED2','')">
</input>
</p>
</tr>
<td>
<input type="button" value="RELÉ 2"
onclick="GetServerFile('0?0=LED1','')"></input>
</td>
</tr>
</form>
<tr>
<td> </td>
</tr><tr>
<td>
<b>Status</b>
</td>
</tr>
</table>
```

Figura 3.29 – Trecho do código fonte da página Web

Capítulo 4. Testes e Resultados

Até este ponto foi vista toda a implementação de hardware e software. Durante o desenvolvimento vários testes foram feitos, algumas tentativas falharam outras tiveram resultados positivos e, dessa forma, o projeto foi elaborado.

Este capítulo apresenta todos os testes que foram realizados no desenvolvimento do projeto, bem como os resultados obtidos. Para facilitar a compreensão, o capítulo está organizado em três tópicos:

- Testes na implementação
- Imagens de osciloscópio
- Testes finais

Cada teste é acompanhado dos resultados obtidos e das ações tomadas, cada imagem de osciloscópio também traz seus comentários particulares e os testes finais também são detalhados. A figura 4.1 apresenta a bancada (improvisada) onde os testes foram realizados.

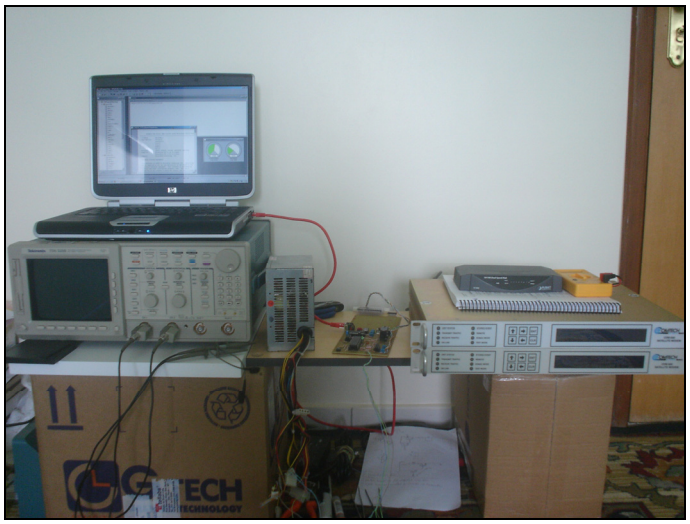


Figura 4.1 – Bancada de testes

4.1 Testes na implementação

Aqui são apresentados os testes que foram realizados durante a implementação. Esses testes foram conclusivos para a definição do hardware e do funcionamento do circuito.

4.1.1 Sensores de contato seco

O sensor de contato seco foi desenvolvido utilizando-se uma porta lógica do tipo NAND, tanto para facilitar a expansão quanto para proteger a entrada do microcontrolador, visto que este possui um custo relativamente alto.

Inicialmente o sensor foi implementado como mostra a figura 4.2, usando um resistor de pull-down e enviando um sinal positivo.

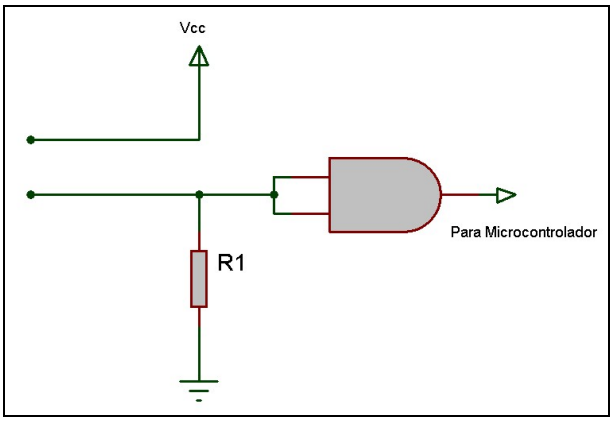


Figura 4.2 – Primeira versão do sensor de contato seco

Dessa forma, a saída da porta lógica se manteria em nível alto a maior parte do tempo e iria para nível baixo quando houvesse a detecção de contato. Nesse caso a lógica do microcontrolador deveria ser invertida, ou seja:

- Nível lógico alto: sistema normal
- Nível lógico baixo: contato detectado

Outro problema seria a possibilidade de acontecer um curto-circuito entre o fio vindo do positivo da fonte e um ponto de aterramento fora do circuito, o que poderia causar danos à fonte de alimentação e, consequentemente, a parada do funcionamento do circuito.

A solução foi substituir o resistor de pull-down, por um de pull-up, além disso, ao contrário de se usar um sinal vindo do positivo da fonte, foi usado um sinal vindo do negativo da fonte, formando-se o circuito da figura 4.3, que é o sensor de contato seco definitivo.

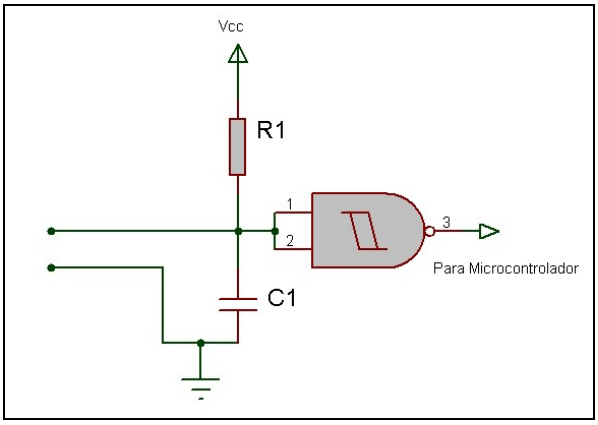


Figura 4.3 – Sensor de contato seco definitivo

4.1.2 Acionador tipo 2

Os acionadores são baseados em relés, estes necessitam de uma corrente constante em suas bobinas para que se mantenham chaveados. No caso do relé utilizado neste acionador, a corrente necessária é de 28mA. Nem o microcontrolador nem a porta lógica NAND do CI 4093 tem essa capacidade de corrente. A título de confirmação, foi feita uma tentativa de acionar o relé utilizando-se apenas uma porta lógica NAND do CI 4093, contudo o resultado foi negativo. A figura 4.4 mostra o circuito que solucionou o problema.

Portanto, foi necessário utilizar um transistor para fornecer a corrente suficiente ao relé. Como o CI 4093 possui quatro portas lógicas NAND e somente duas foram utilizadas nos sensores, optou-se por manter uma porta lógica entre a

saída do microcontrolador e a base do transistor, evitando que um eventual curto-circuito venha danificar o pino de saída do microcontrolador.

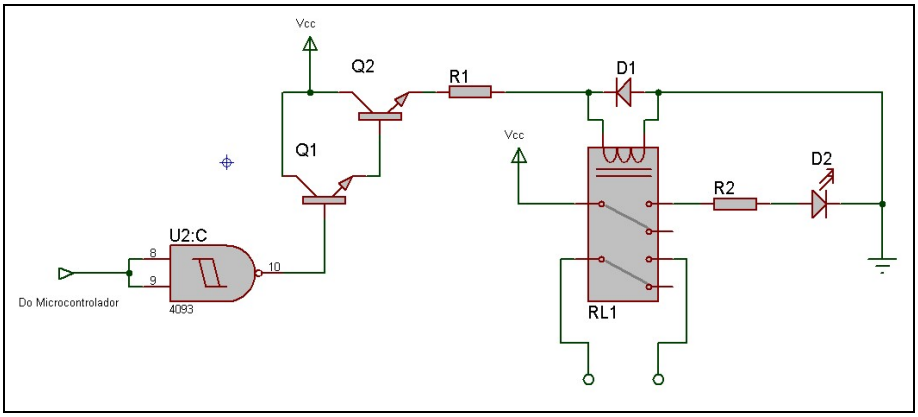


Figura 4.4 – Acionador tipo 2 definitivo

4.1.3 Acionador tipo 1

