UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Juarez Bento da Silva

Monitoramento, aquisição e controle de sinais elétricos, via Web, utilizando microcontroladores.

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Professor orientador: João Bosco da Mota Alves

Florianópolis, fevereiro – 2002

Monitoramento, aquisição e controle de sinais elétricos, via Web, utilizando microcontroladores.

Juarez Bento da Silva

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração (insira o nome área de concentração) e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Fernando Alvaro Ostuni Gauthier

Banca Examinadora

_____

João Bosco da Mota Alves

_____

Luiz Fernando Jacintho Maia

_____

José Manuel Martins Ferreira

Multa espendio siunt quae signibus ardua videntur.
(A persistência supera o que os fracos consideram impossível.)

Tácito

Agradecimentos

A minha esposa Isabel, meus filhos Juarez, Karmel, Isabela e Raul, pelo apoio e tempo que lhes foi “roubado por culpa” desta dissertação.

A meu orientador, João Bosco da Mota Alves, pelo tempo dedicado e pelas contribuições realizadas neste trabalho.

A Muriel Berhardt e Suenoni Paladini, por sua grande contribuição na realização deste trabalho, pois o seu auxílio na documentação e implementação das rotinas de testes foram fundamentais para a realização deste projeto.

A e Alex Moretti, Maurício de Paula e Saulo Zambiasi, meus colegas no RExLab pelas contribuições referentes ao desenvolvimento do software de monitoramento e no auxílio à confecção das placas.

A José de Oliveira Ramos, superintendente da Unisul – Campus de Araranguá pelo incentivo e apoio proporcionado.

Sumário

Lista de Figuras

Lista de Quadros

Lista de Tabelas

Resumo

Uma grande quantidade de sistemas e equipamentos “inteligentes” tem presença cada vez maior em nossa vida diária: telefones celulares, impressoras laser, fornos microondas, terminais de atendimento bancário entre outros. Uma característica comum entre esses equipamentos é possuírem sistemas embutidos, que recebem esse nome por serem sistemas eletrônico-computacionais embutidos em um sistema ou processo maior.

Paralelo ao crescimento do número de dispositivos microcontrolados, avança a Internet, que representa atualmente, um meio de comunicação o qual as pessoas estão familiarizadas e pode até ser vista como rotina, em nossos dias, atualmente “navegar” na Web pode ser considerada uma tarefa rotineira e manusear um “browser Web” não se constitui em novidade para estas pessoas.

As facilidades que a Internet oferece e os recentes desenvolvimentos em sistemas embutidos permitirão que num futuro, não muito distante, possam ser conectados e acessar globalmente dispositivos elétricos domésticos, equipamentos industriais, equipamentos médicos e muitos outros aqui não especificados.

O presente trabalho está focado em um sistema embutido baseado em microcontrolador implementado em uma plataforma prática e adequada para a interconexão de dispositivos elétricos em redes TCP/IP, incluindo a Internet. Neste documento são descritos o projeto desta plataforma e a implementação de um protótipo funcional simples, também são propostos alguns delineamentos para futuros desenvolvimentos nesta área.

Abstract

1.0. Introdução

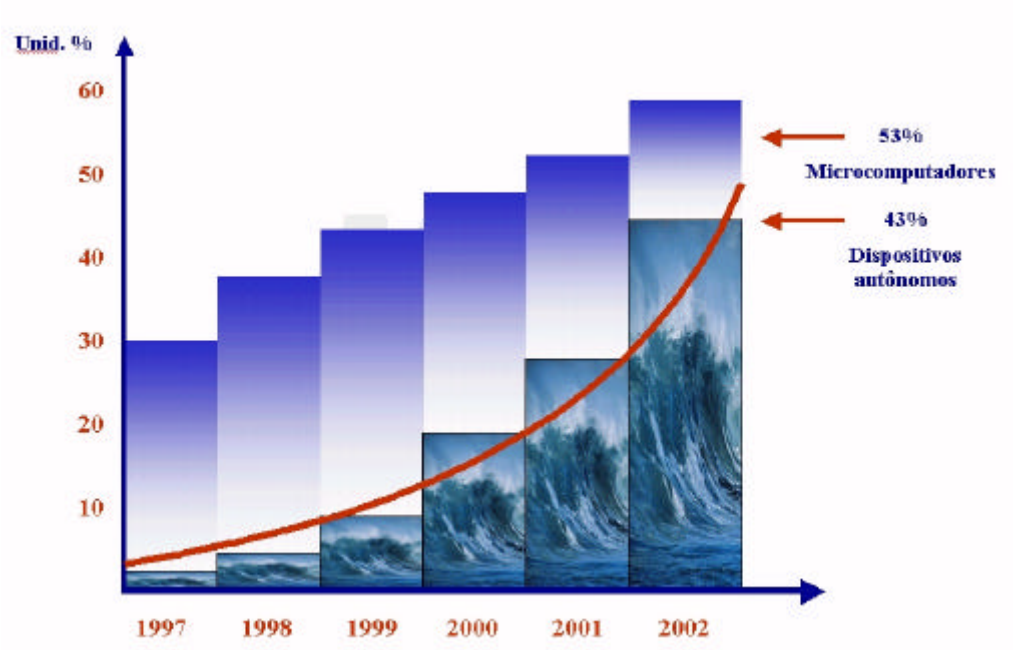
Neste capítulo se estabelece inicialmente o contexto em que se desenvolve o presente trabalho, assim como os motivos que deram lugar ao mesmo. Na continuação é apresentada a proposta de trabalho onde são citados os objetivos perseguidos e as contribuições obtidas. Por último, se realiza um breve resumo do conteúdo de cada um dos capítulos, que trata de dar uma visão global do trabalho realizado.

A diferença entre as redes e os dispositivos conectados a elas está sendo reduzida lentamente, na figura 1, é mostrado estudo feito pela IDC (International Data Corporation) no ano de 2001, onde se percebe um número crescente de dispositivos eletrônicos conectados en rede independentemente de um computador ou servidor, e busca-se contar com características tais como eficiência, economia, confiabilidade, segurança e compatibilidade com os padrões internacionais de comunicação e topologias de redes atuais, tornado-os compactos e funcionais.

Pode ser citado como fator importante para consolidação desta expansão à aplicação de esforços e recursos em projetos na área de domótica, a cada dia surgem mais informações sobre casas inteligentes apresentando múltiplas versões e matizes. De uma maneira geral, um sistema domótico deverá dispor de uma rede de comunicação e diálogo que permita a interconexão de uma série de equipamentos a fim de obter informações sobre o ambiente doméstico e, baseando-se neste, realizar algumas ações sobre este ambiente.

Atentos às diversas situações de nosso cotidiano muitos fabricantes tem direcionado esforços para implementar sistemas embutidos com suporte a rede, área que está em franca expansão, atualmente muitos deles direcionam suas tecnologias para plataformas de sistemas embutidos, inclusive alguns deles com suporte para interface de rede.

FIGURA 1 – DISPOSITIVOS EM REDE, A PRÓXIMA ONDA



Fonte: IDC International Data Corporation - 2001

Coerente com esta linha de trabalho é apresentado este desenvolvimento teórico-prático para conectar dispositivos elétricos a uma rede padrão Ethernet, baseado em um microcontrolador de baixo custo utilizando para a comunicação de dados os protocolos TCP/IP e que permita adquirir, controlar e monitorar remotamente estes dispositivos de maneira segura, eficiente e econômica, mediante o uso de um navegador padrão (browser) para Internet.

1.1. Motivação

A Internet representa, atualmente, um meio de comunicação o qual as pessoas estão familiarizadas, logo, explorar a sua infra-estrutura para conectividade apresenta-se como uma grande oportunidade de desenvolvimento. Pretendendo explorar as potencialidades da Internet e buscando um estudo qualificado na área de microcontroladores, apresenta-se como uma potencial alternativa para desenvolvimento um servidor Web, o baixo custo de desenvolvimento de dispositivos baseados em microcontroladores, componentes eletrônicos bem como ferramentas de desenvolvimento e documentação permitiram pensar em um servidor Web embutido, de baixo custo, rodando uma pilha TCP/IP e hospedando páginas Web.

O real valor de um servidor de Web embutido não é a somente a conexão Web, mas as aplicações embutidas neles.Um pequeno servidor poderá fazer uso da Internet para auxiliar no monitoramento e controle equipamentos de comunicações, sistemas para aquisição de dados, sistemas de inspeção, máquinas industriais, sistemas de controle de processos, equipamentos médicos, controladores internos de veículos, máquinas de escritório e muitos outros dispositivos.

A possibilidade de conexão a Internet embutida em equipamentos convencionais traz vantagens no monitoramento e controle com custos relativamente baixos, utilizando uma interface familiar para os usuários e com um alto grau de padronização. Um servidor Web embutido poderá conter páginas web dinâmicas podem controlar as comunicações com outros dispositivos utilizando os protocolos TCP/IP. Esta metodologia provê uma padronização do mecanismo de comunicação entre os dispositivos através da utilziação de páginas web simples de fácil instalação e manutenção, e que dispõem de uma conhecida Graphical Universal Interface (GUI) através de um web browser. O browser poderá descarregar processamento para um controlador embutido, ou pode habilitar o servidor web embutido para controle de dados via JavaScript e Java.

1.2. Proposta

Este trabalho tem por objetivo utilizar a Internet como uma rede global para monitoramento remoto de dispositivos. A expansão dos recursos disponíveis para a Internet e a ampla documentação sobre suas características técnicas e particularmente dos protocolos usados facilita o desenvolvimento das aplicações, reduzindo custos e o tempo no desenvolvimento de sistemas que venham a utiliza-la.

As estratégias contidas nesta dissertação procuram mostrar a viabilidade de desenvolvimento e implementação de um sistema embutido, autônomo, de pequeno tamanho, flexível, escalável e baseado em microcontrolador que pode ser usado para controlar outros dispositivos elétricos, domésticos, por exemplo, lâmpadas, termômetros, câmeras, cafeteiras, equipamentos industriais como fornos, câmaras frias, equipamentos médicos, entre outros, denominado micro servidor web (MSW).

A implementação do micro servidor web em aplicações na área médica é uma porta que se abre no monitoramento remoto, via Internet, o MSW é um servidor WEB que pode ser conectado via interface serial RS-232 com qualquer equipamento que disponha também deste tipo de interface. Os dispositivos remotos podem ser controlados e monitorados por um microcomputador convencional, desde que esteja conectado na mesma rede ethernet onde estão também os MSW. O sistema aqui apresentado contém um dispositivo de controle, alguns dispositivos remotos, software para monitoração e a rede.

Este documento pretende enfatizar os aspectos da implementação do hardware, discutir as implementações particularmente testadas e a utilização futura da arquitetura proposta. A arquitetura do sistema foi projetada para ser flexível, para que possa ser expandida e modificada no futuro. Esta dissertação também cumpre com o propósito de oferecer aos professores, pesquisadores e estudantes mecanismos para compreender a base de funcionamento de microservidores Web, ou seja; uma forma de compreender o princípio de funcionamento mostrando as diferentes etapas de desenvolvimento de um microservidor Web autônomo, e desta forma adquirir a capacidade para adaptá-lo ou de projetar um novo de acordo com as necessidades que se queira suprir.

1.3. Organização do Texto

Esta dissertação está formalizada em um texto composto por sete capítulos como descritos a seguir.

Capítulo 1: Introdução

Introduz e discute os principais aspectos que levaram ao desenvolvimento e formalização deste trabalho com uma proposta plenamente utilizável no que se refere ao monitoramento e controle de dispositivos remotos, via web. O enfoque do trabalho, as motivações, os objetivos, e a estrutura da dissertação fazem parte desta introdução.

Capítulo 2: Analisando as Soluções Existentes

Neste capítulo é feito um breve estudo das tecnologias disponíveis atualmente e a relevâncias destas neste trabalho. Estas tecnologias estão relacionadas com os avanços

dos sistemas embutidos para Internet. São apresentadas brevemente algumas alternativas comercializadas.

Capítulo 3 – Definindo as características técnicas do MSW

Este capítulo desenvolve a implementação do microservidor WEB apresentando as suas básicas. Serão apresentados as características e o modelo implementado. Também será feita uma revisão sobre aspectos técnicos construtivos, abordagem teórica sobre os componentes eletrônicos utilizados na implementação e mostrados os recursos disponíveis para o projeto e implementação do MSW. Pretende-se efetuar um breve estudo de algumas soluções disponíveis no que se refere a firmware e sistemas operacionais disponíveis atualmente, para sistemas embutidos baseados na arquitetura AVR Atmel e a relevância destes neste trabalho.

Capítulo 4: Implementação do Hardware

Este capítulo apresenta uma estrutura formal para o MSW a partir de uma linha de pesquisa a ser empregada na análise e síntese para a solução dos problemas discutidos nos capítulos anteriores. Nele é concretizada a proposta de desenvolvimento de um novo micro servidor web que possibilite a sua utilização com base nas características introduzidas nos Capítulos 2 e 3 dentro das especificações técnicas propostas no Capítulo 3.

Capítulo 5 – Organização do Software

Este capítulo tem por objetivo apresentar o software básico (firmware) definido no capítulo 3, apresentando detalhes de funcionamento no âmbito deste projeto e também discorre sobre a organização e característica do software das aplicações.

Capítulo 6: Aplicações Práticas

Este capítulo apresenta os resultados computacionais da aplicação de dois modelos propostos. A primeira aplicação trata da utilização de uma câmera com conexão serial conectada ao MSW e a segunda sobre controle de temperaturas. As implementações dos modelos são analisadas separadamente.

Capítulo 7: Conclusões e Perspectivas Futuras

As diversas implicações resultantes da proposição de monitoramento remotas via web como umas potenciais possibilidades são discutidas neste capítulo. Como complemento, uma discussão sobre as tendências e a proposta de possíveis extensões deste trabalho e novas direções a serem tomadas fazem parte deste capítulo.

Para complementar as informações contidas na dissertação, ao final da mesma se apresentam os anexos, onde se encontram os diagramas, circuitos, o código dos programas assim como informações dos principais componentes eletrônicos usados.

2.0. – Analisando as Soluções Existentes

Neste capítulo pretende-se efetuar um breve estudo de tecnologias atualmente disponíveis e a relevância destas neste trabalho. Estas tecnologias estão relacionadas com os avanços dos sistemas embutidos para a Internet.

2.1. Tecnologia de Sistemas Embutidos para Internet

A tecnologia de sistemas embutidos para Internet é uma das tendências atuais para os fabricantes de pequenos servidores WEB, que são dispositivos que permitem o controle e monitoramento via Internet. Atualmente, equipamentos como impressoras, roteadores, câmeras entre outros dispõe desta facilidade, permitindo, então, fácil configuração e disponibilidade de informações sobre seu estado e operações.

A Internet oferece padronização, um ambiente familiar e custo relativamente baixo. A tecnologia dos sistemas embutidos para a Internet apresenta muitas vantagens ao equipamento que permite que seja controlado remotamente, este pode ser monitorado, atualizado e controlado a distância e poderia ter seu manual armazenados on-line para fácil acesso, por exemplo.

Muitos produtos comercialmente disponíveis tem adicionada a característica de Internet embutida neles. A Hewlett-Packard (HP) **[6]** dispõe de uma impressora, a LaserJet 4100 que é caracterizada por ter embutido um servidor WEB e uma máquina virtual embutida. Estas implementações permitem que o administrador do sistema possa efetuar o acesso e controle da impressora a partir de um browser Web a partir da URL da máquina específica. A impressora pode notificar o administrador através de e-mail de eventuais problemas ou do abastecimento de toner e papel.

2.2. Vantagens em Sistemas Embutidos para Internet

São muitas as vantagens para o projeto e desenvolvimento de dispositivos com Internet embutida. Uma delas, é permitir que um dispositivo possa ser remotamente acessado, e, além disso, tendo o seu uso facilitado pela Internet. Um dispositivo habilitado pela WEB possibilita o uso de capacidades que até então não seriam possíveis para um dispositivo único e isolado, por exemplo, o acesso remoto permite o diagnóstico de falhas, atualizações de firmware e controle de dispositivos remoto que em alguns exemplos pode significar uma redução nos custos de manutenção.

Também, faz uso da Internet como meio facilitador e portador de uma rede que utiliza padrões e protocolos que são já extensamente usados e aceitos, por exemplo, o TCP/IP, reduzindo o tempo de desenvolvimento, pois tudo o que se refere à conectividade já está em seu lugar. Os dispositivos também podem tornar-se mais baratos para fabricar, pois não há necessidade de periféricos do tipo monitores, teclados complementares e outros acessórios. Atualmente o foco da Internet começa a trocar da conectividade baseada em PC para conectividade baseada em outros dispositivos.

2.3. Alguns servidores Web embutidos disponíveis

A maioria dos dispositivos de Internet embutida são baseados em implementações de servidores WEB. O mercado atual caracteriza-se por muitos servidores WEB que são diferenciados por custo, funcionalidades e complexidade. A maioria dos servidores oferece os protocolos e serviços básicos que são necessários para a operação eficaz de um servidor WEB embutido; porém, algumas soluções provêem uma gama enorme de opções e capacidades. Neste documento foram selecionados de quatro servidores WEB embutidos, extraídos de uma extensa faixa disponível no mercado, é descrita abaixa.