Este acionador teve o mesmo problema citado no item anterior, mas como seu relé requer uma corrente ainda maior, cerca de 100mA, um único transistor não foi suficiente para acioná-lo. Seguindo a recomendação da professora orientadora, foi utilizado um par de transistores, montados em uma configuração chamada “Par Darlington”. Isso foi suficiente para solucionar o problema. O circuito final é apresentado na figura 4.5.

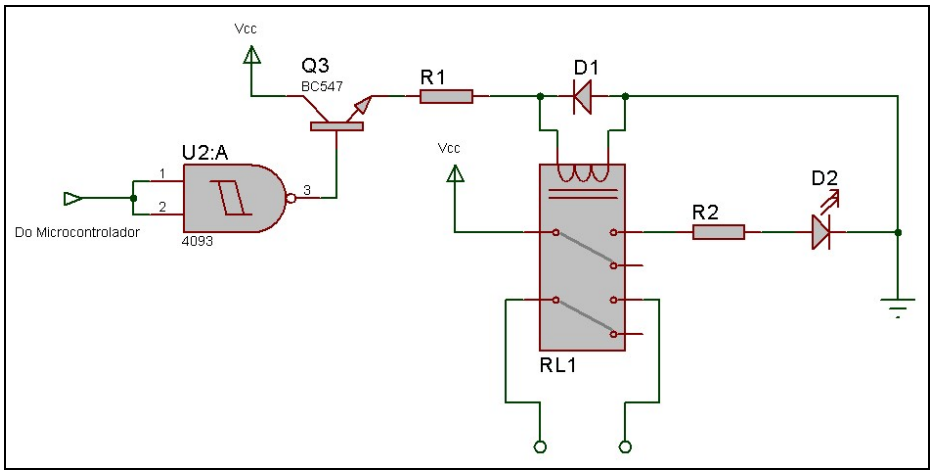


Figura 4.5 – Acionador tipo 1 final

4.1.4 Ausência de aterramento da fonte

A fonte de alimentação do circuito é uma fonte comum de computador. Todos os testes foram realizados com esta fonte ligada a uma tomada AC tripolar, contendo fase, neutro e terra. Contudo, em um dos testes, a fonte foi ligada com uma extensão que não possuía fio terra. Inicialmente o circuito funcionou, mas quando o microcontrolador ou outras partes do circuito eram tocadas com os dedos, ocorria uma instabilidade, ás vezes desligando o microcontrolador, às vezes acionando o relé várias vezes por segundo. A solução foi utilizar sempre tomadas com aterramento.

4.1.5 Botão de reset

Inicialmente o microcontrolador não contava com um botão de reset. Com a seqüência de testes realizados, foram identificados alguns travamentos ao ligar o circuito. Para evitar a necessidade de desligar e religar a fonte de alimentação, foi instalado um botão de reset conforme recomenda o datasheet do microcontrolador. Esta alteração é apresentada na figura 4.6.

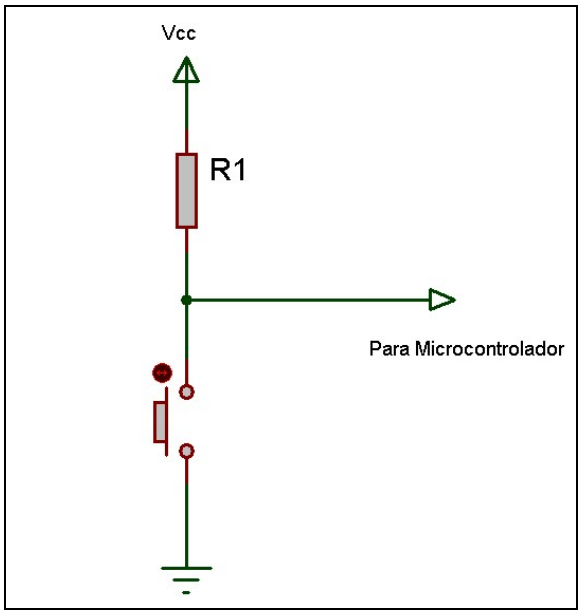


Figura 4.6 – Botão de reset

4.2 Imagens de osciloscópio

Neste tópico são apresentados os vários sinais que trafegam no circuito, desde os mais simples até os sinais de comunicação. As imagens aqui apresentadas foram extraídas do circuito através de um osciloscópio Tektronix modelo TDS 520B.

4.2.1 Sensor de contato seco

Na figura 4.7 é apresentada uma transição de nível alto para nível baixo na entrada do sensor. Essa transição ocorre quando o contato seco do equipamento monitorado se fecha. Para se capturar esta imagem foi simulada uma falha, com o fechamento dos contatos do sensor.

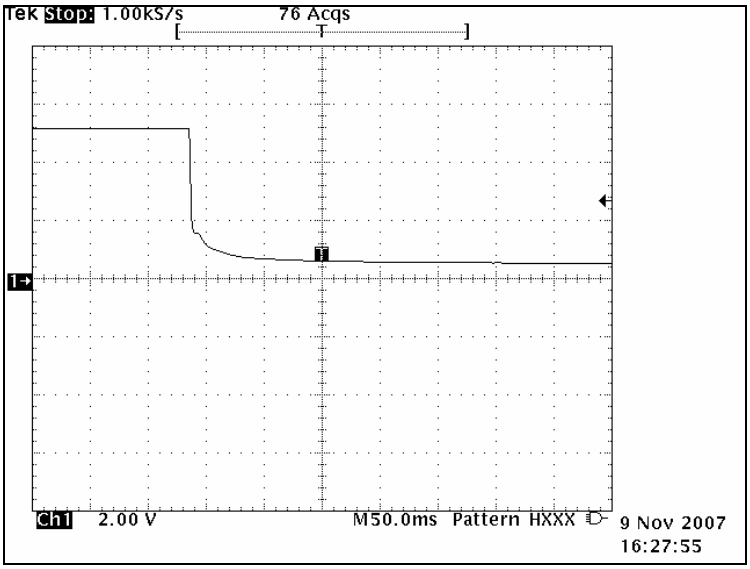


Figura 4.7 – Transição na entrada do sensor

4.2.2 Sensor de temperatura

O sensor de temperatura gera uma tensão analógica na saída. A imagem apresentada na figura 4.8 foi obtida na saída do sensor, que foi aquecido rapidamente e em seguida resfriado sem ventilação.Por isso a reta produzida apresenta uma inclinação acentuada no início e um caimento mais suave no final.

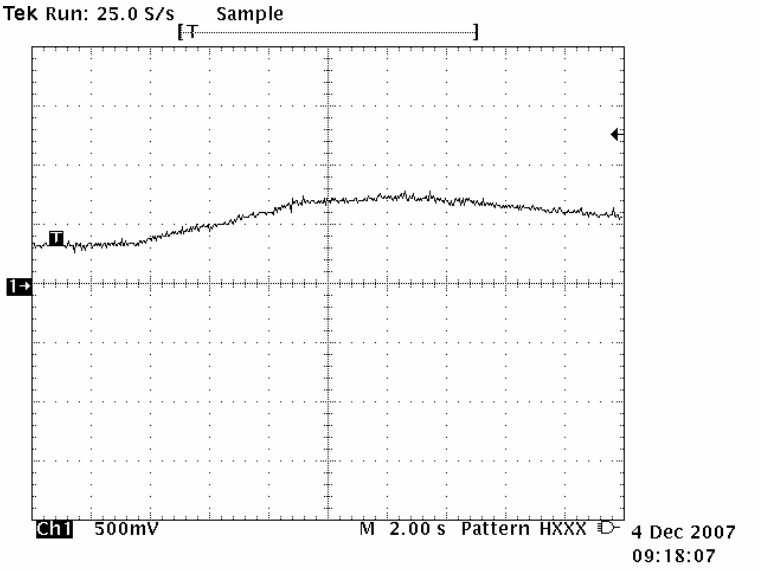


Figura 4.8 – Sinal do sensor de temperatura

4.2.3 Acionamento dos relés

Na figura 4.9 é apresentada a transição de nível baixo para nível alto que executa o acionamento do relé. A medida foi realizada na saída da porta lógica que aciona o transistor. Para gerar a transição, foi executado o comando de acionamento do relé na página Web, o microcontrolador processou o comando e colocou nível baixo na entrada da porta lógica. Dessa forma, antes do comando o nível é baixo e após o comando o nível passa a alto, acionando o transistor, que por sua vez aciona o relé.

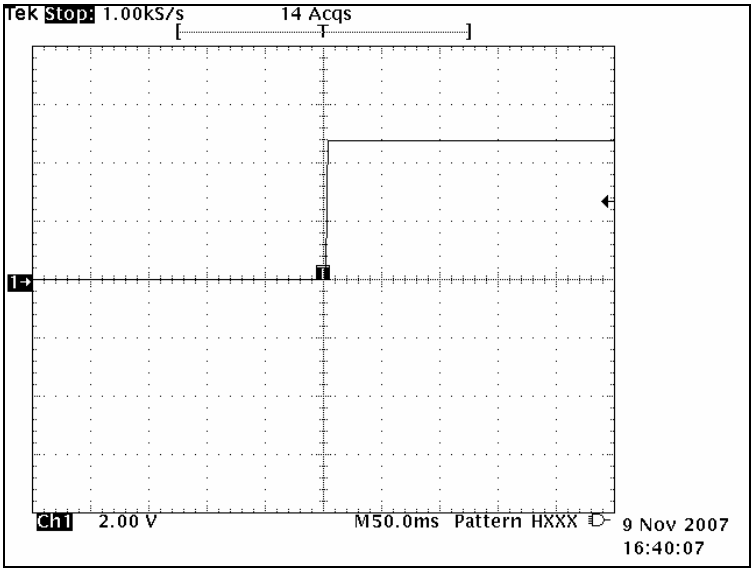


Figura 4.9 – Sinal de acionamento do relé

4.2.4 Clock do microcontrolador

O clock do microcontrolador, gerado por um cristal externo, é apresentado na figura 4.10, onde pode ser vista a freqüência de 19.844,2kHz, que corresponde a um período de 50,4ns. A freqüência recomendada pela microchip é de 20MHz, mas neste caso foi visto que há uma pequena variação, tanto na freqüência quanto na amplitude, porém sem prejudicar o funcionamento.

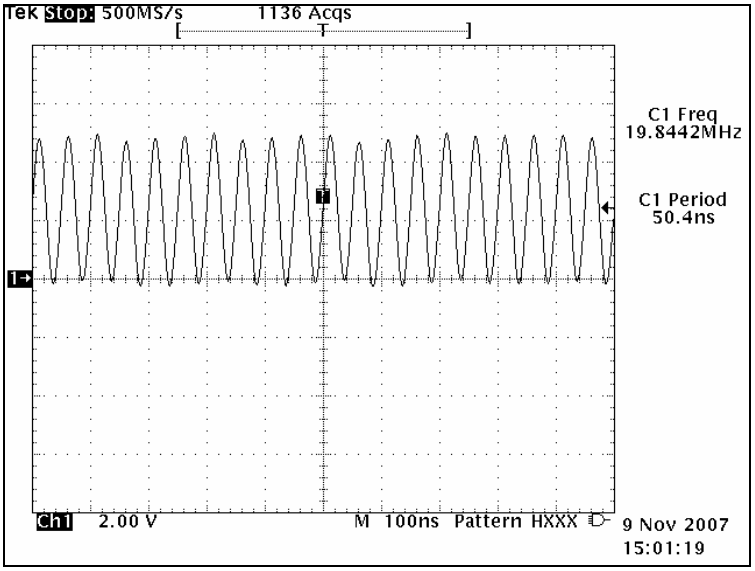


Figura 4.10 – Sinal de clock do microcontrolador

4.2.5 Clock do ENC28J60 (interface de rede)

Assim como o microcontrolador, a interface de rede também possui um sinal de clock vindo de um cristal externo, este é visto na figura 4.11. Sua freqüência é de 25.046,4kHz, correspondendo a um período de 40ns. Da mesma forma que no microcontrolador, o clock apresenta uma pequena variação de freqüência e de amplitude, sem prejudicar o funcionamento.

Acredita-se que a variação na amplitude seja provocada pelas variações presentes na fonte de alimentação.

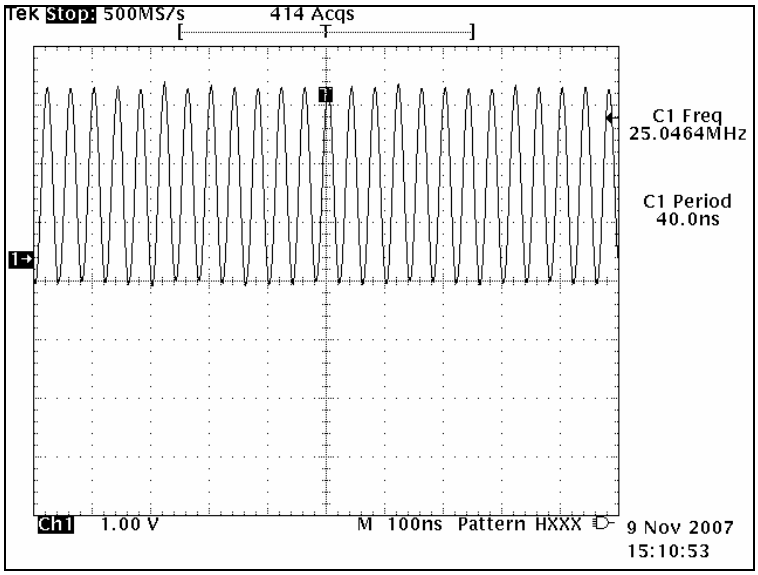


Figura 4.11 – Sinal de clock da interface de rede

4.2.6 Comunicação SPI

Como a comunicação SPI trabalha com três vias, clock TX e RX, neste ponto são apresentadas duas imagens, sendo que a figura 4.12 mostra o sinal de Transmissão de dados juntamente com o clock, enquanto a figura 4.13 mostra o sinal de Recepção de dados juntamente com o clock.

Deve-se observar que os sinais de dados estão na parte superior da imagem, enquanto o clock está na parte inferior. Outra questão é que o sinal de clock não é constante, ocorre apenas quando os dados são transmitidos. Estes sinais também apresentam os ruídos que possivelmente são gerados pela fonte de alimentação.

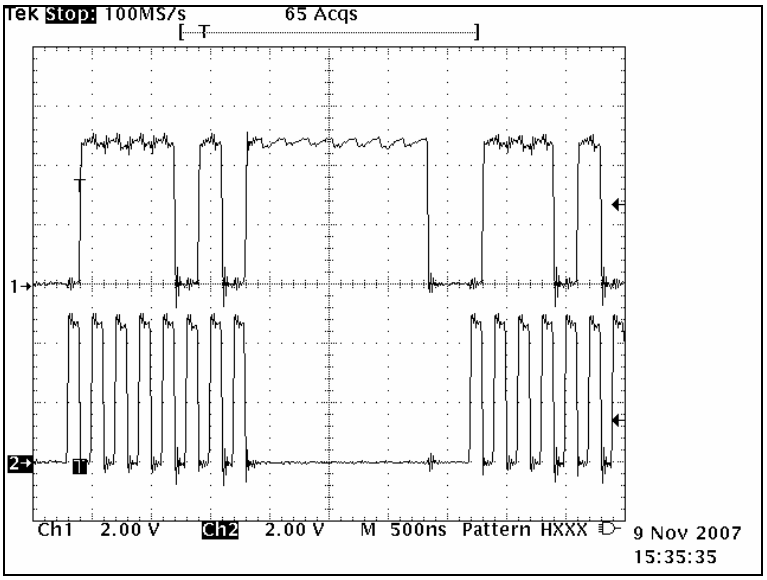


Figura 4.12 – Interface SPI – Transmissão de dados e clock

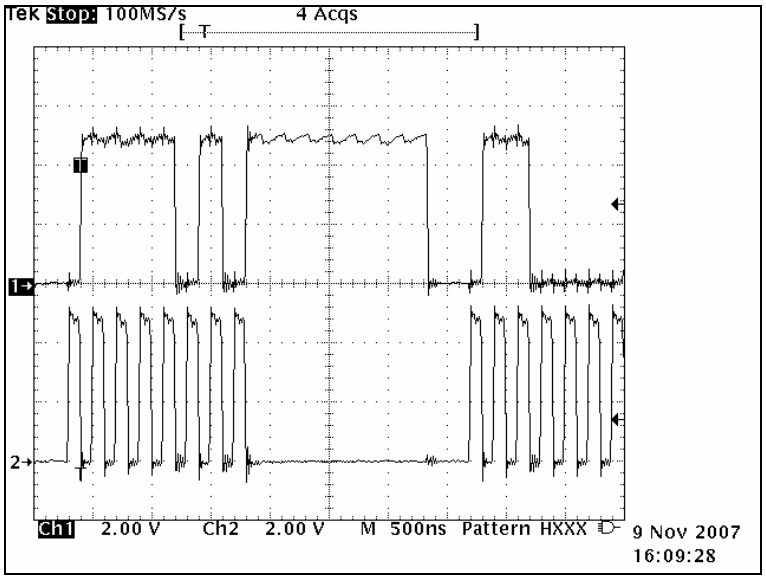


Figura 4.13 – Interface SPI – Recepção de dados e clock

4.2.7 Rede Ethernet

Como os dados são transmitidos através de rede ethernet, na figura 4.14 é apresentado, a título de ilustração, o sinal presente na via de transmissão da interface Ethernet.

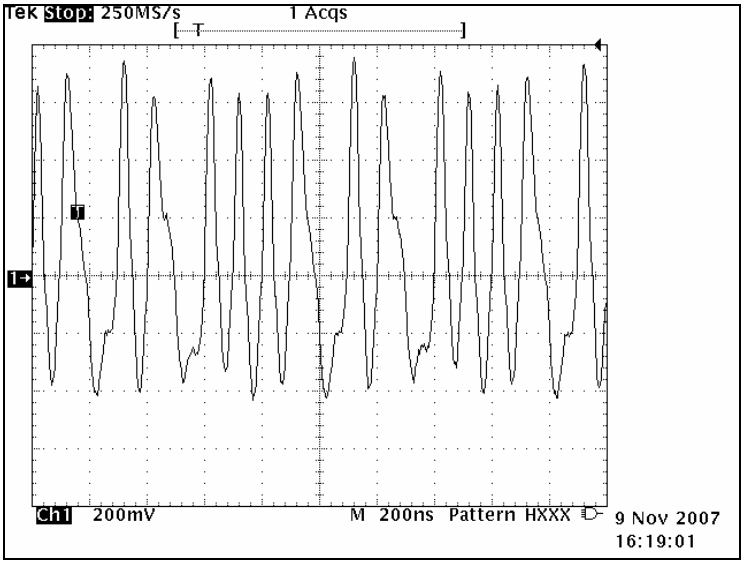


Figura 4.14 – Sinal da interface Ethernet

4.2.8 Ruído da fonte de alimentação

Na figura 4.15 é apresentado o sinal proveniente da fonte de alimentação, com sua tensão nominal de 5V. Pode-se reparar que a qualidade desta fonte não é muito boa, pois traz bastante ruído, contudo isso não causou mal funcionamento do circuito. Essa imagem foi obtida com o circuito ligado e em funcionamento.

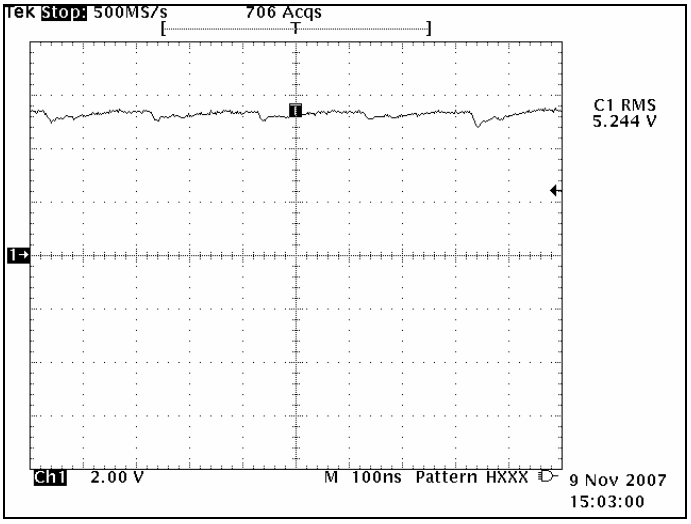


Figura 4.15 – Fonte de alimentação de 5V

Na figura 4.16 é apresentado o sinal da fonte de 3,3V, que possui menos ruídos se comparada com a fonte de 5V. Esta imagem também foi capturada com o circuito ligado e em funcionamento.

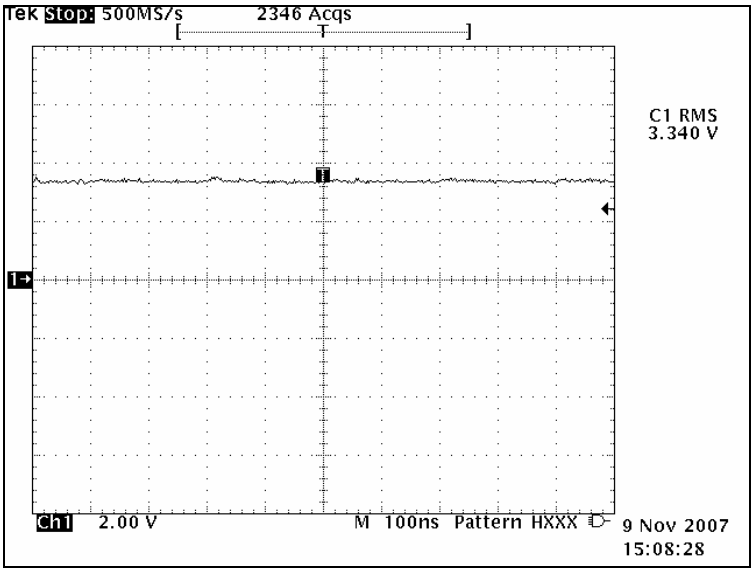


Figura 4.16 – Fonte de alimentação de 3,3V

4.3 Testes finais

Este tópico é destinado a apresentar os testes que foram realizados após o pleno funcionamento do circuito, e serviram para definir se ele funcionaria ou não em algumas condições específicas de rede.

Os testes foram realizados em três topologias diferentes:

Ponto a ponto

Estrela – utilizando-se um hub

Estrela – utilizando-se um roteador sem fio

Em todos os casos o endereço IP do circuito foi 192.168.0.10, definido arbitrariamente. Os acessos foram feitos por browsers, digitando-se o endereço IP informado anteriormente na barra de endereços.

4.3.1 Ponto a ponto

Neste teste foi utilizado um cabo cruzado, ou crossover, para interligar o circuito ao computador, como apresentado na figura 4.17. A interface Ethernet do computador recebeu o endereço IP 192.168.0.5. Com a interligação do computador ao circuito, a conexão de rede foi reconhecida automaticamente.

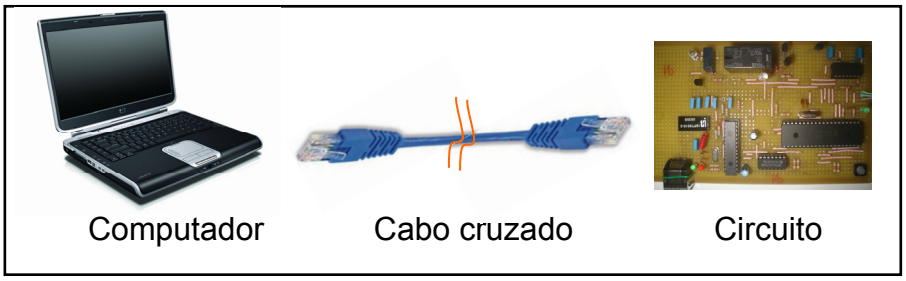


Figura 4.17 – Interligação ponto a ponto

Para confirmar o estabelecimento da conexão, foi executado o comando PING. Depois disso, a página Web armazenada no microcontrolador foi acessada através do browser Internet Explorer. O circuito aceitou comandos e informou a situação dos sensores.

O tempo médio de resposta foi obtido através da média aritmética do resultado de 10 comandos PING. Esse valor foi de 2,2ms. Outro ponto importante é que durante a execução dos comandos PING, a página Web não foi acessada, visto que isso causa um aumento de processamento e de tráfego de dados, retornando valores de tempo de resposta superiores ao medido.