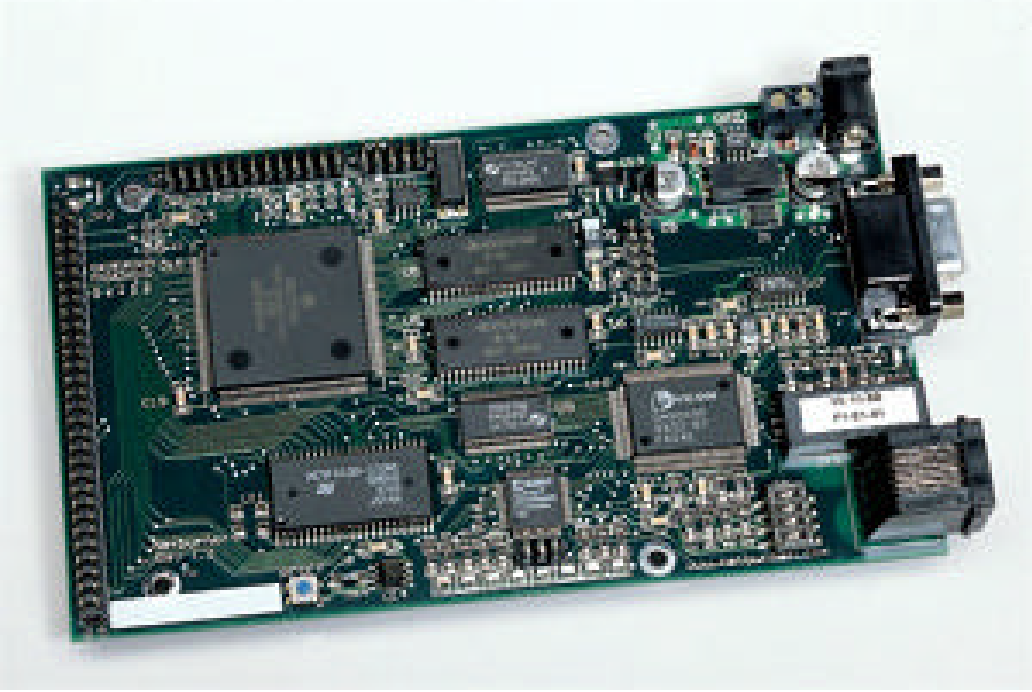
2.3.1. Netburner

O servidor web embutido NetBurner (Figura 2) da empresa homônima **[2]** oferece uma solução completa para habilitar um dispositivo via web. Baseado no processador Motorola ColdFire TM, o NetBurner Network Development Kit (NNDK)

incorpora um sistema operacional Real-Time, pilha TCP/IP, e até 2MBytes de memória FLASH (8K dos quais são usados pelo sistema operacional) e conectividade para a Internet via ethernet 10Base-T ou modem. A pilha TCP/IP suporta diversos protocolos incluindo ARP, DHCP, BOOTP, HTTP, TCP, UDP, POP3, PPP, e Telnet.

É uma implementação de servidor web completa e dispõe de diretório para páginas HTML, imagens e arquivos Java Class, requeridos para applets Java, dentro de um arquivo embutido na runtime da aplicação. O servidor web também suporta geração de HTML dinâmica, que é um componente essencial para aplicações Internet embutidas, e pode enviar e receber e-mail através de uma conexão Ethernet ou PPP. A força do NNDK reside na adição da alta capacidade de memória e o suporte aos diversos protocolos de rede, ou seja, apresenta todas as condições necessárias para desenvolvimento e aplicação de soluções baseadas em servidores web embutidos. Como fator complicador para sua utilização em larga escala, apresenta um preço razoavelmente alto, entre US$ 499 e 995, dependendo da velocidade do processador e da quantidade de memória.

FIGURA 2 – NETBURNER E A DISTRIBUIÇÃO DOS COMPONENTES.

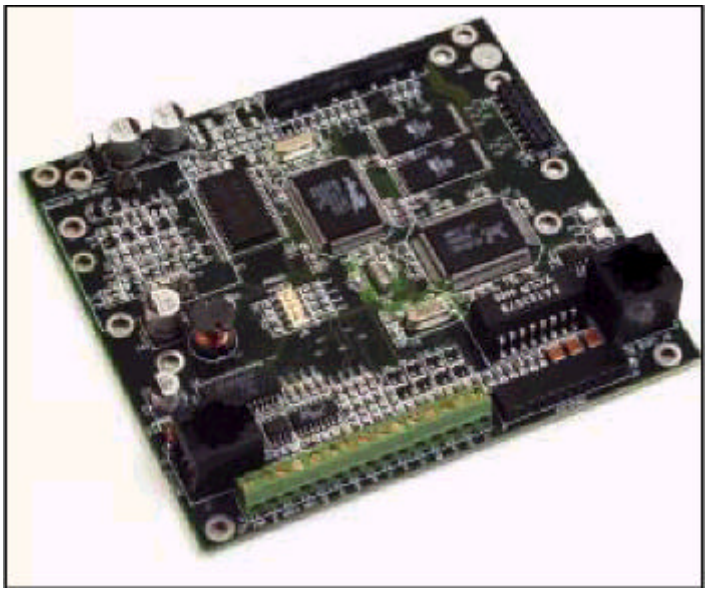


(cortesia NetBurner).

2.3.2. RabbitCore RCM2000

A Rabbit Semiconductor produz um módulo microprocessado com interface padrão Ethernet, o RabbitCore RCM2000 **[5]**, projetado para facilitar a implementação de sistemas embutidos que requerem conexão Ethernet. Entre as características do Rabbit2000, 256KB de memória Flash para armazenar programas e 128KB de SRAM para dados. Também inclui uma porta 10Base-T para conexão em uma LAN ou Internet, e fornece uma pilha TCP/IP com código fonte aberto. O módulo do microprocessador inclui 26 linhas de I/O e 3 portas seriais, o RMC2000 (Figura 3) provê uma conexão completa para Ethernet pelo preço de: US $199.

Figura 3 - O RabbitCore RMC2000



(cortesia da Rabbit Semiconductor).

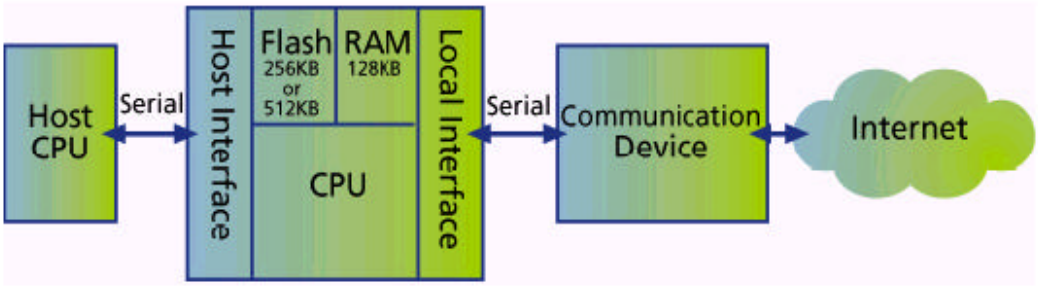
2.3.3. ConnectOne iChip

A ConnectOne's iChip **[4]** comercializa uma solução que integra todas as características de hardware e software para a acesso dial-up à Internet (Figura 4). O iChip trabalha conjuntamente com um host-processor externo, provê acesso completo a uma pilha TCP/IP, que tem associada as funções de rede, através de uma API[1] de alto nível chamada AT+i. Para a implementar uma interface de rede com um protocolo

[1] API é uma abreviação de Application Programming Interface. Todos os sistemas operacionais possuem um conjunto de funções que realizam a interface entre o computador e os programas de usuário. Em qualquer plataforma, estas funções são chamadas API´s, e o conhecimento delas é vital para o desenvolvedor. Por exemplo, para mostrar uma simples palavra na tela de um computador, é necessário conhecer a função da API para este fim, a forma de chamada e seus parâmetros (que são conhecidos como argumentos).

padrão, diferentes versões do iChip podem suportar diversos meios de comunicação com a Internet, podem ser conectados a CPU host sem a necessidade de alguma porta especial, os dados podem ser enviados e recebidos via e-mail, HTTP ou interfaces socket.

FIGURA 4- CONNECTONE'S ICHIP



(cortesia ConnectOne).

2.3.4. Picoweb

O Picoweb é um servidor web miniatura da Lightner Engineering. O projeto do Picoweb é baseado no microcontrolador ATMEL 8515 que também inclui 8k FLASH, 512 bytes de EEPROM, 512 bytes de RAM e 32k EEPROM serial. O microcontrolador 8515 é um processador que opera com freqüência de clock de 8 MHz. O equipamento conta com uma interface padrão ethernet baseada no circuito integrado RTL 8039, fabricado pela empresa Realtek, dotando assim o dispositivo de uma interface compatível com o padrão NE2000.

O projeto do Picoweb **[7]** foi publicado em algumas revistas técnicas reivindicando o título de menor servidor web do mundo. O protótipo apresentado na revista Circuit Cellar em junho de 1999, estimava o seu custo de construção em US$ 25 (vinte e cinco dólares americanos). A versão original utilizava apenas a memória interna do microcontrolador e a comunicação serial não dispunha de circuito integrado para compatibilização de níveis de tensão e corrente e utilizava uma placa de rede com barramento ISA para disponibilizar conexão a redes ethernet. A versão comercial do Picoweb inclui memóia EEPROM serial com 32KB para armazenamento das aplicações a empresa Lightner Engineering vende o dispositivo ao custo de $149 **[11]**.

O firmware do Picoweb inclui um kernel real time, pilha TCP/IP e servidor web http, a conectividade de rede é estabelecida através de um conector RJ-45. A

programação é realizada através da porta paralela de um PC e o Picoweb pode conectar-se a outros dispositivos através da interface serial padrão RS232C.

2.4. Comparação entre os servidores pesquisados

A tabela 1 mostra uma comparação dos servidores web embutidos abordados nesta seção. A comparação dos protocolos de rede suportados, dos meios de conectividade à Internet, e o custo são apresentados. Quando é comparado o aspecto memória se refere à de programa para armazenar o software do lado servidor, e também memória reservada para o armazenamento de elementos do lado cliente, ou seja, páginas WEB, image ns e applets Java, quanto às dimensões físicas foram todos considerados pequenos.

A partir dos dados obtidos, observando as características desejadas e também o preço de compra dos diversos modelos analisados foi possível modelar a aplicação objeto desta dissertação.

Tabela 1 – Comparando entre os varios servidores WEB embutidos.

	NetBurner	**RabbitCore RCM2000**	**PicoWeb**	**ConnectOne iChip**
Protocolos Suportados	TCP/IP Stack ARP DHCP BOOTP HTTP ICMP IGMP POP3 PPP Telnet SMTP SNMP	TCP/IP Stack FTP SMTP TFTP HTTP ICMP	TCP/IP Stack POP3 HTTP SMTP	TCP/IP Stack DNS MIME SMTP POP3 HTTP PPP
Conectividade	10BaseT Ethernet Modem	10BaseT Ethernet	10BaseT Ethernet	 Modem
Memória	512KB-flash 8K Eeprom 4MB DRAM	256KB-flash 128KB SRAM	32KB EEPROM	Depende do Host
Interface Serial	Sim (2)	Sim(3)	Sim	Depende do Host
Custo	**US$ 499-995**	**US$ 199**	**US$ 75**	**US$ 270**

3.0. – Definindo as características técnicas do MSW

Este capítulo desenvolve a implementação do microservidor WEB apresentando as suas características básicas. Serão apresentadas as características e o modelo implementado neste trabalho. Neste capítulo é feita uma revisão sobre aspectos técnicos construtivos, abordagem teórica sobre os componentes eletrônicos utilizados na implementação e são mostrados os recursos disponíveis para o projeto e implementação do MSW.

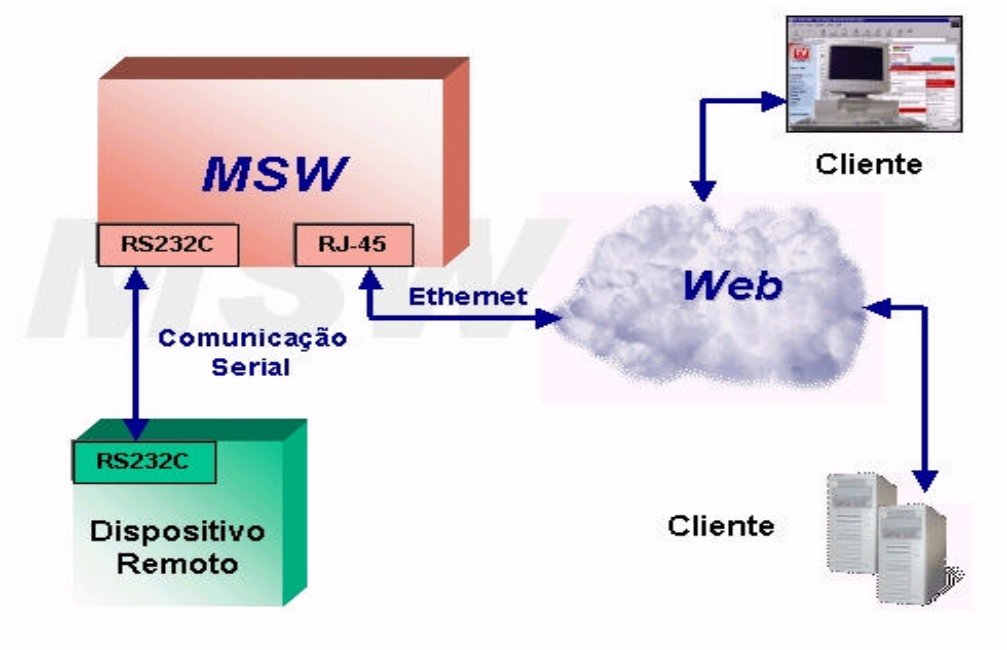
3.1. Avaliação das exigências do sistema

O objetivo básico deste sistema foi descrito brevemente na introdução e será ampliado para mostrar a implementação exata de cada subsistema. Para tomar as decisões pertinentes a sua implementação, é necessário descrever as suas exigências internas e as que são impostas pelos dispositivos a serem controlados, uma delas é a interface entre ambos.

Os dispositivos controlados deverão dispor de interface serial padrão RS232, a conexão serial é particularmente interessante, porque por esta interface podem ser obtidos os dados. Então, a primeira exigência do micro servidor web é dispor de uma interface RS232, padrão. A segunda interface deverá prover uma conexão do MSW com a Internet. As duas opções primárias são Ethernet e acesso discado baseado em modem, mesmo que, uma implementação com modem seja potencialmente mais adequada para um sistema de monitoramento remoto doméstico, as dificuldades técnicas e de logística, tornaram a Ethernet uma escolha mais atraente.

Além de um servidor web que inclui características de I/O serial e Ethernet ou acesso a Internet baseado em modem, também é exigido um protocolo de rede. Além disso, como uma implicação das metas de projeto globais, o MSW deverá ser fisicamente pequeno e deverá ter um custo relativamente baixo, o que nos obriga a um projeto com um preço menor que US $75, o menor preço cotado entre outros fornecedores. A figura 5 mostra uma descrição detalhada da implementação do sistema global e as exigências de servidor web embutido.

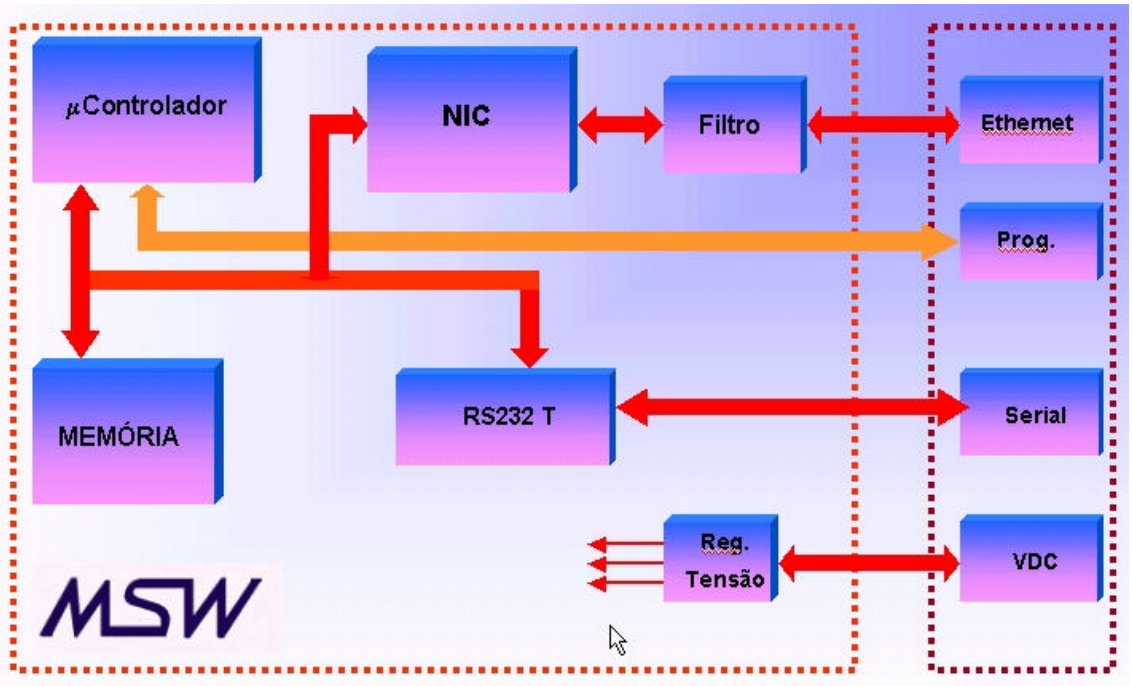
FIGURA 5 – DESCRIÇÃO DETALHADA IMPLEMENTAÇÃO GERAL DO SISTEMA.



Para prosseguimento do desenvolvimento do MSW algumas definições devem ser efetuadas, para definição do microcontrolador a ser utilizado é necessário que seja feita a definição dos recursos adicionais que devem ser disponibilizados tipo: memória para dados, memória de programa e circuitos integrados de interfaceamento. Por se tratar de um servidor Web o dispositivo deverá trabalhar com os protocolos TCP-IP e dispor de servidor http, pretende-se a interação do usuário com o dispositivo o que obriga a utilização de memória adicional, pois os microcontroladores dispõem de baixa capacidade de memória.

Por outro lado, deve-se levar em conta o custo do dispositivo, o que requer o uso de soluções que oferecidas com código livre e também que sejam portadas para sistema operacional livre. Na figura 6 é apresentado o diagrama em blocos do modelo proposto.

FIGURA 6 - DIAGRAMA EM BLOCOS MSW



3.2. – Definição do Firmware

Em termos gerais, quando falamos em firmware, estamos nos referindo a rotinas de software armazenadas na memória de leitura (ROM), isso significa dizer que, rotinas de partida e instruções de I/O de baixo nível ficam armazenadas sob a forma de firmware.

O desenvolvimento integral do firmware para este projeto demandaria muito tempo, devido a sua complexidade e não é o foco principal deste desenvolvimento, que procura concentrar-se na viabilidade de utilização da solução proposta. Existem hoje disponíveis na Web diversos Sistemas Operacionais em Tempo Real disponíveis para implementações em sistemas embutidos, estes podem ser encontrados nas formas de software Freeware, Shareware e simplesmente pagos.

A inteção deste projeto é a de uma arquitetura aberta e utilizando recursos de software tais como driver, kernel, compiladores, e outros, todos de código aberto ou regido por licenças de livre uso. Para a utilização neste projeto foram pesquisadas algumas opções que serão analisadas de forma breve no seguimento desta seção. Cabe

salientar que todas as opções aqui analisadas estão disponíveis, incluindo seus respectivos códigos fonte.

3.2.1. Liquorice project

Amplamente documentado e com os códigos todos escritos em linguagem C, com total permissão de uso e inclusive permitindo modificações no código.O Liquorice Project proporciona um desempenho seguro, alto desempenho e ambiente para desenvolvimento de aplicações embutidas, incluindo:

- Real-time kernel que suporta múltiplas threads, prioridades de threadas e grande quantidade de mecanismos de sincronização.
- Completamente escrito em C.
- Altamente modular, com cada serviço ou mecanismo do kernel implementado como unidade distinta. Isto torna a substituição de módulos relativamente simples, em base de função ou para permitir versões aperfeiçoadas para hardware dedicado.
- Conjunto padrão de manipuladores de dispositivo periféricos e serviços de apoio.
- IP Networking (UDP, TCP, ARP, PPP, SLIP, HTTP, SNMP, etc).
- Biblioteca C para Threads seguras.
- Dynamic binding of services run-time. Isto permite anexar serviços de protocolo a diferentes drivers de dispositivo sem recompilação.
- Escalável para sistemas de 8 bits ou para 16, 32 ou até mesmo 64 bits.
- Suporte para SMP (symmetric multiprocessing).

O Liquorice suporta os microcontroladores AVR Atmel e sistemas baseados no i386 (arquiteturas de 32 bits). As bibliotecas de kernel podem ser usadas com dispositivos que têm o menos 8 kbytes de espaço de código, por exemplo o AT90S8515. Embora seja classificado como um sistema operacional (ou real-time kernel, etc), o Liquorice é implementado como uma série de bibliotecas que rodam no mesmo espaço de endereço que a aplicação. Uma vantagem disso é que são linkadas só as funções que são requeridas para uma aplicação em particular, mantendo o uso de memória tão pequeno quanto possível.

O software foi construído usando o compilador GNU GCC e os GNU binutils. A família AVR é suportada agora na mais recente versão do GCC e dos binutils. O ambiente de desenvolvimento atual é Linux, embora foi projetado e fazer isto é fácil de modificá-lo para ser implementado em outras plataformas de OS. A versão apresentada atualmente, a 0001018 A .Os arquivos do projeto são liberados sob a GNU General Public License v2 com uma cláusula de exceção especial para permitir o unir o código da Liquorice com software não-GPL. **[28]**

3.2.2. UROS - Ultimate Realtime Operating System I

O nome UROS vem de Uros Platise, curiosamente as três últimas letras de seu primeiro nome R O e S, também podem ser interpretadas como RealTime Operating System. O UROS foi projetado para rodar em dispositivos com baixa capacidade de memória como o AT90S8515, porém infelizmente não pode ser compilado pelo Atmel AVR Assembler. Este SO ainda é limitado e somente é compilado no Micro Tools' Assembler (uasm) e no Algebraical Virtual Assembler (ava), desenvolvidos pelo criador do UROS, esforços estão sendo feitos no sentido de porta-lo para a plataforma GNU Atmel, filosofia do autor é semelhante a de Linus Torvalds, porém aqui todo o código está aberto, pois trata-se de um projeto menor que o Linux. **[30]**

O UROS é um sistema operacional pequeno que proporciona unidade de gerenciamento de tarefas e unidades de comunicação as bibliotecas de módulos do kernel já escritas, opção de power saving, pode controlar até 64 usuários e núcleo de tarefa juntos. As características principais do UROS são:

- Sistema operacional Multitasking.
- Task Scheduler pode operar em evento, divisão de tempo ou ambos os modos.
- Kernel debug suporta breakpoint e tracing step by step.
- A comunicação Intertask é feita por sinais e pipes.
- O kernel pequeno e compacto - com todas as características menos de 700 bytes.
- Modo de evento especial pode ser selecionado e requer menos de 410 bytes de memória de programa e só alguns bytes de memória de dados por tarefa.
- Trabalha com dispositivos AVR a partir do AT90S2313.
- Suporta o GNU-C

- A biblioteca padrão do kernel (SKL) dispõe de código assembler já escrito para RS-232 e SPI (Serial Peripheral Interface).

3.2.3. Pico Web

O código fonte do PicoWeb™ pode ser obtido para a versão protótipo na página Web da Lightner Engenering, este firmware pode ser usado para propósitos não-comerciais sem nenhum custo sob a condição de licença GNU.. O firmware pode ser autorizado para uso comercial. Uma licença de uso por hardware pode ser comprada por US $8, da Lightner Engineering, para quantidades maiores o preço pode ser reduzido.

O Sistema de desenvolvimento PicoWeb™, GNU-based é compatível com a versão do compilador C GCC-GNU que suporta código de máquina AVR (em memória de programa). O sistema de desenvolvimento da Lightner Engenering foi direcionado especificamente para o AVR Atmel AT90S8515 sendo constituído de um kernel simples, um debug, um interpretador de pcode (melhor especificado nas próximas seções), driver para adaptador de rede, uma stack TCP/IP e um servidor HTTP. **[3]**

3.2.4. Definição do Firmware

Apesar de ter inciado portando o Liquorice Project, que sem dúvida, é a solução mais atraente do ponto acadêmico, pois apresenta uma ampla documentação e possibilidade ampla e irrestrita de trabalho em todo o código fonte, não foi possível completar a implementação total quando da apresentação desta apresentação.

A segunda alternativa, o UROS Ultimate Realtime Operating System I, apresentado como uma alternativa de uso, não correspondeu, pois está ainda em uma fase intermediária de desenvolvimento, a disponibilidade e interesse manifestados pelo seu criador, cosntituíram-se em aspectos marcantes que incentivam a voltar a utilizar esta alternativa.

Finalmente a escolha recaiu sobre o Sistema de Desenvolvimento PicoWeb™, da empresa Lightner Engenering, que embora não disponha de seu código livre para uso comercial, é liberado para uso acadêmico, constituindo-se na alternativa de implementação mais rápida no momento, a seguir serão fornecidos mais detalhes a seu

respeito.

O Sistema de desenvolvimento PicoWeb, GNU-based é compatível com a versão do compilador C GCC-GNU que suporta código de máquina AVR (em memória de programa). Porém, como há uma quantia limitada de memória de programa em no microcontrolaor (~2K bytes), apenas uma quantia limitada do código de aplicação pode ser escrita em linguagem C, as aplicações mais complexas do MSW deverão ser escritas com o PicoWeb Pcode, que usa rotinas que foram escritas em assembler para uma máquina virtual interpretada de 16-bits, isto torna a programação o código do programa em pcode, mais simples se comparado com o código assembler AVR nativo.

3.3. Definição do Hardware

Após a definição das aplicações do MSW e conseqüentemente dos tipos de conexões necessárias e também da definição do firmware, faz-se necessária à definição dos diversos componentes de hardware do sistema. Os principais componentes a serem definidos podem ser mais bem visualizados no diagrama em blocos, apresentado na figura 6, são: microcontrolador, controlador ethernet, memória adicional para dados, transceiver para comunicação serial e regulador de tensão.

3.3.1. Escolha do microcontrolador

Analizando as familias de microcontroladores existentes no mercado, foram descartadas as de 16 bits, pois são contrárias aos objetivos de simplicidade, difusão e baixo custo. Os microcontroladores de 8 bits são utilizados na maioria dos sistemas embutidos, seu maior campo de aplicação, e isso resulta em menor custo e maior disponibilidade de ferramentas de desenvolvimento e documentação, a figura 7 apresenta comparação entre os modelos analisados, entre as características observdas foi dada especial atenção aos aspectos a seguir:

- Aproximadamente 8K bytes de memória;
- Programação “on-circuit”;
- Facilidade de interfaceamento com barramento ISA;
- Suport ao barramento I2C;

- Ferramentas de desenvolvimento gratuitas e baseadas em sistema operacionais livre.

FIGURA 7 – TABELA COMPARATIVA ENTRE MICROCONTROLADORES

	AT90S8515	PIC16F877
Memória Flash	8 k Bytes	8 k Bytes
Memória RAM	512 Byte	368 Byte
Memória Eprom	512 Byte	256 Byte
Instruções	118	35
Temp/Contador 8 bits	1	2
Temp/Contador 16 bits	1	1
Conversor A/D	Sim	Sim
UART	Sim	Sim
Watchdog Timer	Sim	Sim
Comparador Analógico	Sim	2
Suporte - I2C	Sim	Sim
SPI	Sim	Sim
Preço	6,45	6,9

A escolha do microcontrolador recaiu sobre o AT90S8515, a Atmel levou a filosofia de projeto RISC aos microprocessadores de 8 bit, denominada AVR esta arquitetura proporciona todos os benefícios habituais do RISC: Taxa de clock mais rápida, melhor desempenho, e una otimização mais eficiente no compilador. A AVR compete com várias famílias de microprocessadores, tais como 8051, 6805, 68HC11 e PIC. Outro fator que influenciou decisivamente na escolha do microcontrolador AVR são as atividades que atualmente efetuamos com esta tecnologia em nossos laboratórios e também na atividade acadêmica que exerço.

3.3.2. O Microcontrolador Atmel AT90S8515

O AT90S8515 é um microcontrolador CMOS de 8 bits com baixo consumo baseado na arquitetura AVR RISC. Executando um poderoso jogo de instruções de um ciclo, o AT90S8515 consegue uma capacidade de processamento cerca de um 1 MIPS[2] por MHz , o AT90S8515 pode transferir sinais através de seu barramento, compatível com o padrão ISA, a uma grande velocidade (cerca de 2 MBps), proporcionando grande eficiência na a comunicação de dados, permitindo ao projetista de um sistema otimizar o consumo graças à grande velocidade de processamento do microcontrolador.

A tecnologia AVR combina um razoável set de instruções com 32 registradores de uso geral, iguais de 8 bits e qualquer um deles pode conter endereços ou dados. Como os ponteiros de endereço de 8 bit resultam medianamente inúteis ainda para os dispositivos de 8 bit, os últimos três registradores podem ser usados aos pares, como ponteiros de endereço. Denominados X, Y, y Z, estes três meta-registradores servem para qualquer operação de carga ou armazenamento.Os 32 registradores estão conectados diretamente à unidade lógica aritmética (ALU), permitindo que dois registradores independentes sejam acessados mediante uma instrução simples executada em um só ciclo de instrução. A arquitetura resultante é muito eficiente permitindo uma capacidade de processamento aproximadamente 10 vezes superior aos microcontroladores CISC convencionais. As características básicas são listadas abaixo:

- Utiliza a arquitetura RISC;
- 118 instruções, as mais simples são executadas em um só ciclo de clock;
- 8K bytes de memória Flash;
    - SPI: gravação através de interface serial;
    - Duração: 1000 ciclos de escrita/leitura;
- 512 bytes de memória EEPROM;
    - Duração: 100.000 ciclos de escrita/leitura;
- 512 bytes de memória SRAM;
- 32 x 8 registradores de trabalho de uso geral;
- 32 linhas de entrada/saída programáveis;
- UART programável;
- Interface serial SPI;
- Vcc: 2,7 – 6,0 V;
- Freqüência de funcionamento: 0 – 20 MHz;
- Tempo de ciclo de instrução: 50 ns a 20 MHz;
- Um Temporizador/Contador de 8 bits com preescaler próprio;
- Um Temporizador/Contador de 16 bits com preescaler próprio e modos de Captura e Comparação;
- PWM dual de 10 bits;
- Fontes de interrupção internas e externas (interrupções vetoradas);
- Watchdog Timer programável com oscilador On-Chip;
- Comparador Analógico On-Chip;
- Dois modos de SLEEP: Idle Mode e Power Down Mode;
- Suporte a barramento I2C[3].

O microcontrolador possui características muito interessantes, timer/contadores flexíveis com modos de comparação, interrupções internas e externas, uma UART[4]

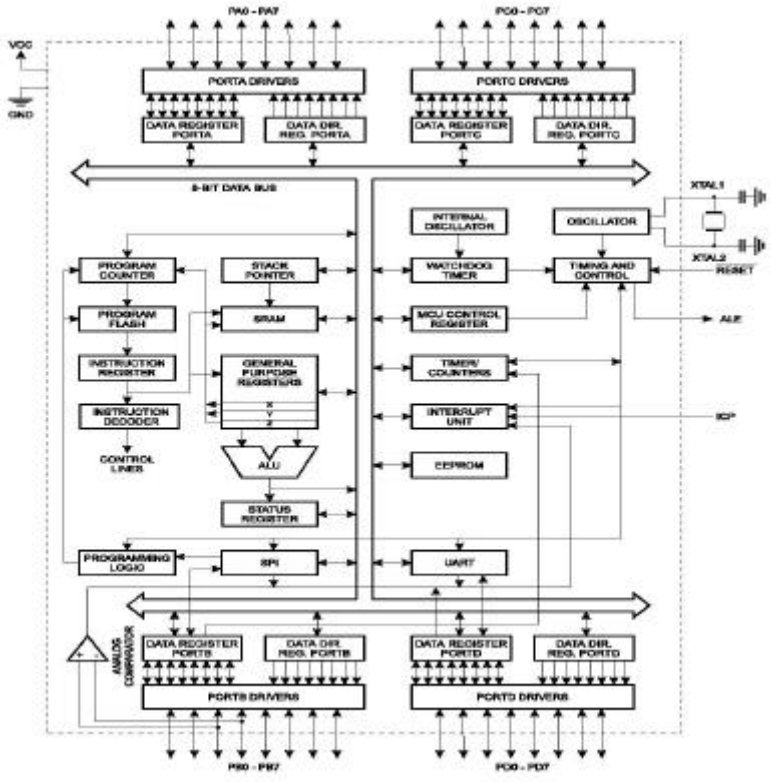
[2] Milhões de Instruções por Segundo, é a medida de frequência de execução de uma dada instrução em uma máquina específica

[3] I2C também conhecido como Inter-IC foi desenvolvido pela Philips para ser utilizado como um padrão de interconexão de diversos circuitos integrados a um único barramento, é um barramento simples de duas vias, bidirecionais.

serial programável, Watchdog Timer programável com oscilador interno, uma porta serial SPI[5], e dois modos de funcionamento selecionáveis por software. O modo “Idle Mode” detém a CPU embora permite que a SRAM, os contadores/Timer, a porta SPI e o sistema de interrupções continuem funcionando. O modo de baixo consumo guarda o conteúdo dos registradores, porém detém o oscilador, desabilitando todas as funções do chip até que se produza uma interrupção ou um reset.