4.3.2 Topologia estrela com hub

Para a interligação com o hub foi necessário utilizar dois cabos diretos para conectar tanto o computador quanto o circuito. Essa topologia é apresentada na figura 4.18. O hub utilizado foi o SD-800, do fabricante Planet.

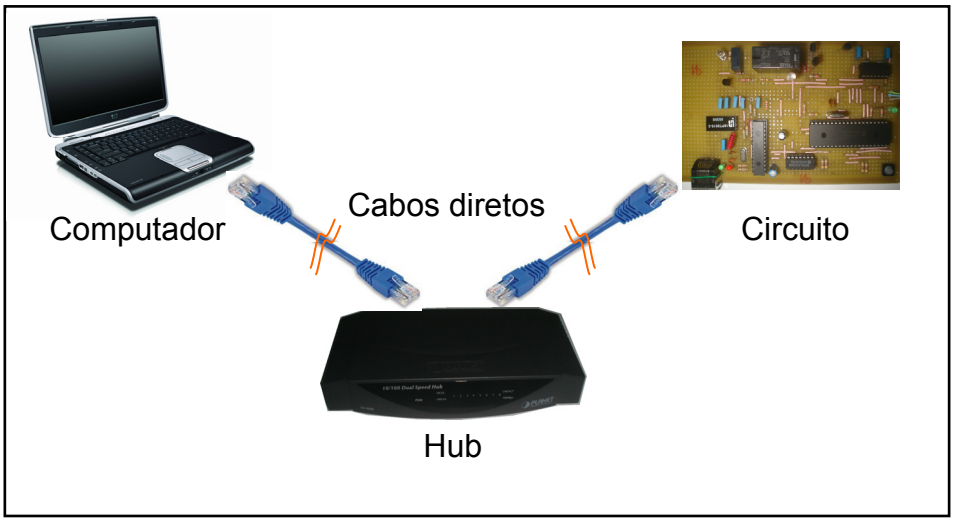


Figura 4.18 – Topologia estrela com hub

Ao conectar o circuito, o hub detectou sinal na sua porta. Após conectar o computador, o comando PING foi executado novamente. Em seguida, foi realizado o acessado através do browser FireFox, os comandos foram aceitos e a situação dos sensores foi informada normalmente.

Da mesma forma que o teste anterior, foi calculada a média do tempo de resposta após os dez comandos PING, resultando em 2,3ms.

4.3.3 Topologia estrela com roteador sem fio

Na figura 4.19 é apresentada a topologia de rede utilizada neste teste. Foi utilizado o roteador sem fio DI-524 do fabricante D-Link. O circuito foi interligado a uma das portas de padrão Eethernet do roteador através de um cabo direto.

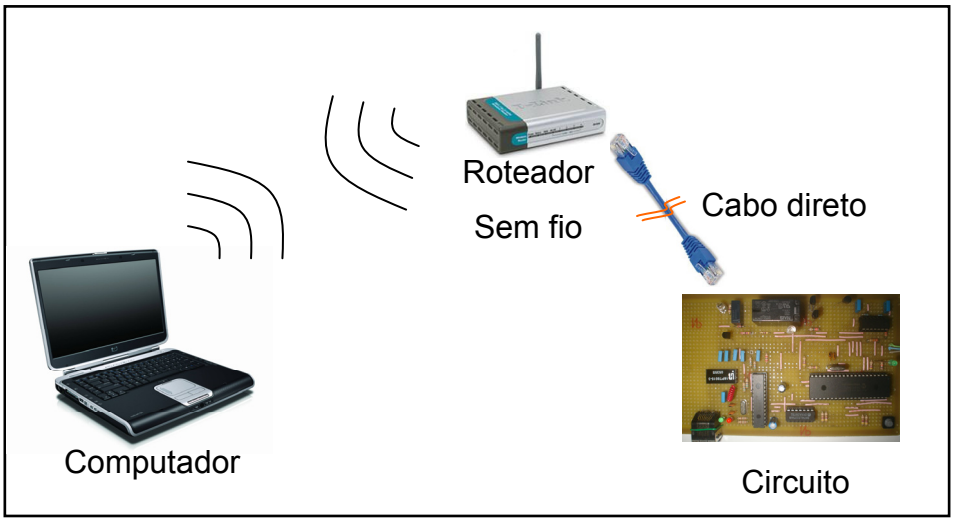


Figura 4.19 – Topologia estrela com roteador sem fio

Foi utilizada a interface de rede sem fio do computador para estabelecer a conexão deste com o roteador, esta interface recebeu o endereço IP 192.168.0.100, que foi atribuído por DHCP pelo roteador. Mais uma vez o comando PING foi executado e o circuito foi acessado.

Mais uma vez a média do tempo de resposta foi calculada após os dez comandos PING, resultando em 4,1ms. Isto sem que houvesse acesso à página Web.

4.4 Testes em condições reais de funcionamento

Estes testes têm por finalidade simular uma condição real de funcionamento, na qual o circuito é conectado a um modem satélite de modelo CDM-600 do fabricante Comtech. Este modem é utilizado para transmitir e receber sinais via satélite, em taxas de transmissão que variam entre 64Kbps a 20Mbps. Ele é utilizado em estações terrenas, ou seja, em pontos de concentração de tráfego para transmissão por satélite. As partes frontal e traseira do modem podem ser vistas na figura 4.20.

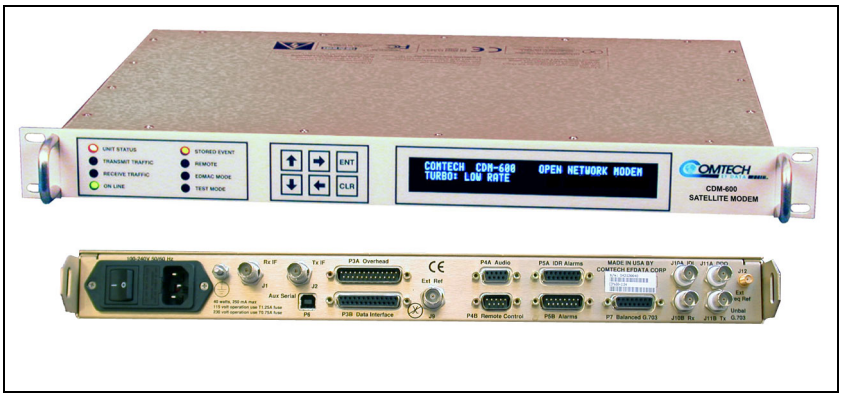


Figura 4.20 – Modem satélite CDM-600

4.4.1 Informações do modem

Este modem disponibiliza três tipos de informações de alarme através de contato seco, que é a forma de entrada de sinal no circuito desenvolvido neste projeto. Estas informações são referentes a falhas nos tráfego de transmissão e recepção, além de falhas na unidade. O modem também disponibiliza uma entrada do tipo contato seco, atuando-se nesta entrada é possível ativar ou desativar a transmissão.

Todos estes recursos são disponibilizados no conector P5B, localizado na parte traseira do modem, este é um conector do tipo DB-15 macho. É neste ponto que o circuito de monitoração e controle será conectado. As duas entradas do circuito de monitoração e controle serão conectadas aos pinos relativos aos alarmes de transmissão e de recepção. O acionador de maior capacidade será usado para ligar e desligar o modem, enquanto o acionador de menor capacidade será ligado aos pinos de controle da transmissão. Desta forma, o sensor 1 será ligado aos pinos 7 e 8, o sensor 2 aos pinos 13 e 14 e o acionador 2 aos pinos 1 e 9. Na tabela 4.1 pode ser vista a pinagem deste conector, segundo informações do próprio fabricante.

Pino	Função do sinal	Nome
8	Tráfego de RX (Desenergizado, em falha)	RX-NC
15	Tráfego de RX (Energizado, sem falha)	RX-NO
7	Tráfego de RX	RX-COM
14	Tráfego de TX (Desenergizado, em falha)	TX-NC
6	Tráfego de TX (Energizado, sem falha)	TX-NO
13	Tráfego de TX	TX-COM
5	Falha na unidade (Desenergizado, em falha)	UNIT-NC
12	Falha na unidade (Energizado, sem falha)	UNIT-NO
4	Falha na unidade	UNIT-COM
11	RX canal I (monitor de constelação)	RX-I
3	RX canal Q (monitor de constelação)	RX-Q
10	Não conectado	N/C
2	Tensão de AGC (nível de sinal de recepção)	AGC
9	Desativa a portadora por comando externo	EXT-OFF
1	Terra	GND

Tabela 4.1 – Pinagem do conector de alarmes do modem CDM-600

4.4.2 Testes

Foi simulado um enlace satélite com a utilização de dois modems CDM-600, onde o sinal de transmissão (FI – 70MHz) do primeiro foi ligado ao conector de recepção do segundo e vice-versa. As interfaces de dados de cada modem foram mantidas em loop, ou seja, sinal de recepção ligado ao conector de transmissão. Na figura 4.21 são ilustradas estas ligações. Além disso, foram feitas as devidas configurações para se estabelecer a conexão, contudo, estes detalhes não serão apresentados aqui.

A topologia completa do sistema montado para esse teste é apresentada na figura 4.22, onde podem ser vistas as ligações entre o computador, o circuito e os modems satélite.

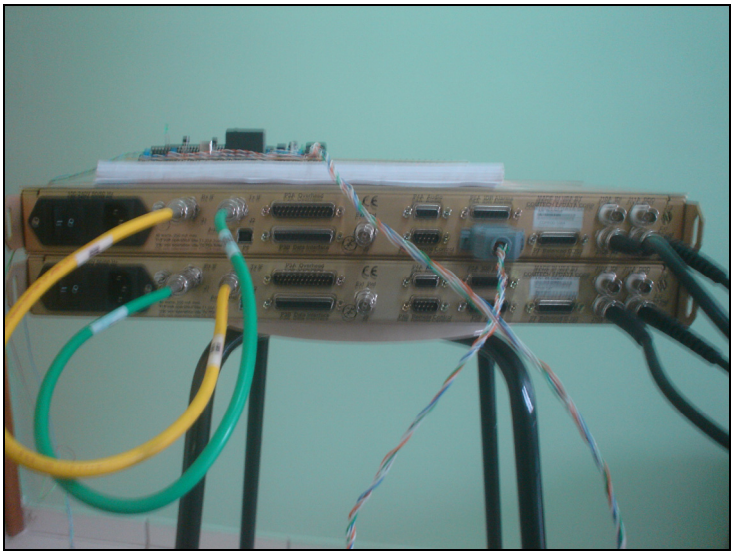


Figura 4.21 – Conexões entre os modems

Define-se então que o modem 1 (superior) representa o equipamento local monitorado, e que o modem 2 (inferior) representa o modem distante, ou seja, instalado no local de destino do link.

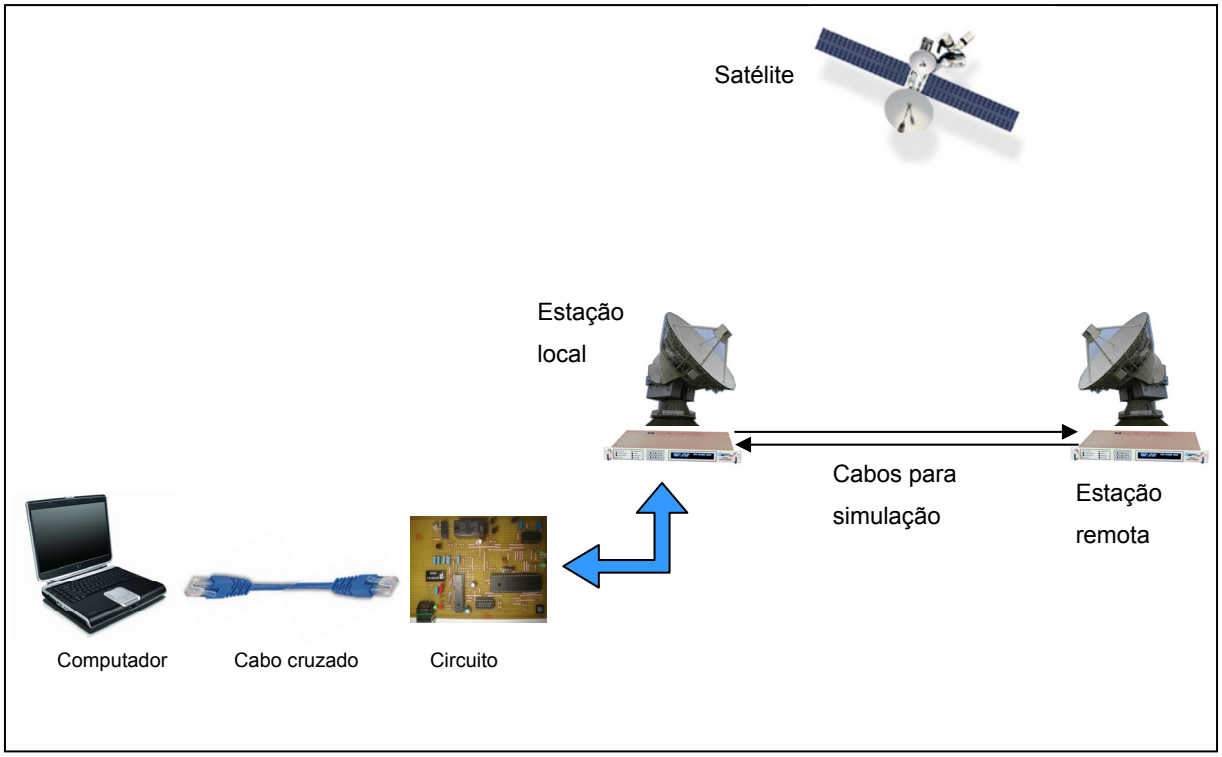


Figura 4.22 – Conexões entre os modems

O primeiro teste realizado foi o de ligar o modem local através do comando executado pelo botão RELÉ 1 da página Web. Ao clicar no botão, o relé 1 foi acionado, ligando o modem satélite. O esquema elétrico simplificado da ligação é apresentado na figura 4. 23.

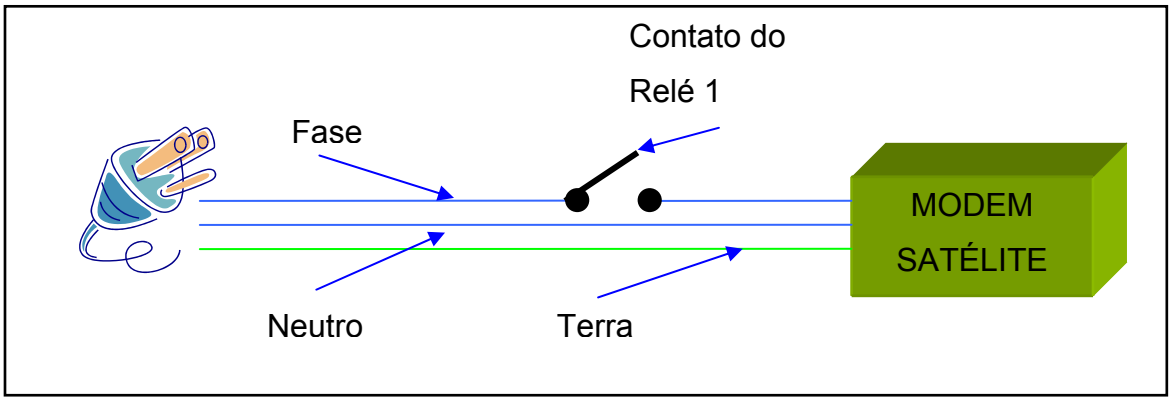


Figura 4.23 – Esquema simplificado

O segundo teste foi realizado com a simulação de uma interrupção do enlace no sentido de recepção. Isto foi feito com a desativação da transmissão do modem 2, o resultado é apresentado na figura 4.24, onde a informação do sensor 2 é alterada.

Figura 4.24 – Resultado do segundo teste

O terceiro teste foi realizado com a simulação de uma falha no sentido de transmissão, retirando-se o sinal de entrada na interface de dados (loop). Como resultado houve alteração na informação do sensor 1 na página Web, isto é visto na figura 4.25.

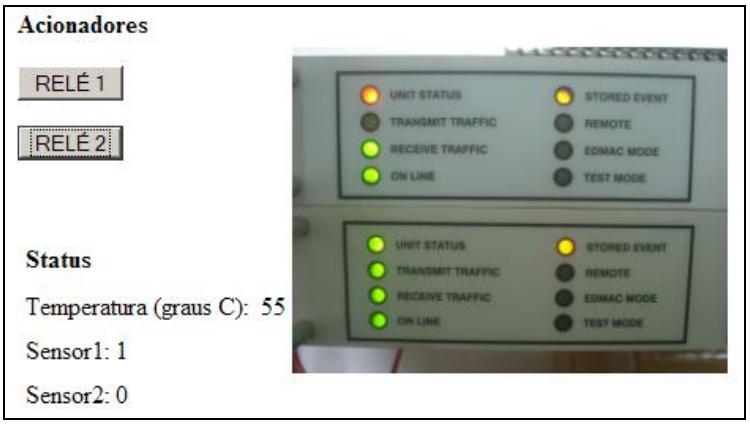


Figura 4.25 – Resultado do terceiro teste

O quarto teste foi feito no sentido de atuar no modem satélite, desativando sua portadora de transmissão. Para isso, o procedimento foi clicar no botão Relé 2 da página Web. Como resultado o relé 2 foi acionado e a portadora do modem 1 foi desativada, fazendo com que o modem 2 indicasse perda de sinal de recepção. O resultado é apresentado na figura 4.26.

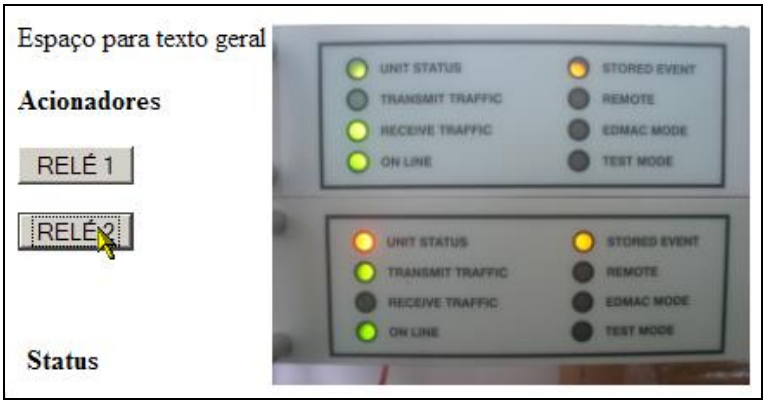


Figura 4.26 – Resultado do quarto teste

Como quinto, e último teste, o modem 1 foi desligado pressionando-se novamente o botão Relé 1 da página Web. O modem foi desligado.

Estes testes foram conclusivos e confirmam o pleno funcionamento do circuito.

Capítulo 5. Conclusão

No circuito desse projeto foram utilizados apenas dois sensores e dois acionadores, mas isto não significa que outros não possam ser associados. As limitações são a disponibilidade de memória e a adequação do hardware. Para facilitar esta ampliação podem ser criados módulos, sejam de sensores, sejam de acionadores. Dessa forma, além de se facilitar a ampliação, a manutenção também será facilitada, reduzindo a indisponibilidade do circuito em caso de defeito.

Em casos de instalações com uma grande quantidade de sensores e acionadores, também podem ser utilizados blocos de interligação, evitando a interligação diretamente ao circuito.

O padrão Ethernet foi escolhido por ser bastante difundido e usual. As redes Ethernet estão presentes em grandes empresas, nas suas redes com centenas de computadores, ou mesmo nas residências, nas ligações entre computador e modem ADSL, por exemplo. Dessa forma, a interface Ethernet, associada a um circuito, agrega muito valor.