A memória Flash on-chip Downloadable permite que a memória de programa do chip seja reprogramada no próprio sistema através da interface SPI ou mediante um programador convencional. O AT90S8515 AVR é apoiado por um completo conjunto de programas e sistemas de desenvolvimento incluindo: compiladores C, assembladores, simuladores, emuladores de circuitos, e kits de avaliação.A Figura 8 demonstra o diagrama da arquitetura do microcontrolador AT90S8515.

‘.+FIGURA 8 – DIAGRAMA EM BLOCOS MICROCONTROLADOR AVR AT90S8515



Fonte: Data Book AT90S8515 Atmel

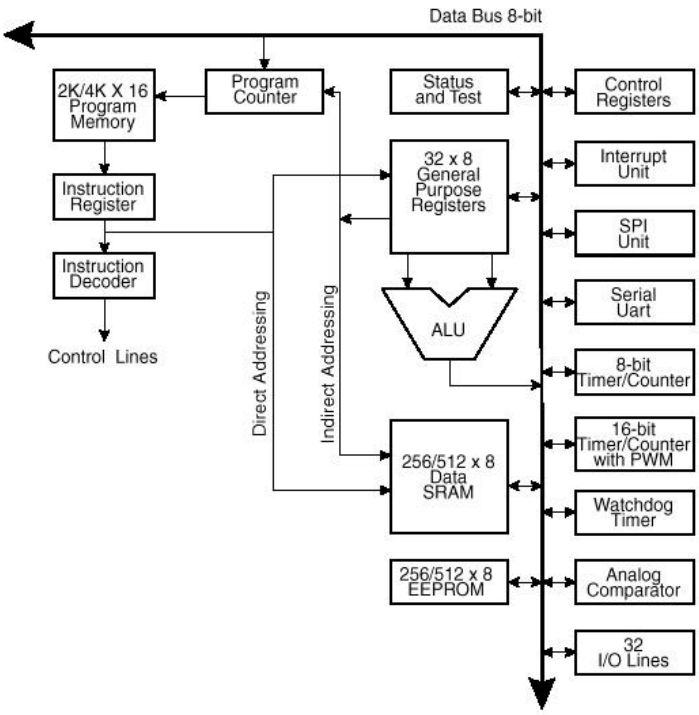
[4] Universal Assynchronous Receiver/Transmitter - Circuito Integrado utilizado para controlar uma interface serial de um computador (porta COM), terminal ou dispositivo similar, necessário para as comunicações seriais assíncronas

[5] SPI (Serial Peripheral Interface). O canal SPI permite a transmissão de dados de maneira síncrona a altas velocidades. A transmissão pode ser realizada entre o microcontrolador AT90S8515 e outros equipamentos e entre microcontroladores do mesmo tipo.

3.3.2.1. A Arquitetura AVR RISC

A arquitetura RISC, (Reduced Instruction Set Computer), requer menos instruções que, por exemplo, a tradicional arquitetura CISC[6], permitindo que os sistemas, nela baseados, possam rodar mais rápido porque o microprocessador tem funções limitadas, em benefício de seu desempenho. Exatamente por isso, a utilização da arquitetura RISC é uma grande tendência da indústria de microcontroladores: uma vez que há um pequeno número de instruções a serem manipuladas. Os benefícios em se utilizar uma arquitetura RISC são chips menores, número menor de pinos, e considerável redução do consumo de energia por parte do processador.

FIGURA 9 – A ARQUITETURA AVR RISC AT90S8515



Fonte: Data Book AT90S8515 Atmel

A arquitetura AVR do AT90S8515, como mostra a Figura 9, combina um rico "instruction set" com 32 registradores de uso geral, todos diretamente conectados à Unidade Lógica Aritmética (ALU), permitindo que as maiorias das 118 instruções (inclusive muitas aritméticas de 16-bits) sejam executadas em apenas um ciclo de clock, o que proporciona um alto desempenho: cerca de 8 MIPS com clock de 8 MHz. Como resultado desta arquitetura se tem uma maior eficiência do código obtendo-se uma

velocidade 10 vezes superior aos microcontroladores CISC [Data book]. Outra característica é que nem todas as instruções estão disponíveis para todos os registros, basicamente os 16 superiores são utilizáveis com quase todas, e os outros 16 inferiores não.

A AVR utiliza o conceito de arquitetura Harvard[7] – com barramentos de dados e instruções separados. A memória de programa é executada em dois estágios de pipeline[8]. Enquanto uma instrução está sendo executada, a próxima instrução é "pre-fetched" (previamente buscada) da memória de programa.

Esse conceito habilita a execução de instruções em todo ciclo de clock. A memória de programa é do tipo flash. Com as instruções relativas de "jump" e "call", todo o espaço de endereçamento de 8 K é diretamente acessado. A maioria das instruções AVR tem um único formato de palavra de 16-bit. Todo endereço da memória de programa contém uma instrução de 32-bit.

Durante as interrupções e chamadas de subrotinas, o endereço retornado pelo "program counter" (PC) (contador de programa) é armazenado na pilha. Todos os programas usuário devem inicializar o "stack pointer" (SP) (ponteiro de pilha) na rotina de reset (antes que subrotinas ou interrupções sejam executadas). O SP 16-bit é lido/gravado no espaço de I/O acessível.

Os espaços de memória na arquitetura AVR são mapas de memória lineares e regulares. Como todo microprocessador RISC, o modelo AVR caracteriza-se por apresentar uma arquitetura Load/Store, homogeneidade nos modos de endereçamento na execução de cada função. A arquitetura Load/Store é uma solução quando não se deseja que instruções referenciem-se à memória com freqüência. Isso porque, este tipo de arquitetura busca seus operandos nos registradores e faz o armazenamento de seus resultados também nos registradores. É exatamente este fator que possibilita a manipulação das instruções em apenas um ciclo de clock.

---

[6] Complex Instruction Set Computer (Computadores com um Conjunto Complexo de Instruções), não é de interesse ser aqui abordada, uma vez que foge ao escopo deste projeto.

[7] A arquitetura Harvard dispõe de duas memórias independentes uma, que contem somente instruções e outra, somente dados. Ambas dispõem de seus respectivos sistemas de barramentos de acesso tornando possível realizar operações de acesso (leitura ou escrita) simultaneamente em ambas memórias.A existência de barramentos de dados e instruções separados permite que sua execução ocorra em paralelo

[8] Pipeline- Ponto ou situação onde a saída de um comando serve como entrada em outro.

3.3.2.2. Memória EEPROM

A EEPROM (Electric Erasable Programable ROM) presente no microcontrolador AT90S8515 possui 512 bytes e está ligada ao barramento de dados 8-bit interno permitindo que possa ser escrita diretamente sobre o microcontrolador durante o processo de gravação ou que o próprio microcontrolador escreva os dados nas posições desta memória. . As memórias EEPROM permitem aproximadamente 100.000 ciclos de gravação/apagamento. O tempo de acesso de gravação é em média de 2,5 a 4 ms, dependendo da tensão a qual é submetida.

Uma função “self-timing” (auto-temporizador), entretanto, permite ao software usuário detectar quando o próximo byte poderá ser escrito. Quando a memória EEPROM é lida ou gravada, a CPU é mantida em dois ciclos de clock, antes que a próxima instrução seja executada.

3.3.2.3. Memória Flash

A memória flash ou flash ROM pode ser definida como “uma memória EEPROM que utiliza baixas tensões de apagamento, e este é feito em um tempo bem menor: em um “flash”, daí seu nome”. O que se observa é que, existem algumas diferenças básicas e importantes entre as duas tecnologias de memória ROM existentes no microcontrolador AT90S8515: o apagamento da memória flash é extremamente rápido e, ao contrário da EEPROM, não é possível reprogramar apenas um único endereço, isto é, quando a memória é apagada, todos os seus endereços são zerados. Na EEPROM é possível apagar o conteúdo de apenas um endereço e reprogramar somente um determinado dado.

O microcontrolador AT90S8515 presente no MSW, apresenta 8 Kbytes de memória Flash Programável on-chip para armazenamento de programas. Como todas as instruções são palavras de 16 ou 32-bits, a Flash está organizada como 4 K x 16.

3.3.2.4. Memória SRAM

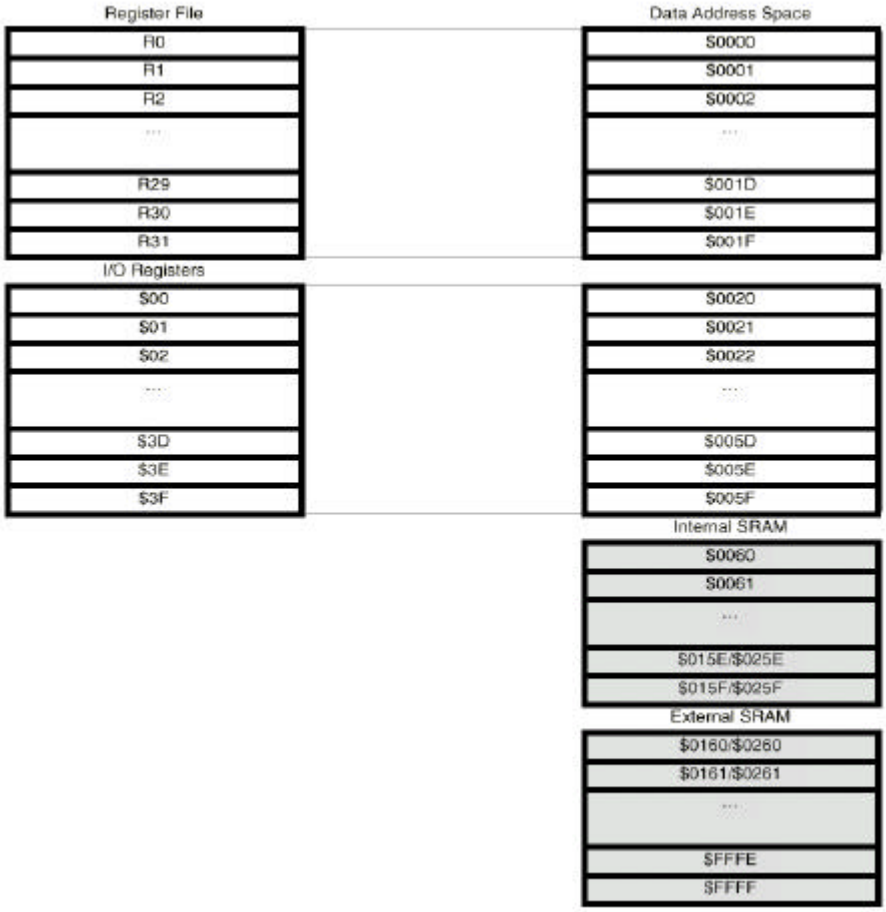
As informações existentes em uma memória RAM (Random Access Memory - Memórias de Acesso Aleatório) não são estáveis e, caso não sejam salvas fisicamente, são perdidas ao se desligar o computador. A chamada memória RAM estática (SRAM), é um chip de memória mais rápida que a DRAM[9]. Não precisa ser renovada (refresh) periodicamente para preservar os dados que contém, desde que a corrente elétrica seja mantida. É em geral, utilizada em caches de alto desempenho, e com o objetivo de acelerar a DRAM. A SRAM utiliza em seu funcionamento, ao invés de capacitores, flip-flops[10] para o armazenamento de cada "0" e "1". É devido a utilização dos flip-flops que os dados são armazenados sem que haja a necessidade de qualquer ciclo de "refresh" (atualização). "Em lugar de um capacitor, a SRAM oferece um circuito completo: no mesmo espaço onde podería-se ter vários capacitores, tem-se somente, alguns flip-flops." (TORRES, 1999). A Figura 13, a seguir, mostra como está organizada a memória SRAM do microcontrolador AT90S8515.

Os 608 endereços baixos pertencem às alocações da Memória de Dados do Register File, à Memória de I/O e a SRAM interna. Os primeiros 96 endereços representam o Register File + Memória de I/O, e os próximos 512 endereços a SRAM interna. Uma SRAM externa pode ser localizada em algum espaço da memória estática. Esta SRAM externa ocupará a alocação seguinte à interna, cerca de 64K, dependendo do tamanho da RAM estática. Quando o acesso aos endereços do espaço de memória de dados excede as alocações da SRAM interna, a SRAM externa é acessada as mesmas instruções que são utilizadas para acessar a interna. Quando o espaço interno de dados é acessado, os pinos strobes[11] de leitura e gravação (RD e WR, respectivamente) estão inativos durante todo este ciclo de acesso.

[9] Dynamic RAM - memória RAM dinâmica, é a memória principal de um computador, bem mais lenta que a memória estática SRAM

[10] Circuito que é capaz de assumir uma de duas condições, "ligado" ou "desligado", em resposta a um sinal de entrada e manter esta condição até que o sinal de entrada mude. (BIGNELL & DONAVAN, 1995)

FIGURA 10 - ORGANIZAÇÃO DA SRAM DO MICROCONTROLADOR AT90S8515



Fonte: Data Book AT90S8515 Atmel

O acesso a SRAM externa utiliza um ciclo de clock adicional por byte se comparado ao acesso da SRAM interna. Isso significa que os comandos LD, ST, LDS, STS, PUSH e POP utilizam um ciclo de clock adicional. Se a pilha está localizada na SRAM externa, interrupções, chamadas de subrotinas e retornos, utilizam dois ciclos de clock extra porque o PC é “pushed” e “popped” (empilhado e desempilhado). Quando a interface da SRAM externa é utilizada com estado de espera, dois ciclos de clock adicionais são utilizados por byte, em outras palavras, as interrupções que já utilizavam dois ciclos de clock extra, precisarão agora de quatro ciclos de clock.

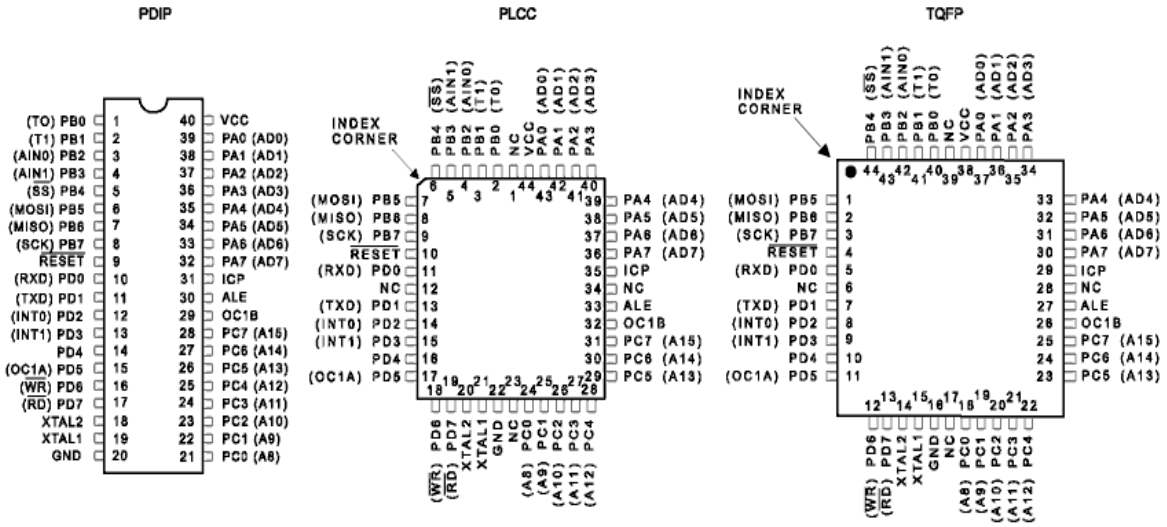
3.3.2.5. Pinagem do AT90S8515

A Figura 11 mostra as três formas de encapsulamento em que o microcontrolador AT90S8515 é dipsonível PDIP, PLCC e TQFP. A descrição funcional

[11] Diz-se strobe o envio de sinal de controle para que haja sincronização ou validação em transmissão de dados

de cada um dos pinos pode ser vista com detalhes no ANEXO 2.

FIGURA 11 – ENCAPSULAMENTO DO AT90S8515



3.3.2.6. Registradores da AVR

Uma característica básica de sistemas microprocessados é a presença de um grupo de registradores internos. A arquitetura AVR presente no modelo AT90S8515 do microcontrolador, apresenta 32 registradores de 8 bits, que podem ser manipulados tanto para leitura como para escrita, como 16 palavras de 16-bit. Além disso, seus três registradores mais altos (X, Y, e Z), podem ser utilizados no endereçamento direto da memória. Há também os registradores de I/O, apresentados no Anexo 1, que são em número de 64 e podem ser endereçados diretamente em instruções de apenas um ciclo de clock.

Todas as entradas do AT90S8515 e periféricos estão localizados no espaço de I/O. As alocações de I/O são acessadas pelas instruções de IN e OUT transferindo dados entre os 32 registradores de uso geral e o espaço de I/O. Os registradores de I/O dentro do espaço de endereço $00 - $1F são diretamente acessados utilizando as instruções SBI e CBI. Nesses registradores, os valores de bits únicos podem ser verificados pela utilização das instruções SBIS e SBIC. Para comandos IN e OUT de I/O, os endereços de $00 - $3F devem ser utilizados. Quando do endereçamento dos registradores de I/O como SRAM, o valor $20 deve ser adicionado a este endereço.

A arquitetura AVR RISC do microcontrolador suporta diferentes modos de

endereçamento. Tais modos são muito poderosos e eficientes para que o AT90S8515, possa acessar sua memória de programa (Flash) e sua memória de dados (SRAM, Register File e Memória de I/O). Os modos de endereçamento do AT90S8515 são relacionados abaixo e podem ser vistos com mais detalhes no Anexo 2.

- Endereçamento Direto com um Registrador;
- Endereçamento Direto com dois Registradores;
- Endereçamento Direto com Registradores de I/O;
- Endereçamento Direto de Dados;
- Endereçamento Indireto com Substituição;
- Endereçamento Indireto de Dados;
- Endereçamento Indireto de Dados com Pré-Decremento;
- Endereçamento Indireto de Dados com Pós-Incremento;
- Endereçamento de Memória de Código Constante;
- Endereçamento Indireto da Memória de Programa;
- Endereçamento Relativo da Memória de Programa.

3.3.2.7. Interrupções

As interrupções são modificações ocorridas no fluxo de controle cuja causa não está relacionada à execução de um programa, mas geralmente a operações de I/O. De modo geral, existem três fontes e interrupção: a interrupção por software (instrução), a pedida por periférico externo e a interrupção pedida por periférico interno (Timer/Contador, porta serial, dentre outros).

O AT90S8515 conta com 13 interrupções diversas, vetoradas[12] no início da memória de programa. Esse fato possibilita a implementação de um código de inicialização, simplificando o trabalho do compilador C. Há registradores de I/O específicos para controlar o nível, a prioridade e a habilitação de cada interrupção. A maior prioridade tem o vetor de RESET e a continuação o vetor INT0 e assim sucessivamente.

FIGURA 12 - LISTA COMPLETA DE VETORES

Nº de vect.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$004	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	$005	TIMER1 COMPB	Timer/Counter1 Compare Match B
7	$006	TIMER1 OVF	Timer/Counter1 Overflow
8	$007	TIMER0 OVF	Timer/Counter0 Overflow
9	$008	SPI, STC	Serial Transfer Complete
10	$009	UART, RX	UART, Rx Complete
11	$00A	UART, UDRE	UART Data Register Empty
12	$00B	UART, TX	UART, Tx Complete
13	$00C	ANA_COMP	Analog Comparator

O tempo de resposta de uma interrupção na arquitetura AVR RISC é de 4 ciclos de clock, tanto para atender quanto para retornar (instrução RETI), ao atender, o valor do PC (program counter) é posto na pilha (stack), e o jump para o vetor é feito; no retorno, o valor da pilha é restaurado no PC e o fluxo do programa continua. Nunca uma interrupção é atendida durante a execução de uma instrução (quando a mesma dura mais que um ciclo), e também, nunca uma interrupção é atendida quando se está retornando de outra, ou da mesma. É executada, no mínimo, mais uma instrução do programa.

O microcontrolador AT90S8515 tem dois registradores de 8 bits para o controle de interrupções, o GIMSK – registrador geral de interrupções – e o TIMSK – registrador para interrupções dos Timer/Counter, quando se produz uma interrupção o bit I do registrador de estado que permite que se possam produzir interrupções é posto a zero de modo que todas as interrupções estejam desabilitadas. Uma vez atendida a interrupção e executado o código de programa necessário o usuário tem que voltar a por este bit em um (instrução SEI) para que de novo as interrupções possam ser produzidas, quando o contador de programa se situa no vetor correspondente para executar uma interrupção e assim executar a rotina programada no flag que assinala a fonte de interrupção que se encontrará no registrador GIFR ou no TIFR que é posto automaticamente em zero.

[12] As interrupções vetoradas são aquelas que possuem o vetor de interrupção (endereço de início da interrupção) fixo, e não pode ser modificado pelo usuário. Para o AT90S8515 todas as interrupções têm um vetor de reset próprio no espaço de memória de programa.

3.3.2.8. Fontes de Reset

Existem três possibilidades para que se produza um Reset:

- Reset na conexão, o microcontrolador se reseta-rá quando for aplicada tensão de alimentação nos pinos Vcc e GND;
- Reset externo, quando um nível baixo é detectado no pino RESET durante mais de dois ciclos de Clock;
- Reset por Watchdog, se o registrador estiver habilitado e seu tempo de temporização é resetado antes de produzir-se à instrução WDR o microcontrolador se resetará.

Uma vez produzido um RESET, todos os registradores do microcontrolador são postos com seus valores iniciais e o programa inicia sua execução no endereço $000. Neste endereço deve colocar-se uma instrução de salto RJMP para executar a rotina de começo de programa. Na continuação do endereço $000 e até o $00d se encontram os vetores para as diferentes interrupções, porém se estes vetores de interrupção não são utilizados pode colocarse o código de programa neles

3.3.2.9. Modos de Sleep.

O AT90S8515 possui dois tipos de sleep mode que são ativados pelo bit SE do registrador MCUCR, quando habilitado o modo SLEEP o conteúdo do arquivo de registradores, a SRAM e a memória permanecem inalteradas. Se ocorrer um Reset mesmo que se esteja em modo SLEEP o microcontrolador desperta e será executa o programa desde o vetor $000. Estas funcões são requisitadas via software , desativando algums periféricos ou até mesmo a CPU , fazendo o consumo baixar para menos de 1 ìA . Os modos de sleep são:

O modo idle é um estado em que se permite ao timer/counter, e ao Watchdog e ao sistema de interrupções que sigam funcionando. Isto permite que o microcontrolador desperte tanto por interrupções externas como internas (produzidas pelos timer´s) assim como as interrupções do Watchdog. Se a interrupção pelo Comparador Analógico não for requerida pelo programa do microcontrolador, o comparador analógico pode ser desativado através do bit ACD do registrador de Controle do Comparador Analógico ACSR. Isto reduzirá o consumo durante o Idle Mode.

No Power Down Mode, o usuário pode selecionar se quer que o Watchdog esteja ativado durante o Power Down Mode. Se o Watchdog estiver ativado quando houver overflow fará com que o microcontrolador seja despertado. Se o Watchdog estiver desativado, somente um RESET externo ou uma interrupção externa podem despertar o microcontrolador.

3.3.2.10. Timers/Contadores

Uma grande vantagem dos microcontroladores quando comparados a sistemas que utilizam microprocessadores está no fato de que, os primeiros já trazem incorporadas estas características, tornando o projeto de hardware e software mais simples.

Especificamente, o AT90S8515 dispõe de dois Timers/Contadores distintos: um de 8 bits (Timer0) e outro de 16 bits (Timer1), ambos de uso geral com interrupções definidas. Todavia, o segundo (Timer 1) conta com um modo de operação para geração independente de PWM (Pulse Width Modulator), além de outros modos como o Capture, que a partir de um sinal externo, para o funcionamento do timer gerando uma interrupção, e o Compare que, também a partir de um sinal externo, compara o valor capturado no timer com um, de dois possíveis, pré-determinados, gerando assim a respectiva interrupção.

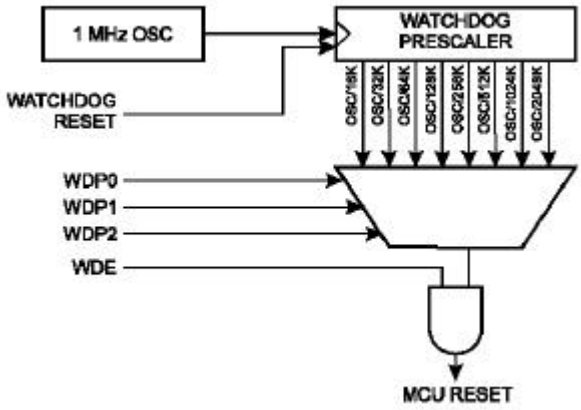
3.3.2.11. WatchDog Timer

O Watchdog Timer do AT90S8515 é acionado a partir de um oscilador on-chip separado, que apresenta uma freqüência de 1 MHz. Esse é um valor típico, quando submetido a uma tensão de 5 V. Através do controle do Watchdog Timer, o intervalo de reset do Watchdog pode ser ajustado.

A instrução WDR (Watchdog Reset) reinicia o Watchdog Timer. Oito diferentes períodos do ciclo de clock podem ser selecionados para determinar o período de reset. Se o período de reset expira sem que haja um outro reset do Watchdog, o microcontrolador AT90S8515 reinicia e executa a partir do vetor de reset. Para prevenir uma “desabilitação” não intencional do Watchdog, uma seqüência especial de “turn-off”

(desligamento) deve ser seguida quando o mesmo é desabilitado.

FIGURA 13 - WATCHDOG TIMER



Fonte: Data Book AT90S8515 Atmel

3.3.2.12. UART

Permite a comunição com outros dispositivos microcontrolados e microprocessados está programada para modo de transmissão/recepção simultânea (full-duplex[13]), para o qual conta com dois registradores independentes: um de somente leitura e o segundo de somente escrita. A UART (Universal Asynchronous Receiver/Transmiter - Transmissor/Receptor Assíncrono Universal) presente na arquitetura AVR é muito semelhante às encontradas em equipamentos convencionais, como em microcomputadores PCs, por exemplo. Capaz de receber e enviar bytes completos, com o stop bit programável permite a otimização da transmissão serial, o que possibilita taxas de comunicação que vão de 2400 a 115200 bauds[14] (faixa da norma RS232C). Há, no entanto, detalhes que tornam tal dispositivo mais robusto, no que diz respeito à agilidade e confiabilidade:

- Possui interrupções independentes para RX Complete, TX Complete e Data Registre Empty;
- Possui um mecanismo de confirmação de dados automáticos, tais como o Framing Error, Overrun e False Startbit;
- Filtro para eliminação de ruídos por amostragem múltipla;
- 8 ou 9 bits de dados;
- Alta taxa de transmissão em baixas freqüências XTAL.

[13] Full-Duplex ou simplesmente duplex, consiste num modo pelo qual os sistemas podem transmitir e receber dados simultaneamente
[14] Unidade de Medida; o número máximo de variações de sinal por segundo

Segundo (SILVA JUNIOR, 1999), existem quatro possíveis modos de operação, denominados modos 0,1,2 e 3; sendo que, destes quatro, o primeiro (modo 0) possui transmissão e recepção síncronas ao passo que nos demais são assíncronas. Especificamente sobre o microcontrolador AT90S8515, o dispositivo serial, a UART, trabalha em modo 2 de operação, neste modo, em cada pacote são recebidos 11 (onze) bits sendo: um start bit (nível 0), 8 (oito) bits de dados, um nono bit (relacionado ao registrador de controle do dispositivo responsável pela comunicação serial), e o stop bit (nível 1). A seguir, nas próximas seções, é feita uma abordagem mais específica dos modos de transmissão e recepção de dados da mesma.

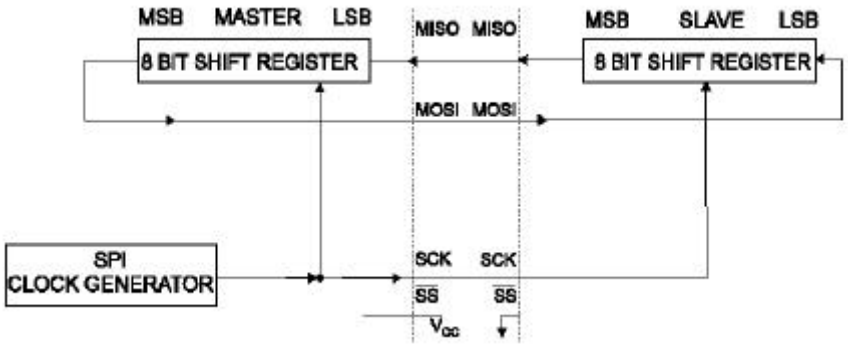
3.3.2.13. SPI

Uma grande vantagem de utilização do AVR é que pode ser reprogramado “em campo” (ISP – In System Programable), isto se consegue utilizando, no modo de inicialização (rst) a SPI (Serial Peripheral Interface) que permite a transferência síncrona de dados a uma alta velocidade, chegando a 2 Mbits, entre o AT90S8515 e seus dispositivos periféricos ou entre vários dispositivos AVR. Suas características incluem:

- Transferência Síncrona de Dados, Full-duplex;
- Operações em modo mestre ou escravo;
- Possibilidade de transmitir primeiro o LSB ou o MSB;
- freqüência máxima de 5 Mbit/s;
- Quatro velocidades de transmissão selecionáveis;
- Flag para a sinalização de fim de transmissão;
- Flag de proteção para a colisão de escrita.;
- Possibilidade de despertar o microcontrolador de Idle Mode (somente no modo escravo).

Este periférico controla um barramento de comunicação síncrona de alta velocidade, chegando a 2 Mbits, tornando-se muito útil na transmissão de dados e essencialmente do programa em si. A interconexão entre dispositivos mestre e escravo com a SPI é mostrada na Figura 14. O pino PB7 (SCK) é à saída de clock para o SPI master e a entrada de clock para o SPI escravo.

FIGURA 14 - INTERCONEXÃO MESTRE-ESCRAVO SPI



Fonte: Data Book AT90S8515 Atmel

Quando o modo SPI está habilitado o endereço de dados nos pines MISO, MOSI, SCK, SS corresponde à seguinte tabela:

FIGURA 15 – TABELA DA INTERFACE SPI

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
SS	User Defined	Input

Fonte: Data Book AT90S8515 Atmel

3.3.3. Adaptador de Rede

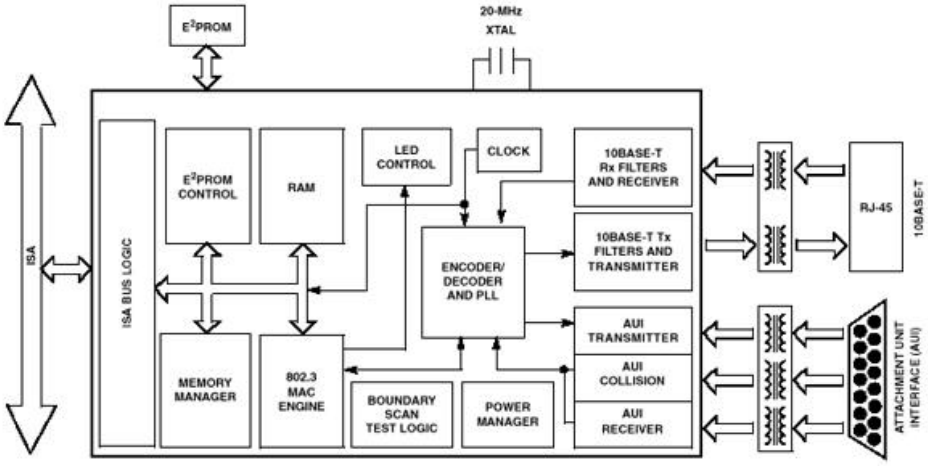
Na implementação do microservidor Chiron foi utilizado um adaptador de rede Ethernet com barramento ISA “NE2000 Compatível”. Para a nova implementação, havia então a necessidade de se encontrar uma solução baseada em um ou mais circuitos integrados que pudessem substituir a placa, uma vez que o nosso propósito ficaria descaracterizado se viéssemos a utilizar uma placa Ethernet plugada ao microcontrolador, foi efetuado um estudo sobre 2 (duas) soluções existentes, e os resultados são mostrados a seguir.

3.3.3.1. Crystal CS8900A

Este circuito é um controlador Ethernet 802.3 full duplex que implementa o nível MAC ao nível de hardware. Ele implementa uma interface ISA e funciona com

tensões entre 3,3 V e 5V. Sua interface padrão é a 10BaseT (RJ45), mas também pode ser implementado com interface AUI para os modos 10Base2, 10Base5 e 10BaseF. Os filtros de transmissão e recepção são incorporados no chip. É oferecido no encapsulamento TQFP 100 e permite o acesso em mode I/O (16 bytes para endereços).Sua arquitetura pode ser vista resumidamente na figura 16.