Com a utilização da interface Ethernet, o acesso às informações disponibilizadas se torna fácil, sem a necessidade de se criar outros módulos para fazer a comunicação. Este acesso também poderá ser feito através da Internet, possibilitando o gerenciamento a longas distâncias. Pode-se criar, por exemplo, uma rede composta de vários circuitos, espalhados em várias regiões do país e um ponto central de controle.

Uma outra questão importante que foi percebida durante o desenvolvimento deste projeto, é a sua utilização em outras áreas de serviços, já que podem ser feitas alterações no hardware e no software, de forma a adaptar o circuito a condições específicas de trabalho. Com isso, sua área de aplicação se torna bastante ampla, podendo atuar, por exemplo, nas áreas de segurança empresarial ou residencial, automação, em linhas de produção ou, até mesmo, na agricultura mecanizada.

Vale lembrar também, que o circuito dispensa a utilização de um computador, tanto na gerência dos sensores e acionadores quanto na transmissão das informações. Isto simplifica o uso, minimiza os custos de hardware e software

adicionais e reduz o espaço necessário para a instalação, bem como o consumo de energia.

5.1 Sugestões para projetos futuros

Ficam registradas aqui, algumas sugestões para projetos futuros, tais como:

- Captura e transmissão de áudio a partir desta base;
- Captura e transmissão de imagens a partir desta base;
- A troca da interface Ethernet por uma interface sem fio.

Referências Bibliográficas

COMER, Douglas E. Interligação em rede com TCP/IP.; trad. ARX Publicações – 3.ed – Rio de Janeiro: Elsevier, 1998.

COMER, Douglas E. Redes de Computadores e Internet; trad. Marinho Barcelos – 2.ed – Porto Alegre; Bookman, 2001

MICROCHIP. 2007. Disponível em: <http://www.microchip.com>. Acessado em: 10 de dezembro de 2006.

MICROCHIP. ENC28J60 - Data Sheet. 2006. Disponível em: <http://www.microchip.com >. Acessado em: 25 de setembro de 2006.

MICROCHIP. PIC18F2525/2620/4525/4620 - Data Sheet. 2007. Disponível em: <http://www.microchip.com >. Acessado em: 20 de fevereiro de 2007.

MICROSOFT. Glossário de rede doméstica. 2006. Disponível em: <http://www.microsoft.com/brasil/windowsxp/using/networking/getstarted/glossary.mspx>. Acessado em: 14 de setembro de 2006.

MUSEU DO COMPUTADOR. Enciclopédia. 2004. Disponível em: <http:// www.museudocomputador.com.br/enciredе.php>: Acessado em: 07 de setembro de 2006.

RAJBHARTI, Nilesh. The Microchip TCP/IP Stack. 2002.Disponível em: <http://www.microchip.com>. Acessado em: 7 de março de 2007.

SILVEIRA, Jorge Luis da. Comunicação de dados e sistemas de teleprocessamento. São Paulo: Makro, McGraw-Hill,1991.

SIQUEIRA, Leonardo Francisco. Ethernet óptica. 2004. Disponível em: <http:// www.gta.ufrj.br/grad/04_1/ethernet-opt/parte3.html>. Acessado em: 07 de setembro de 2006.

SOARES, Luiz Fernando G. Redes de computadores: das LANS, MANs e WANs às redes ATM. 2ª ed. Rio de Janeiro: Campus, 1995.

SOUZA, David José de; LAVÍNIA, Nicolas Cézar Conectando o PIC 16F877A: Recursos Avançados 1.a Ed. São Paulo: Érica, 2003

TANEMBAUM, Andrew S. Redes de computadores. Trad. Vandenberg D de Souza. 4ª ed. Rio de Janeiro: Elseveir, 2003.

TAROUCO, Liane Margarida Rockenbach. Redes de computadores locais e de longa distância. São Paulo: Mc Graw-Hill,1986.

TORRES, Gabriel. Redes Locais – Placas e Cabos. 1998. Disponível em: <http://www.clubedohardware.com.br>. Acessado em: 23 de janeiro de 2007

Apêndice A – Código fonte da página Web

Código fonte da página Web armazenada no microcontrolador.

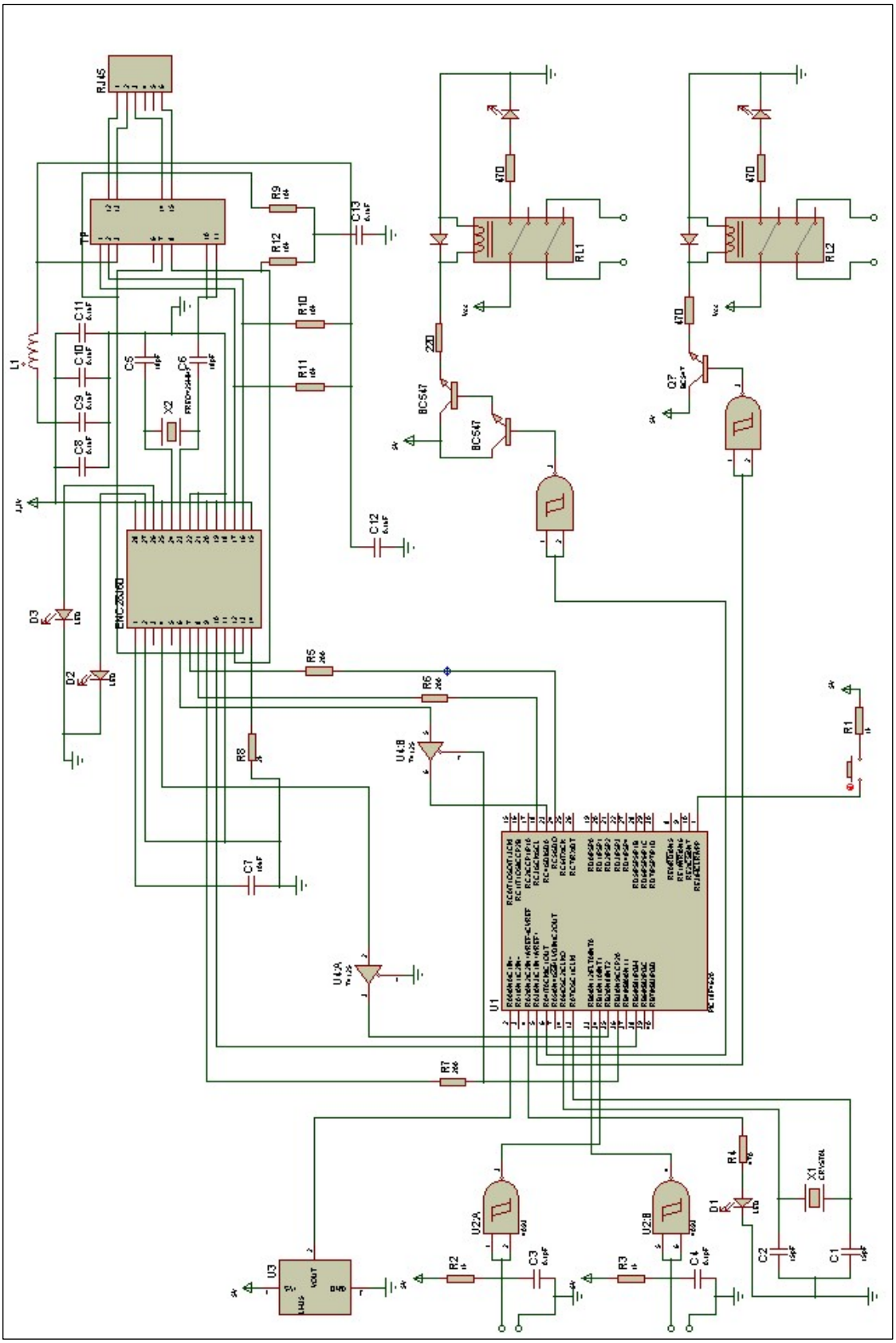
```
<html>
	<head>
		<title>Monitoração e Controle através de Rede Ethernet</title>