FIGURA 16 : ARQUITETURA DO CIRCUITO INTEGRADO CS8900A



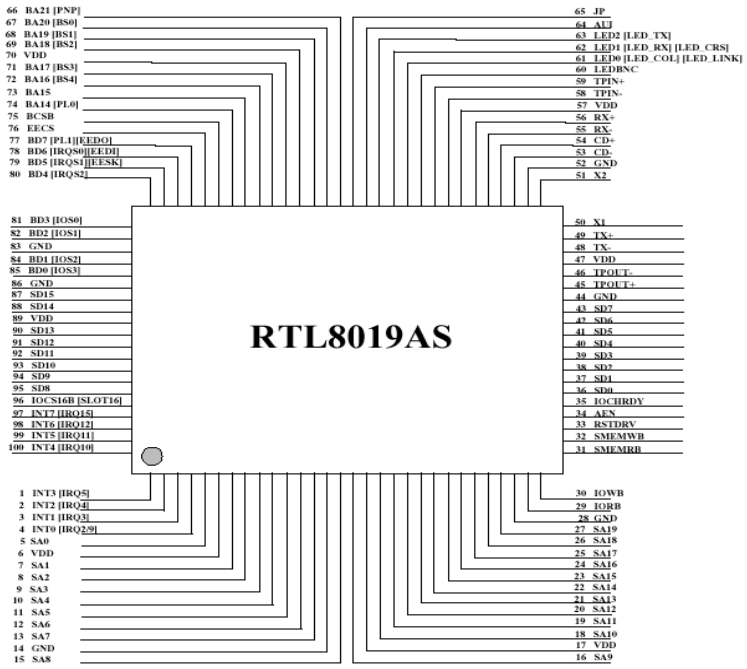
O interesse pelo modo I/O é evidente, pois permite que o CS8900A possa ser interfaceado facilmente com microcontroladores/microprocessadores de 8 bits como o AVR, PIC, 68HC11 e outros. A Crystal fornece gratuitamente os drivers para os ambientes Linux, Microsoft Windows, pSOS, VxWorks, SCO, estes drivers permitem o interfacemanto entre o nível MAC do CS8900A e os protocolos Internet de nível superior (IP, TCP, UDP) que podem ser implemetados em sistemas operacionais com conexão à Internet. **[31][32]**

3.3.3.2. Realtek RTL8019AS

Este circuito é um controlador Ethernet 802.3 full duplex que implementa ao nível MAC e ao nível de hardware. Ele implementa uma interface ISA e funciona com 5V. Sua interface padrão é 10BaseT (RJ45), porém possui possibilidade de interface AUI para os modos 10Base2, 10Base5. Ele é apresentado em encapsulamento PQFP 100 pinos. Sendo acessado em modo I/O. O interessante do modo I/O é que permite que

o RTL8019AS possa ser interfaceado facilmente com microcontroladores e microprocessadores de 8 bits. Sua arquitetura é resumida na figura 17. **[12]**

FIGURA 17 : ARQUITETURA DO CIRCUITO RTL8019AS



A REALTEK fornece gratuitamente os drivers para as palataformas Linux, Microsoft Windows, SCO, estes drivers permitem a interface entre o nível MAC do RTL8019AS e os protocolos da Internet de nivel superior (IP, TCP, UDP) que permitem a implementação de sistemas de navegação e conectividade Internet. A tabela da figura abaixo resume os pontos importantes desta solução.

3.3.3.3. Escolha do Chip Controlador

A escolha recaiu sobre o chip Realtek RTL8019AS, pois embora o Crystal CS8900A disponha de características similares, a grande quantidade de placas comercializada atualmente com chipset Realtek, torna o acesso à documentação mais fácil. Outro motivo que torna muito interessante a utilização deste componente é a limitada memória presente em nosso processador. Enquanto o microcontrolador Atmel apresenta apenas 512 bytes de SRAM, o RTL8019AS apresenta 16 KB de SRAM onchip - isso significa 32 vezes mais que o microcontrolador.

Além disso, oferece um ring buffer, para permitir a recepção de frames Ethernet “back-to-back”, para fins de controle, caso o processador esteja realizando outras tarefas

e outros controladores Ethernet restantes, que são utilizados para agrupar pacotes (Ethernet) transmitidos. Desta forma, dos 512 bytes de SRAM oferecidos pelo microcontrolador AVR AT90S8515, pouquíssimo torna-se necessário para o envio e recepção de pacotes Ethernet. Além, é claro, de possibilitar o envio de um único pacote Ethernet de 1500 bytes.

3.3.4. Memória EEPROM Externa

Além da existente internamente no microcontrolador, há também a existência de uma memória adicional para dados , localizada fora do microcontrolador, esta memória não-volátil terá um papel muito importante no funcionamento do MSW; uma vez que será detentora de todo o código de programação das aplicações desenvolvidas, incluindo dentre outras coisas, programas CGI e imagens JPEG utilizadas na estrutura da página web apresentada pelo MSW. Ela será organizada como um espaço de dados em separado, no qual, os bytes possam ser lidos e gravados.

A utilização de memória serial na implementação do hardware foi motivada pela simplicidade de conexão e uso desta. A tecnolgia escolhida foi de uma memória com suporte para o barramento I2C. Este é simplesmente um protocolo de comunicações desenvolvido pela Philips e esta orientado às aplicações de 8-bits controladas por um microprocessador. O Barramento I2C é multi-master, isto significa que mais de um dispositivo é capaz de controlar o barramento conectado a ele. Os master são geralmente microcontroladores, pelo que um microcontrolador pode ser master ou escravo. A possibilidade de conectar mais de um microcontrolador ao Barramento significa que um ou mais microcontroladores podem iniciar o envio de dados ao mesmo tempo. Os tipos de transferência de dados no barramento são:

- Modo Padrão aproximadamente a 100 kBits/Sg.
- Modo Rápido aproximadamente a 400kbits/Sg.
- Modo Alta velocidade mais de 3,4 Mbits/Sg.

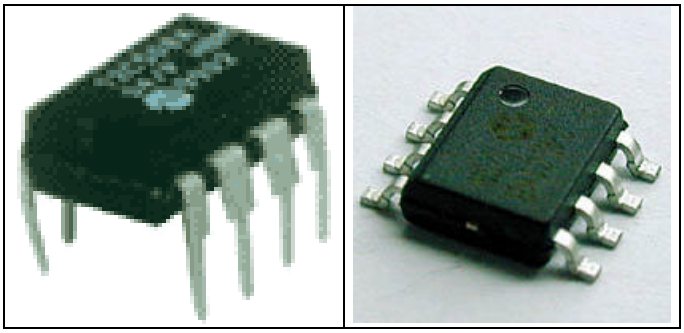
O barramento I2C suporta qualquer tipo de componente (NMOS, CMOS, bipolar, etc.), dispõe de duas linhas físicas uma de dados (SDA) e outra de clock (SCL) que transportam a informação entre os diversos dispositivos conectados ao barramento.

Cada dispositivo é reconhecido por um único endereço e pode operar qualquer como transmissor ou emissor de dados, dependendo da função do dispositivo.

Observando estas características foram pesquisadas as memórias fabricadas pela Microchip Inc, série 24AA515/24LC515/24FC515 (24XX515 *) são memórias do tipo EEPROM serial com capacidade de 64K x 8 (512K bit), capazes de operar com uma faixa de voltagem de 1.8V a 5.5V e foram desenvolvidas para aplicações avançadas de baixo consumo como comunicações ou aquisição de dados. Este dispositivo tem capacidade de até 64 bytes de dados nos modos byte-write e page-write e também é capaz de leituras sequenciais ou randomicas. As leituras podem ser seqüenciais dentro das faixas de endereços 0000h-7FFFh e 8000h-FFFFh.

As linhas de endereço funcionais permitem até quatro dispositivos no mesmo barramento de dados. Isto permite para até 2Mbits de memória total de EEPROM no sistema. Maiores detalhes sobre a memória EEPROM externa do MSW podem ser encontrados no ANEXO 5.

FIGURA 18 – ENCAPSULAMENTO DO CHIP 24XX515

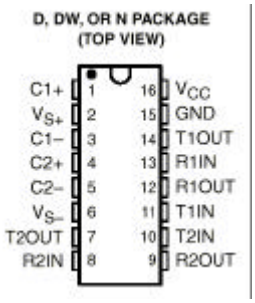


Fonte: Microchip 24xx515 Data Sheet

3.3.5. Comunicação Serial RS-232

Para a conexão serial RS232C será utilizado o dispositivo MAX232 da Maxim [22] que é um driver/receiver dual que inclui um gerador de voltagem capacitivo para prover voltagem nos níveis determinados pela EIA-232 a partir de uma fonte de 5-V. Cada receptor converte entradas EIA-232 para níveis de 5V compatíveis TTL/CMOS. Estes receptores têm um típico (threshold) limiar de 1.3 V e uma hysterese típica de 0.5 V, e pode aceitar entradas de ±30-V. Cada driver converte entradas níveis TTL/CMOS em níveis EIA-232.

FIGURA 19 – PINAGEM DO MAX 232

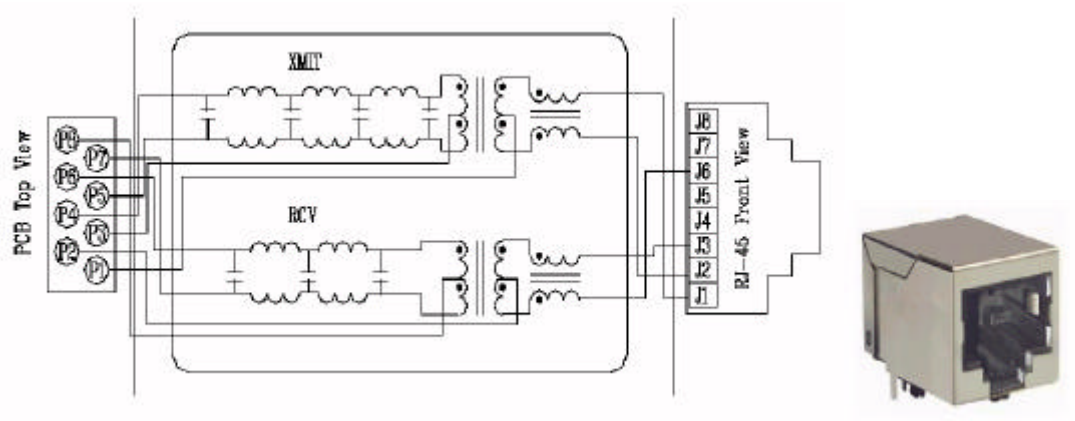


O MAX232 é adequado para utilização em alimentação de nível TTL, pois necessita de apenas quatro componentes externos, quatro capacitores de 10uF , responsáveis pela elevação da tensão para 12 V simétricos necessários para a RS232. A saída esta disponível em um conector DB25 com pinagem padrao, estando desta forma a mais funcional possível para ser utilizada diretamente ligada a porta serial de um microcomputador.

3.3.6. Transformador e filtro Ethernet

As placas adpatadores de rede comerciais, normalmente utilizam conectores e transformadores em compontes separados, para a implementação do MSW será utilizado o LF1S022 fabricado pela empresa Bothhand e que integra transformador e conector em uma mesma peça e ocupa o mesmo espaço na placa de circuito impresso, outro aspecto positivo é que esta integração possibilita uma melhor performance, segundo observado na documentação do fabricante**.[27]**. O diagrama esquemático é a fotografia do componente na podem ser vistos na figura 20.

FIGURA 20 – ESQUEMA ELÉTRICO DO LF1S022

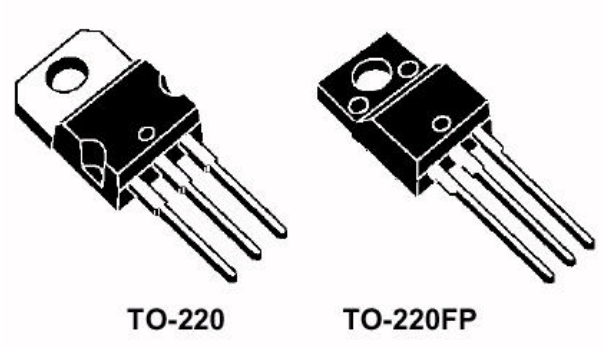


Fonte: Bothhand – LF1S022 Data Sheet

3.3.7. Fonte de Alimentação

A fonte de alimentação "onboard" é baseada no 7805 (figura 21) que é um regulador de tensão padrão ele prove voltagem para o dispositivo a partir de tensão DC que entra através do conector de alimentação. O circuito dispõe de um diodo retificando para proteção do circuito para tensão reversa e capacitores para filtragem e desacoplamento. **[40]**

FIGURA 21 – REGULADOR DE TENSÃO PARA 5 V



Fonte: ST L800 Series Data Sheet

3.3.8. Fontes de Clock

O dispositivo conta com duas fontes de clock, uma para suprir o microcontrolador e também para servir como referência para a geração de taxa de baud rate para a porta de comunicação serial, composta de cristal de 8MHz e dois capacitores cerâmicos de 22pF e outra composta por cristal de 20MHz para suprir o circuito integrado RTL 8019AS, que faz a parte de interfaceamento Ethernet. Para o clock do microcontrolador o valor ideal para o cristal é de 7.32MHz, que permitiria trabalhar com baud rate mais elevado, a utilização do cristal de 8MHz deveu-se a dificuldade em adquiri-lo.

4.0. – Implementação do Hardware

Este capítulo apresenta uma estrutura formal para o MSW a partir de linha de pesquisa empregada na análise e síntese para a solução dos problemas discutidos nos capítulos anteriores. Nele é concretizada a proposta de desenvolvimento do MSW com base nas características introduzidas nos Capítulos 2 e 3 dentro das especificações técnicas propostas no Capítulo 3

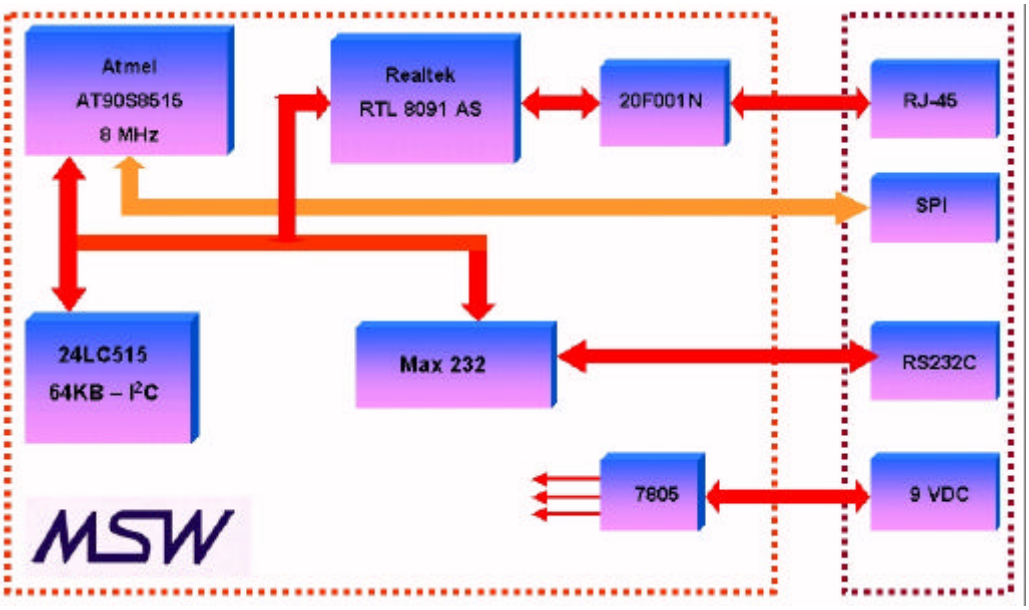
4.1. Descrição funcional dos componentes eletrônicos

O hardware do MSW (ver diagrama elétrico no Anexo 1) é comandado pelo microcontrolador AT90S8515 da Atmel e utiliza o controlador para Ethernet RTL 8019AS, que está conectado diretamente às portas do microcontrolador, dispensando, desta maneira a utilização de circuitos integrados nas funções de “drivers” ou “latchs” pois não existe a necessidade de criação de algum tipo de barramento.

O hardware do microservidor é formado por 4 (quatro) circuitos integrados e mais alguns componentes discretos, neste capítulo será analisada a conexão destes circuitos integrados. Como mostra a Figura 23, o circuito está centralizado no microcontrolador Atmel AT90S8515, que dispõe internamente de três tipos de memória: RAM estática (512 bytes), memória flash programável (8 Kbytes), e EEPROM[15] “on-chip” (512 bytes), em sua arquitetura AVR RISC, não serão abordadas outras características pois este componente foi abordado com detalhes no capítulo 3.

[15] EEPROM (Electric Erasable Programable Read Only Memory) A vantagem da memória EEPROM é de ser uma RAM não é volátil, ou seja, que uma vez produzido um RESET o valor que tenham os registradores da EEPROM não se perderão. Por esta razão um uso muito freqüente das memórias EEPROM dos microcontroladores é para armazenar as chaves ou códigos de acesso para distintas aplicações.

FIGURA 23 – MODELO PROPOSTO



Para o armazenamento das aplicações é utilizada a memória I^2C EEPROM Serial, modelo 24LC515, fabricada pela Microchip, que possui 64 Kbytes sendo responsável pelo armazenamento do código das aplicações e imagens, entre outros, é uma espécie de “sistema de arquivos” do MSW. A estrutura do sistema de arquivos da I^2C EEPROM é igual à estrutura de arquivo da FlashROM, com uma exceção – os itens do diretório iniciam em um endereço fixo. Uma string precede a identificação da estrutura de diretório na I^2C EEPROM e indica que é uma memória externa para dados WWW.

Para o interfaceamento Ethernet é utilizado o circuito integrado RTL8019AS, que é um chip moderno do tipo "tem tudo em uma única implementação de chip", padrão NE2000, integra 16 kbytes de SRAM, modulador e demodulator para a interface física, controlador de protocolo Ethernet, entre outras funções. De forma resumida pode-se dizer que o RTL8019AS, dispõe de todos os requisitos necessários para transmitir e receber pacotes Ethernet.