		<script language="JavaScript">
			var	xmlHttp;
			var ObjArray = new Array;

			function GetXmlHttpObject(handler)
			{
			var objXmlHttp = null;

			if(navigator.userAgent.indexOf("MSIE")>=0)
			{
			var ClassName = "Msxml2.XMLHTTP";
			if(navigator.appVersion.indexOf("MSIE 5.5")>=0)
			{
			ClassName = "Microsoft.XMLHTTP";
			}
			try
			{
			objXmlHttp = new ActiveXObject(ClassName);
			objXmlHttp.onreadystatechange = handler;
			return objXmlHttp;
			}
			catch(e)
			{
			alert("Error: ActiveX scripting may be disabled.");
			return;
			}
			}
			else
			{
			try
			{
			objXmlHttp = new XMLHttpRequest();
			objXmlHttp.onload = handler;
			objXmlHttp.onerror = handler;
			return objXmlHttp;
			}
			catch(e)
			{
			alert("Error: Browser may not be supported or browser security restrictions are
too high.  XMLHttpRequest() support is required.");
			}
			}
			}

			function StateChanged()
			{
			if(xmlHttp.readyState == 4 || xmlHttp.readyState == "complete")
```

```
                        {

        document.getElementById("txtAutoUpdateStatus").innerHTML=xmlHttp.responseText;
                        xmlHttp = null;
                        UpdateStatus();
                        }
                        }


                        function UpdateStatus()
                        {
                        xmlHttp = GetXmlHttpObject(StateChanged);
                        xmlHttp.open("GET", "Status.cgi" , true);
                        xmlHttp.send(null);
                        }

                        function GetServerFile(FileName, AssignTo)
                        {
                        var NiftyObj = new Object();
                        NiftyObj.XMLDevice = new GetXmlHttpObject(StateChanged2);
                        NiftyObj.XMLDevice.open("GET", FileName, true);
                        NiftyObj.XMLDevice.send(null);
                        NiftyObj.Text = AssignTo;
                        ObjArray.push(NiftyObj);
                        }

                        function StateChanged2()
                        {
                        for(i in ObjArray)
                        {
                        if(ObjArray[i].XMLDevice.readyState == 4 || ObjArray[i].XMLDevice.readyState
== "complete")
                        {
                        if(ObjArray[i].Text != "")
                        {

        document.getElementById(ObjArray[i].Text).innerHTML=ObjArray[i].XMLDevice.responseText
;
                        }
                        if(ObjArray[i].Text == "txtAutoUpdateStatus")
                        {
                        GetServerFile("Status.cgi", "txtAutoUpdateStatus");
                        }
                        delete ObjArray[i].XMLDevice;
                        delete ObjArray[i];
                        }
                        }
                        }



                </script>


        </head>
        <body bgcolor="white" onload="UpdateStatus(); GetServerFile('Version.cgi','txtStackVersion');
GetServerFile('BuildDate.cgi','txtBuildDate');">

                <table border="40" width="100%" bordercolordark=<table border="40" width="100%"
bordercolordark="#238E23" bordercolorlight="#99FF99">
```

```
                    <tr>
                        <td>
                            <p >
                            <img src="logo_ceub.gif"></img>
                            </p>
                        </td>
                        <td width="100%">


                                    <p align="center" >

                                    <font size="7" face="NewCenturySchlbk"

color="#238E23" >                                                   <b>
                                        Monitoração e Controle através

Rede Ethernet</hr>

                                            <br>
                                        </b></y>
                                    </font>

                                    </p>
                                    <p align="center">
                                    <font      size="4"      face="NewCenturySchlbk"
color="#238E23">

                                        <b>
                                            Daniel de Oliveira

Santos
                                        </b>
                                    </font>

                                    </p>
                                    <p align="center">
                                    <font      size="3"      face="NewCenturySchlbk"
color="#238E23">

                                        <b>
                                            Professora orientadora:

Maria Marony Sousa Farias Nascimento
                                                    </b>
                                    </font>
                            </p>
                        </td>
                    </tr>
                </table>


                <br>

                    <table border="0" cellpadding="0" cellspacing="0" width="100%">
                        <tr>
                            <td valign="top" width="49%">
```

```
                                        <br>
                                            <p>
                                                Espaço para texto
geral
                                                </p></p>
                                                </td>
                        </tr>
                    </table>

                                        <td>
                                            <b>Acionadores</b>
                                        </td>

                    <table cellpadding="3">
                        <form>

                                <tr>
                                <p>

                                        <input type="button"value="RELÉ 1"
onclick="GetServerFile('0?1=LED2','')">
                                        </input>
                                </p>
                                </tr>
                                <tr>
                                <p>
                                        <input type="button" value="RELÉ 2"
onclick="GetServerFile('0?0=LED1','')">
                                        </input>
                                </p>
                                </tr>
                                </td>
                        </tr>
                    </form>
                    <tr>
                        <td> </td>
                    </tr>
                    <tr>
                        <td>
                            <b>Status</b>
                        </td>
                    </tr>
                </table>

                <span id="txtAutoUpdateStatus">Carregando...</span>


        </body>
</html>
```

Apêndice B – Diagrama completo do circuito



Anexo A – Código fonte do programa principal

Programa principal armazenado na memória de programa do microcontrolador

```
/********************************************************************
 *
 *      Monitoração e Conrtole através de rede Ethernet
 *
 ********************************************************************
 *
 *                  Centro Unicersitário de Brasília - UniCeub
 *                  Aluno: Daniel de Oliveira Santos
 *                  Orientadora: Prof. Maria Marony
 *
 *
 *
 *                  Este é o programa principal do projeto, ele faz toda a inicialização
 *         e configuração do microcontrolador. Este programa também faz as chamadas
 *         das rotinas de comunicação através da rede Ethernet.Ele é baseado na
 *         pilha TCP/IP da Microchip
 *
 *
 ********************************************************************/

// Cabeçalhos necessários
#include <string.h>
#include "..\Include\Compiler.h"
#include "..\Include\StackTsk.h"
#include "..\Include\Tick.h"
#include "..\Include\MAC.h"
#include "..\Include\Helpers.h"
#include "..\Include\Delay.h"
#include "..\Include\UART.h"
#include "..\Include\MPFS.h"
#include "..\Include\LCDBlocking.h"
#include "..\Include\GenericTCPClient.h"
#include "..\Include\HTTP.h"



// Definição das variáveis de endereçamento IP
APP_CONFIG AppConfig =
        {
                {MY_DEFAULT_IP_ADDR_BYTE1,                  MY_DEFAULT_IP_ADDR_BYTE2,
MY_DEFAULT_IP_ADDR_BYTE3, MY_DEFAULT_IP_ADDR_BYTE4},
                {MY_DEFAULT_MAC_BYTE1,                          MY_DEFAULT_MAC_BYTE2,
MY_DEFAULT_MAC_BYTE3,       MY_DEFAULT_MAC_BYTE4,       MY_DEFAULT_MAC_BYTE5,
MY_DEFAULT_MAC_BYTE6},
                {MY_DEFAULT_MASK_BYTE1,                         MY_DEFAULT_MASK_BYTE2,
MY_DEFAULT_MASK_BYTE3, MY_DEFAULT_MASK_BYTE4},
                {MY_DEFAULT_GATE_BYTE1,                         MY_DEFAULT_GATE_BYTE2,
MY_DEFAULT_GATE_BYTE3, MY_DEFAULT_GATE_BYTE4},
                {MY_DEFAULT_DNS_BYTE1,                          MY_DEFAULT_DNS_BYTE2,
MY_DEFAULT_DNS_BYTE3, MY_DEFAULT_DNS_BYTE4},
```

```
                {0b00000001},
        };

BYTE myDHCPBindCount = 0;
#if defined(STACK_USE_DHCP)
   extern BYTE DHCPBindCount;
#else
        #define DHCPBindCount          (0xFF)
#endif


//Configuração do microcontrolador
                #pragma config OSC=HS, WDT=OFF, MCLRE=ON, PBADEN=OFF, LVP=OFF,
XINST=OFF

static void InitAppConfig(void);
static void InitializeBoard(void);
static void ProcessIO(void);

BOOL StringToIPAddress(char *str, IP_ADDR *buffer);
static void DisplayIPValue(IP_ADDR *IPVal);
static void SetConfig(void);
static void FormatNetBIOSName(BYTE Name[16]);


#define SaveAppConfig()



// Correção recomendada pela Microchip para correção de bug no processo de interrupção
#if defined(HI_TECH_C)
void interrupt HighISR(void)
#else
#pragma interruptlow HighISR
void HighISR(void)
#endif
{
#ifdef __18CXX
   TickUpdate();
#endif

#if defined(STACK_USE_SLIP)
   MACISR();
#endif
}

#if defined(__18CXX) && !defined(HI_TECH_C)
#pragma code highVector=0x08
void HighVector (void)
{
   _asm goto HighISR _endasm
}
#pragma code
#endif

ROM char NewIP[] = "New IP Address: ";
ROM char CRLF[] = "\r\n";

void main(void)
{
```

```
    static TICK t = 0;

    // Inicialização do circuito com suas características específicas
    InitializeBoard();


    // Inicialização dos componentes da pilha TCP/IP
    TickInit();

    MPFSInit();

            memcpypgm2ram(AppConfig.NetBIOSName, (ROM void*)MY_DEFAULT_HOST_NAME, 16);
            FormatNetBIOSName(AppConfig.NetBIOSName);

    InitAppConfig();

     StackInit();

    HTTPInit();

    // Loop infinito
            while(1)
            {
        // Pisca o LED indicando atividade
        if ( TickGetDiff(TickGet(), t) >= TICK_SECOND/2 )
        {
            t = TickGet();
            LED0_IO ^= 1;
        }

        // Verificação de pacotes recebidos
        StackTask();

        // Executa as funções do servidor HTTP
        HTTPServer();

        // Ações executadas pelo circuito
        ProcessIO();
    }
}


static char AN0String[8];

static void ProcessIO(void)
{
    // Configura porta AN0 como entrada analógica
    ADCON0bits.GO = 1;

    // Aguarda a conversão A/D
    while(ADCON0bits.GO);

    // Converte o valor dos 10 bits em ASCII
    itoa(*((WORD*)(&ADRESL)), AN0String);
}

// Códigos de comando CGI
#define CGI_CMD_DIGOUT      (0)
#define CGI_CMD_LCDOUT      (1)
#define CGI_CMD_RECONFIG    (2)
```

```
// Código das variáveis CGI
#define VAR_ANAIN_AN0       (0x02)
#define VAR_DIGIN0          (0x04)  // Sensor 1
#define VAR_DIGIN3          (0x0F)  // Sensor 2
#define RELE1               (0x0)
#define RELE2               (0x1)


// Tratamento dos comandos recebidos

#if defined(STACK_USE_HTTP_SERVER)

ROM char COMMANDS_OK_PAGE[] = "INDEX.CGI";
ROM char CONFIG_UPDATE_PAGE[] = "CONFIG.CGI";
ROM char CMD_UNKNOWN_PAGE[] = "INDEX.CGI";

#define COMMANDS_OK_PAGE_LEN        (sizeof(COMMANDS_OK_PAGE))
#define CONFIG_UPDATE_PAGE_LEN  (sizeof(CONFIG_UPDATE_PAGE))
#define CMD_UNKNOWN_PAGE_LEN    (sizeof(CMD_UNKNOWN_PAGE))

void HTTPExecCmd(BYTE** argv, BYTE argc)
{
   BYTE command;
   BYTE var;
#ifdef ENABLE_REMOTE_CONFIG
   BYTE CurrentArg;
   WORD_VAL TmpWord;
#endif

   command = argv[0][0] - '0';

    switch(command)
   {
   case CGI_CMD_DIGOUT:

      var = argv[1][0] - '0';

      switch(var)
      {
      case RELE1:
         LED1_IO ^= 1;
         break;

      case RELE2:
         LED2_IO ^= 1;
         break;
       }

       memcpypgm2ram((void*)argv[0],          (ROM          void*)COMMANDS_OK_PAGE,
COMMANDS_OK_PAGE_LEN);
       break;

   default:
                memcpypgm2ram((void*)argv[0],      (ROM      void*)COMMANDS_OK_PAGE,
COMMANDS_OK_PAGE_LEN);
      break;
   }

}
```

```
#endif

#if defined(STACK_USE_HTTP_SERVER)
WORD HTTPGetVar(BYTE var, WORD ref, BYTE* val)
{
        static BYTE VarString[20];

        // Identificação da variável
   switch(var)
   {

   case VAR_ANAIN_AN0:
      *val = AN0String[(BYTE)ref];
      if(AN0String[(BYTE)ref] == '\0')
         return HTTP_END_OF_VAR;
                else if(AN0String[(BYTE)++ref] == '\0' )
         return HTTP_END_OF_VAR;
      return ref;

   case VAR_DIGIN0:
      *val = BUTTON0_IO ? '1':'0';
      break;

   case VAR_DIGIN3:
      *val = BUTTON3_IO ? '1':'0';
      break;

   }

   return HTTP_END_OF_VAR;
}
#endif

//Adequação às características do circuito
static void InitializeBoard(void)
{

        LED1_IO = 1;
        LED3_IO = 1;

        // Habilita oscilador 4x
   OSCTUNE = 0x40;

        // Configura as características analógicas do PORTA

        ADCON0 = 0b00000001;
        ADCON1 = 0b00001110;
        TRISA  = 0x01;
        ADCON2 = 0b10111110;


   // Habilita resistores de pull-up no PORTB
   INTCON2bits.RBPU = 0;

        // Habilita interrupções
   T0CON = 0;
   INTCONbits.GIEH = 1;
   INTCONbits.GIEL = 1;

}
```

```
static void InitAppConfig(void)
{
   AppConfig.Flags.bIsDHCPEnabled = FALSE;
}

static void FormatNetBIOSName(BYTE Name[16])
{
	BYTE i;

	Name[15] = '\0';
	strupr(Name);
	i = 0;
	while(i < 15)
	{
		if(Name[i] == '\0')
		{
			while(i < 15)
			{
				Name[i++] = ' ';
			}
			break;
		}
		i++;
	}
}
```