Os pacotes recebidos ou prontos para serem enviados são armazenado na memória de on-chip que é acessível externamente através de um canal de DMA. A parte interna do chip é controlada através de quatro bancos de registradores. O número dos registradores é especificado diretamente através dos sinais de endereçamento de A0 a A4, em nosso dispositivo conectados aos sinais PC0, PC1, PC2, PC3 e PC4 do

microcontrolador. Os primeiros 16 registradores são usados para controlar as operações de leitura do dispositivo. Os próximos oito registradores 16 a 23 são de fato um único registrador de dados que serve para comunicação de DMA. Permanecendo oito registradores de 24 a 31 também, como um único registrador que é usado para reinicializar o dispositivo. Os bancos de registradores não são diretamente endereçáveis, ao invés disso dois bancos são usados para chaveamento de bits no registrador 0, esses podem ser acessados por todo os quatro bancos de registradores.Os pacotes recebidos ou transmitidos pelo RTL8019 podem consistir de pacotes de 60 a 1514 bytes, a composição destes é mostrada na tabela abaixo:

Tabela 2 – Formatação da informação no RTL8019

Endereço MAC do nó destino	6 bytes
Endereço MAC do nó que transmite	6 bytes
Tamanho do Pacote	2 bytes
Dados	46 to 1500 bytes

O último circuito integrado à ser comentado é o MAX 232 é um driver/transceiver para RS232C, fabricado por diversas empresas, o detalhes funcionais deste componente foram abordados no capítulo 3. Os demais componentes que integram a parte elétrica do projeto tais como capacitores, resistores, diodos e conectores podemser visualizados no diagrama elétrico, anexo 1.

4.2. Montagem do protótipo

Definidos o diagrama elétrico tornou-se necessário montar protótipo para validar o diagrama e testar a conexão entre o microcontrolador e a NIC (network internet controller), a implementação do protótipo utilizando uma placa padrão é uma placa adaptadora para ethernet, NE2000 compatível e com barramento ISA.

O conector da placa ISA é conectado na PORTA e PORTC do microcontrolador AVR, a PORTA é usada como um barramento de dados de 8-bits bidirecional entre a placa Ethernet e o microcontrolador AVR (D0 - D7 da placa ISA). As 5 linhas baixas da PORTC são usadas como linhas de endereços (ISA A0 - A5). As próximas duas linhas são os strobes de Read e Write, ativos em nível lógico baixo, (ISA -IOR e -IOW). A

última linha de sinal é o reset de hardware para a placa ethernet, ativo em nível lógico alto (ISA RESET).

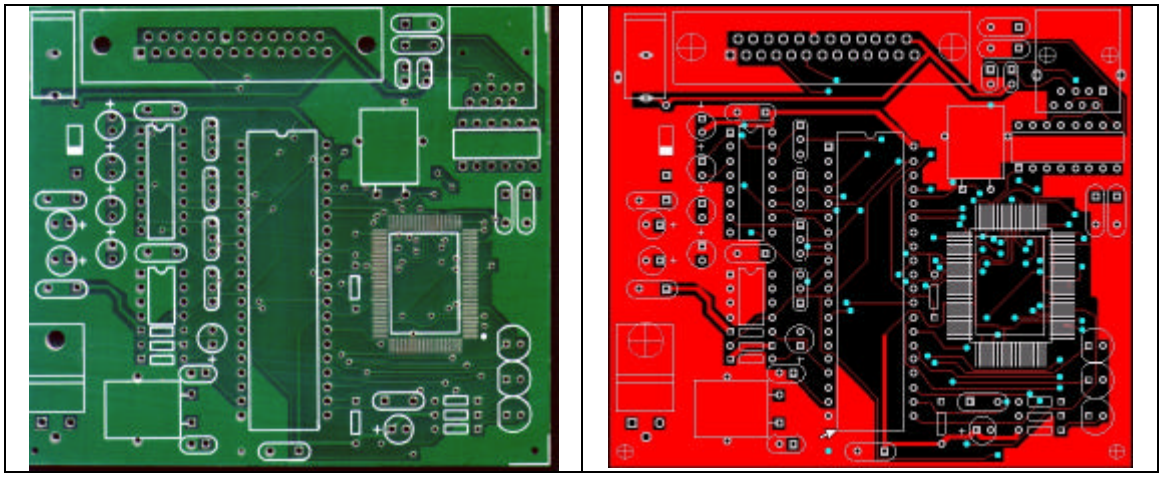
O conector ISA teve vários pinos os quais foram permanentemente conectados ao terra ou VCC da placa. Os sinais –SMEMR, -SMEMW, A8, A9 são permanentemente setados em nível lógico alto, as linhas A5-A7, A10-A19, e AEN são permanentemente conectadaas ao terra. O endereçamento da placa Ethernet foi feito através dos 5 bits de endereços (A0 - A4) vinda para oAVR para ser adicionada para um permanente offset de 0x300, permitindo ao AVR para os endereços de localização de memória 0x300 - 0x320 usando somente 5 linhas de endereços.

Convenientemente, esta é exatamente uma faixa de endereços usada por uma placa NE2000 Ethernet. Muitos dos pinos não utilizados no conector ISA permaneceram desconectados, entre estes os sinais: +12v, -5v, -12v, DMA, IRQ e os sinais para transferência em dados de 16-bits. O protótipo impelementado pode ser visto na figura 23.

FIGURA 23 - PROTÓTIPO

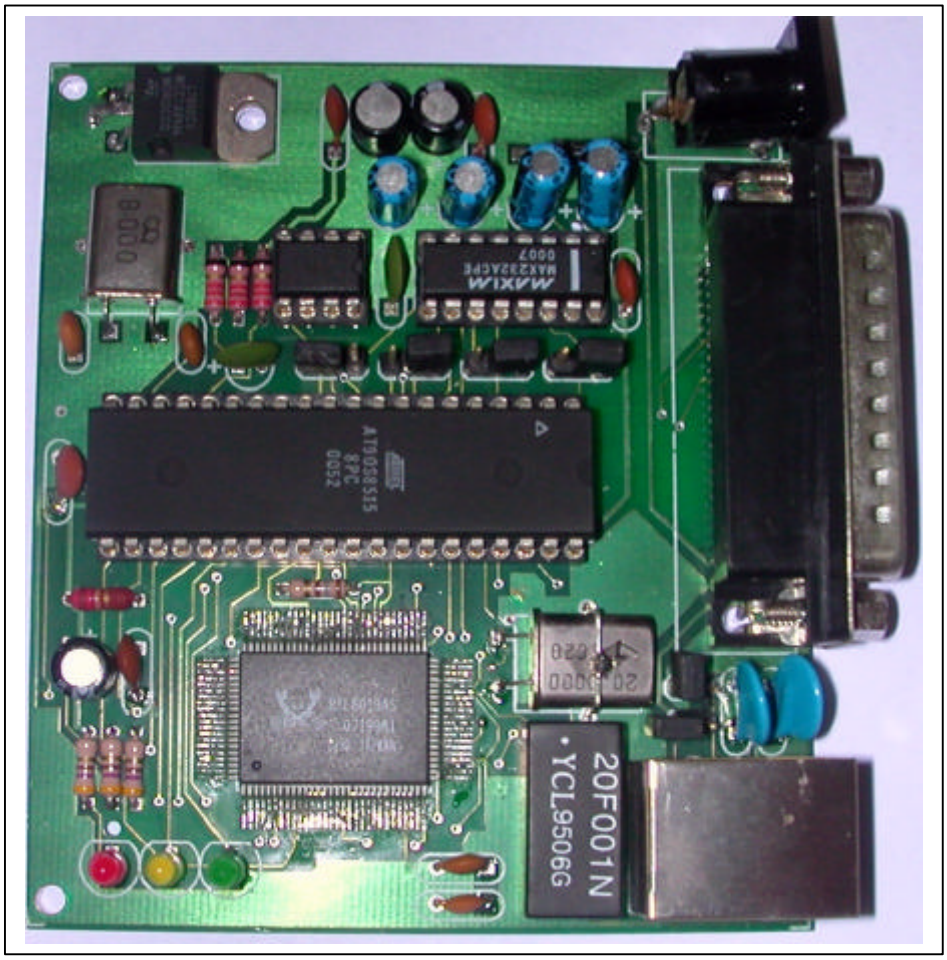
Após a montagem do protótipo e depois de serem realizados todos os testes, foi inciado o trabalho para confecção de placa de circuito impresso definitiva, as ferramentas utilizadas foram os softwares Tango 2000 PCB e Schematic, da empresa EDA. Os resultados obtidos são mostrados na figuras 24 e 25.

FIGURA 24 – PLACA DE CIRCUITO IMPRESSO



(a) Protótipo Confeccionado **(b) Formato Tango2000**

FIGURA 25 – PRODUTO FINAL



4.3. Outras Características

4.3.1. Porta de Comunicação Ethernet

O MSW dispõe on-board de conector padrão RJ-45 para par trançado. Esta porta é conectada ao controlador Ethernet Realtek 8019AS através de um transformador/filtro padrão 10Base-T. Esta interface suporta o tamanho máximo de cabo de 100 metros entre a placa e um HUB. A descrição dos pinos é mostrada na figura 26.

FIGURA 26 – DESCRIÇÃO DOS SINAIS DO CONECTOR ETHERNET

Pino	Sinal
1	TPTX+
2	TPTX-
3	TPRX+
4	TP1+
5	TP1-
6	TPRX-
7	TP2+
8	TP2-

4.3.2. Porta de programação In-System

Esta porta é utilizada para programação do microcontrolador, não sendo necessário retira-lo do sistema. A transferência dos dados é feita através da porta paralela de um microcomputador compatível PC. A descrição dos sinais é vista na figura 27.

FIGURA 27 – SINAIS DA PORTA SPI

Pino	Sinal	Descrição do sinal
25	MOSI	Slave serial data input
20	RST	Reset deverá estar em nível baixo durante a programação
21	SCK	Slave input serial clock
23	MISO	Slave serial data output
13	GND	Sinal de ground

4.3.3. Porta de Comunicação Serial

O MSW dispõe de uma porta serial on-board que pode ser acessada através de um conector DB-25 para comunicação de dados padrão RS-232. Esta porta é conectada ao microcontrolador através de um único chip, que é um RS-232 driver/receiver

interface circuit , o qual converte as tensões para os níveis de 5V requeridos. Os sinais disponíveis no conector DB25 são o sinal de RX no pino 16 e TX no pino 17 do conector. A figura 28 mostra a descrição completa dos pinos do conector DB25 na placa.

FIGURA 28 – CONEXÕES DO DB25

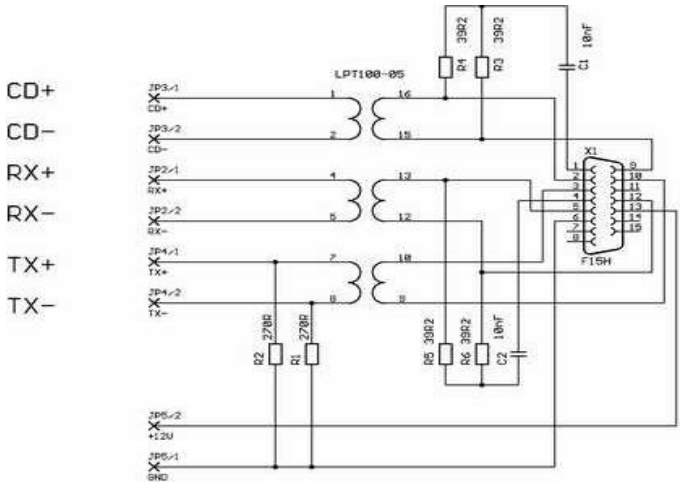
Pino	Sinal	Pino	Sinal	Pino	Sinal	Pino	Sinal
1	PD4	8	PB1	15	ICP	22	PD7/SCL
2	PB4	9	PD6	16	RXD	23	PB6/MISO
3	TXD2/PD5	10	PB0	17	PD1/TXD	24	RXD2
4	PB3	11	PD0	18	OC1B	25	PB5/MOSI
5	PD3	12	PB5	19	RESETB		
6	PB2	13	GND	20	GND		
7	PD2	14	PWR	21	PB7/SCK		

4.3.4. Interface de rede AUI

Para a comunicação com fibra ótica, pelo menos três componentes são necessários: Transmissor ótico e seu driver, Cabo ótico e Receptor ótico com circuito interno para “moldar” sinais. O transmissor e receptor ótico, junto com os circuitos elétricos apropriados, normalmente são parte do módulo da MAU (Unidade de Acesso ao Meio), ambos os módulos normalmente são externos e conectados à AUI, como também poderão ser módulos internos disponíveis em algum HUB ou SWITCH e conversores de mídia, os quais podem ser simplesmente conectados ao conector RJ45.

Para o caso específico do MSW, a interface AUI não está implementada no projeto atual, porém sua implementação é relativamente simples, é necessário acoplar um conversor com transformador de isolamento para os sinais de TX/RX/CD. Na figura 29 pode ser visto um circuito completo utilizando o transformador, 16PT-005A, muito utilizado em placas NE-2000 compatíveis, conectado entre o AUI e a placa do MSW. A interface AUI poderá ser conectada através de um cabo nos sinais internos da placa, RX, CD e TX. Os sinais TX+ e TX - precisam de resistor de pull-up de 270R. Maiores detalhes do diagrama elétrico podem ser vistos no Anexo 7, onde é mostrado parte do diagrama utilizado pela empresa Realtek em seus adaptadores para rede, com barramento ISA e compatíveis com o padrão NE-2000.

FIGURA 29 – CIRCUITO PARA INTERFACE AUI



4.3.5. LED´S Indicadores e Sistema de Clock

O MSW é dispõe de três LEDs. Um D1 é ativado durante a programação do microcontrolador, os outros D2 e D3 são usado para indicar atividade na porta Ethernet, transmissão e recepção.

No sistema de clock do MSW existem duas fontes uma de 8MHz para o microcontrolador e comunicação serial e outra de 20MHz para o circuito controlador da interface ethernet.

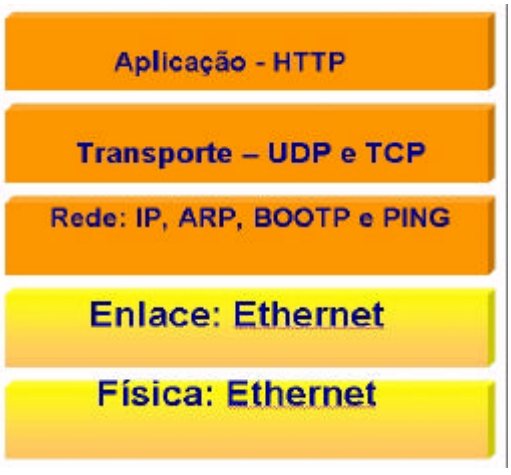
5.0. Implementação do Software

Este capítulo tem por objetivo apresentar o software básico (firmware) definido no capítulo 3, apresentando detalhes de funcionamento no âmbito deste projeto e também discorre sobre a organização e característica do software das aplicações.

5.1. O software básico do MSW

O firmware utilizado é composto de um kernel simples, a organização é mostrada na figura 30. O controlador ethernet proporciona suporte para as duas camadas mais baixas de controle, por conseguinte o microcontrolador Atmel cuida de todas as que está acima. As duas próximas camadas contêm o driver do adaptador de rede e a pilha TCP-IP. A camada de aplicação é o servidor web http. Em relação à alternativa utilizada foram necessárias revisões e implementações de novos códigos, um deles foi à implementação para o protocolo UDP, não contemplado.

FIGURA 30 – CAMADAS DO FIRMWARE UTILIZADO NO MSW



Assim sendo, o firmware padrão do MSW apresenta um kernel simples, um debug minúsculo, um interpretador para pseudocódigo (pcode), um driver adaptador de rede, uma pilha TCP/IP e um servidor HTTP (uma vez que, a função inicial do MSW é prover acesso web como servidor). Uma vez conectado à rede, o firmware do MSW

suporta uma série de protocolos de rede, que merecem aqui uma explanação mais detalhada.

Em um nível mais baixo, o MSW responde a solicitações de rede ARP (Address Resolution Protocol), permitindo então, que outros computadores façam uma associação entre o endereço IP determinado para o MSW e seu endereço Ethernet. Para todo MicroServidor é fixado um único endereço Ethernet que é enviado como parte do pacote ARP replicado.

A seguir, temos a solicitação de BOOTP[16]. O endereço IP do MSW pode ser determinado de forma estática ou dinâmica: estática porque ocorre o armazenamento do endereço IP do dispositivo na memória flash do microcontrolador; e dinâmica através da utilização do protocolo BOOTP. Nesta última, solicitações do tipo BOOTP são enviadas periodicamente pelo MSW, até que uma replicação apropriada seja recebida. É também transmitido como parte do pacote BOOTP solicitado, o endereço Ethernet único do MSW. Se uma resposta de BOOTP válida é recebida, é responsabilidade do MSW utilizar seu conteúdo para configurar seu endereço IP.

O microservidor também responde a ICMP[17] Echo Requests (Solicitações de Ressonância ICMP). Essas solicitações de ressonância ICMP são, normalmente, geradas pela utilização do comando ping, em um host remoto e permitem também a avaliação do tempo de viagem (Round Trip Time - RTT) de uma ressonância solicitada e/ou replicada.

Faz ainda parte do potencial do microservidor, enviar e receber pacotes UDP (User Datagram Protocol). Este é um dos protocolos de transporte sem conexão que antecederam o TCP/IP, utilizado em aplicações como o gerenciamento de redes (SNMP[18]) e de serviços de fornecimento de nomes de domínios na Internet (DNS[19]). No nível TCP/IP, o MSW responde a solicitações HTTP GET (mais tarde abordadas para

[16] O protocolo BOOTP é utilizado para efetuar a partida em redes IP. Permite que uma pilha IP mínima sem informação de configuração, típicamente armazenada na ROM, obtenha informações suficiente para começar o processo de descarregar o código de inicialização necesario. As especificações do BOOTP podem ser encontradas nas *RFC 951* e *RFC 1497*.

[17] Internet Control Message Protocol - protocolo utilizado na Internet e que define mensagens de erro, o funcionamento de pacotes de teste e o comando ping

[18] Simple Network Management Protocol - protocolo utilizado pra monitorar e controlar serviços e dispositivos de uma rede TCP/IP

uma explicação mais detalhada) que são endereçadas à porta TCP80. As solicitações HTTP GET, são enviadas pelos browsers de rede. O MSW responde a essas solicitações, retornando documentos HTML, textos, imagens, e assim por diante, como se fosse um servidor web real. O diferencial aqui, é que não há a necessidade de utilização de um Sistema Operacional gigante, para que se possa obter os mesmos resultados. O kernel do firmware do microservidor provê todo o código necessário para implementar esses protocolos da Internet acima listados.

O firmware do restringe ao máximo o tamanho de uma resposta HTTP GET para um único pacote Ethernet (não mais que 1400 bytes), para que se possa conservar recursos de memória. Diversas técnicas de codificação HTML podem ser utilizadas para que se possa trabalhar dentro desses limites, incluindo frames HTML e múltiplas imagens GIF e JPEG. A resposta básica do firmware para uma solicitação HTTP GET é o retorno de uma página web.

O MSW permite a utilização de JavaScript (embutidos ou em código HTML ou como arquivos separados) e Java applets para serem armazenados em sua EEPROM serial, juntamente com código HTML e imagens.

Há ainda, no firmware, a presença de um debugger simples que propicia, dentre outras coisas, “dumps” de memória, alterações da EEPROM e pcode. Existem rotinas de firmware permitem que a EEPROM serial 64 KB seja programada remotamente via interface Ethernet. Uma utilidade do programa é prover que dados, imagens gráficas, HTML, Java applets e rotinas pcode sejam carregados na memória EEPROM serial. Os segmentos desta, são então transferidos através da rede utilizando um programa TCP-based (baseado no protocolo TCP). Esta característica também permite a imagens e rotinas pcode serem atualizadas mesmo quando o MSW está ativo.

5.2. Escrevendo as aplicações

Para que se pudesse realizar os trabalhos de implementação lógica, no ambiente de desenvolvimento de software utilizado para o projeto do MSW, foi feito o uso do Assembler livre da Atmel. Para que o Assembler tivesse suas capacidades acentuadas,

[19] Domain Name Server - sistema de fornecimento de endereço numérico na Internet

uma versão Linux do processador GNU[20] C foi utilizada para adição de macros.O processador AT90S8515 permite programação “in-circuit” de sua memória flash via interface SPI.

É feita também, a utilização de código HTML modificado, onde tags especiais, iniciando com um apóstrofo (`), são embutidas no código HTML padrão; tais tags são responsáveis por chamar rotinas de firmware. As tags podem ser utilizadas para inserir dinamicamente variáveis de dados e texto em uma página web, quando esta é referenciada.

A grande questão que envolve toda a implementação do MSW é, compreender como todo o firmware irá se ajustar em um microcontrolador com 8 KB de memória flash. Para tanto, é feito o uso de uma técnica de pseudocódigo (pcode) para conservar o espaço de código em troca de uma velocidade de execução um pouco reduzida.

O interpretador de instruções pcode desenvolvido pela Lightner proporciona simplificação do código do programa e a opção de executá-lo fora da EEPROM (incluindo EEPROM serial externa, e em alguns casos, uma redução no tamanho do código do programa quando comparado ao código nativo).

A simplificação do código é obtida porque o pcode não faz referências aos registros nativos, e porque a “máquina virtual” pcode utiliza tipos de dados “16-bit-wide” para a maioria dos operandos. A execução do pcode a partir da EEPROM é possível porque o ponteiro da instrução pcode é de 16 bits, com a indicação do bit mais alto (quando ativado) que, a próxima instrução pcode seria fetched a partir da EEPROM e não a partir dos 8 KB da memória flash programável do microcontrolador Atmel. A redução do tamanho do programa resulta da combinação de fatores, incluindo aplicações específicas e eficientes de rotinas pcode, e modos de endereçamento do operando pcode flexíveis.

A máquina virtual pcode não tem registros explícitos, com exceção de um “pcode flags register” e um “pcode counter program”, a maioria das instruções pcode referenciam alocações da SRAM no microcontrolador. Os mesmos labels de alocações

[20] Organização/associação sem fins lucrativos que promove o desenvolvimento de softwares de todo o tipo (sistemas operativos, compiladores, etc.) compatíveis com o sistema Unix. O objetivo maior do

de memória utilizados como parte da linguagem de programação assembly AVR normal podem ser utilizados como parte de uma instância de instrução pcode. As instruções pcode podem ter um máximo de três operandos, trabalhando com palavras de 16-bits. A exemplo do microcontrolador AVR, tais palavras são armazenadas na memória em formato “little-endian” (o byte menor é armazenado primeiro na memória).

O ambiente de construção do software é baseado nos sistemas operacionais Windows ou Linux. Apesar de estar-se utilizando o Assembler Atmel padrão para a geração do código de firmware, o código fonte é primeiro passado através de um processador C e de um script Perl antes de ser enviado para o assembler. O procedimento de construção é controlado por um arquivo batch[21] simples.

A construção do arquivo batch executa uma série de passos, o primeiro dos quais é rodar o processador C para gerar um arquivo intermediário (`.i`) a partir de um conjunto de arquivos de entrada de origem (`.asm`). O próximo passo é rodar um programa de extração de string “Perl-based” (baseado no script Perl) que coleta todos os textos de string citados e os armazena em um arquivo separado para inclusão (utilizando `.include`) ao final da memória de programa.

Como terceiro passo, o arquivo batch roda um script Perl que gera o assembler e pós-processa quaisquer mensagens de erro para converter a linha de números associada a um nome de arquivo de entrada original e número de linha. Em seguida, um script Perl que pós-processa o arquivo `.EEP` para separar os seguimentos EEPROM Atmel a partir dos segmentos da EEPROM serial externa é rodado.

O passo final é rodar um script Perl que processa um arquivo texto com uma lista de páginas web e imagens para ser incluído na construção do arquivo batch e produz um arquivo binário de imagens adequado para a utilização com o loader EEPROM serial “network-based” (`netprog.pl`). A programação do microcontrolador Atmel não toma muito tempo, cerca de alguns poucos segundos; a programação da EEPROM serial leva uma quantia similar de tempo, dependendo da quantia de pcode e

---

projeto GNU é o de redistribuir gratuitamente softwares Unix.

[21] Um arquivo texto, com extensão .bat, contendo um ou mais comandos que podem ser executados por um único comando, sendo aqueles (se mais de um) processados da mesma sequência em que escritos, como uma unidade

dados na página web situados na EEPROM serial.

O objetivo do método Pseudocode (pcode) é descrever a funcionalidade de um módulo ou de um procedimento de um modo estruturado, onde a operação lógica é descrita com palavras pré-definidas que são na maioria adaptadas de uma linguagem de programação estruturada ou notações de uma linguagem de programação ou descrições textuais são utilizadas para declarações simples. Como forma de representação, apresenta declarações de controle como:

- **Seqüência :** como uma seqüência de declarações que estão sendo executadas em uma sucessão imediata;
- **Seleção :** como declarações que estão sendo apenas executadas dependendo de certas condições;
- **Iteração :** como uma de várias declarações que estão sendo executadas repetidamente, dependendo de uma condição, podendo ser representada de ambas as formas: como texto ou como gráfico.

Sobre a sua seqüência operacional, o pcode utiliza-se de um procedimento “top-down”, onde o objeto é descrito em um nível relativamente alto via aplicação de estruturas de controle e frases para condições e declarações. Um grande benefício apresentado em sua utilização é, sem dúvidas, relacionada à ausência de limites quanto a seu método de aplicação. Pelas características apresentadas, faz-se necessário o desenvolvimento de um interpretador de instruções pcode para o MSW. O uso do pcode provê:

- Simplificação do código de programação;
- Permitir que um código de programa possa ser executado a partir de outra memória interna de programa, por exemplo, para executar um programa a partir de uma memória externa serial I2C; e
- Em alguns casos, uma redução no tamanho do código de programação quando comparado ao código nativo.

Uma grande vantagem na utilização do pcode é que, os usuários podem adicionar seus próprios opcodes[22] pcode, desde que certas convenções sejam

[22] Código Operacional ou Código de Operação, como também pode ser encontrado em algumas literaturas

obedecidas. O exemplo a seguir mostra o código para uma instrução pcode fornecida pelo usuário chamada `mypcode` que tem três operandos:

```
mypcode:    pcode_routine 3
            ;
            ;      (o código fornecido pelo usuário entra aqui)
            ;
            ret   ; retorna ao interpretador pcode
```

A primeira linha desta rotina exemplo diz ao sistema de desenvolvimento do MSW que `mypcode` é uma nova e legal instrução pcode que requer exatamente três operandos. Percebe-se que o número máximo de operandos pcode permitidos é três.

Uma rotina assembly opcode pcode fornecida pelo usuário é livre para utilizar quaisquer dos registradores processadores, exceto o registrador 16-bit *p* (que é definido como r4 e r5). Se uma rotina desta natureza precisa chamar uma outra rotina (ambas pcode), aquela deve primeiro salvar, e mais tarde restaurar o registrador *p*. Este tipo de rotina não pode depender de quaisquer dos registradores processadores, sendo os mesmos preservados de uma execução para a próxima. Se uma rotina opcode pcode precisa preservar o estado entre chamadas, então a SRAM do microcontrolador deve ser utilizada para salvar este estado.

Uma rotina pcode fornecida pelo usuário pode acessar o pcode flags register da máquina virtual. O formato do pcode flags register é idêntico ao registrador de flags do microcontrolador AVR. Uma rotina pcode fornecida pelo usuário pode copiar o pcode register flags em um registrador de flags AVR com a instrução da linguagem assembly `rcall get_pcode_flags`. O estado atual do registrador de flags AVR pode ser copiado em um registrador de flags pcode com a instrução da linguagem assembly `rcall set_pcode_flags`.

O fonte de instruções pcode é pré-processado pelo ambiente de desenvolvimento do MSW e convertido em uma série de *pseudo-ops* AVR com extensão `.dw`. Cada instrução pcode gera uma única palavra opcode pcode 16-bit seguida de nenhuma ou outras palavras opcode pcode 16-bit. Os 12 bits finais da palavra opcode são o endereço da rotina de execução nativa. O bit superior da palavra opcode é um flag de “endereçamento estendido”. Se este bit está ativado, o endereçamento estendido (responsável pelo controle de interpretação dos operandos) está habilitado, caso

contrário está desabilitado. Quando o endereçamento estendido está desabilitado, todas os operandos são considerados operandos imediatos; quando habilitado, os três bits superiores em cada operando especificam qual modo(s) de endereçamento estendido está sendo aplicado; os 13 bits inferiores do operando são tratados como um endereço. As tabelas 2 e 3 a seguir, ilustram:

Tabela 2 - Formato de Palavra Opcode Pcode

Formato da Palavra Opcode	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Endereçamento estendido habilitado	1	0	0	0	A	A	A	A	A	A	A	A	A	A	A	A
Endereçamento estendido desabilitado	0	0	0	0	A	A	A	A	A	A	A	A	A	A	A	A

Fonte: PicoWeb Pcode

AAAAAAAAAAAA = endereço da rotina pcode na memória EEPROM externa ou on-chip

Tabela 3 - Formato da Palavra Operando Pcode

Formato da Palavra Operando	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Endereçamento estendido habilitado	0	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
	X	W	S	A	A	A	A	A	A	A	A	A	A	A	A	A
Endereçamento estendido desabilitado	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Fonte: PicoWeb Pcode

IIIIIIIIIIIIIIII = valores de dados imediatos
AAAAAAAAAAAA = endereço da rotina pcode na memória EEPROM externa ou on-chip
W = 0: busca a palavra de dados 16-bit no endereço AAAAAAAAAAAA (endereçamento indireto da palavra)
W = 1: busca o byte de dados 8-bit no endereço AAAAAAAAAAAA (endereçamento indireto de byte)
S = 0: nada acontece após a busca de dados
S = 1: troca bytes altos/baixos depois da busca de dados

Todas as instruções pcode têm uma das formas abaixo:

```
opcode
opcode          p1
opcode          p1,p2
opcode          p1,p2,p3
```

onde:

opcode	opcode pcode listado em uma das tabelas de instrução pcode para o MSW
p1-p3	operandos pcode (p1, p2, p3) mostrados nas tabelas de instruções pcode do MSW

Todos os operandos pcode (p1, p2, p3) podem ser expressos em uma de sete diferentes formas, uma **forma normal**, três **formas indiretas** diferentes, e três **formas**

de string literal diferentes, como as seguintes:

`p`	o valor 16-bit *p* é o operando efetivo (forma normal)
`[p]`	a palavra 16-bit armazenada no endereço inicial *p* da SRAM é o operando efetivo
`[byte p]`	byte 8-bit armazenado no endereço *p* da SRAM é o operando efetivo
`[swap p]`	a palavra 16-bit no endereço *p* da SRAM com bytes altos/baixos trocados é o operando efetivo
"..."	o endereço da string ("...") no segmento de código atual (.cseg ou .eseg) é o operando efetivo
`cseg` "..."	o endereço da string ("...") na memória de programa (.cseg) é o operando efetivo
`eseg` "..."	o endereço da string ("...") na memória SEEPROM externa (.eseg) é o operando efetivo

É importante salientar que, se **quaisquer** das formas de operandos indiretas forem utilizadas em uma dada instância de instrução pcode, então o tamanho de quaisquer dos operandos imediatos (*imm* operandos) utilizados naquela instância da instrução pcode é reduzido de 16 bits para 14 bits.Notas relacionadas às tabelas pcode do MSW que seguem:

- `a` indica um endereço SRAM;
- `*a` indica a palavra 16-bit iniciada no endereço *a* da SRAM;
- `a[n]` indica o conteúdo da SRAM no endereço (`a + n`);
- `eeprom[e]` indica o conteúdo da EEPROM on-chip no endereço `e`;
- `seeprom[e]` indica o conteúdo da EEPROM serial externa no endereço `e`;
- `imm` é um valor imediato da palavra 16-bit;
- `ppc` representa o contador do programa pcode;
- `s` representa o endereço inicial de uma string terminada em zero em uma memória flash on-chip ou memória SEEPROM externa;
- `addr` representa o endereço de um opcode pcode (em uma memória flash on-chip ou EEPROM serial externa);
- Os flags pcode (Z, C, N) mostrados na coluna flags estão separados dos flags do processador AVR;
- Z é o flag zero pcode;
- C é o carry flag pcode;
- N é o flag navegador do pcode;
- `low(x)` indica os 8 bits baixos de uma palavra 16-bit valor `x`;
- `swap(a)` indica uma troca de bytes altos e baixos de uma palavra 16-bit `a`.

As tabelas que constam no anexo 6 listam as instruções pcode do MSW predefinidas, agrupadas por função.

5.2.3. CGI

O CGI (Common Gateway Interface) é um tipo de programa ou padrão que permite a servidores web executar aplicativos externos. Trata-se de um conjunto de regras que descreve como um servidor web se comunica com um programa existente na mesma máquina (programa chamado CGI). Possibilita a criação de páginas HTML dinâmicas - muito utilizadas na Internet em páginas interativas, como as que enviam informações de bancos de dados para os usuários, utilizam formulários para consultas, etc. Os programas são geralmente escritos em linguagens de scripting, como a Perl, mas que podem ser escritos em outras linguagens.

Um programa CGI, quando solicitado, manipula dados; tal manipulação ocorre no servidor web. Os programas CGI podem gravar dados no servidor, ou ler dados que estão armazenados no servidor web, e utilizá-los para gerar código HTML. Este código HTML “dinâmico” é transmitido para o navegador que chamou o CGI, e apresentado na forma de uma página HTML.

O acesso ao CGI é proporcionado por um software instalado no servidor web. Geralmente este software de suporte ao CGI prevê que tais programas estejam instalados num determinado diretório, geralmente denominado cgi-bin. O ponto está no fato de que, não existe uma “linguagem” CGI. Teoricamente, qualquer linguagem de programação pode ser utilizada, porém, o programa CGI precisa ser compilado para o sistema do servidor ou, o servidor precisa disponibilizar um interpretador em tempo real que execute o programa. Para a linguagem Perl, adotada para o MSW, o servidor precisa ter um interpretador correspondente instalado. A linguagem Perl será melhor caracterizada nas seções que se seguem.

Uma plataforma CGI é constituída por:

- Um diretório específico no servidor onde os scripts CGI devem ficar. Geralmente este diretório tem o nome de cgi-bin ou cgi-local. Os programas ou scripts CGI só podem ser executados se estiverem neste diretório especificado pelo provedor;

- Alguns provedores permitem scripts CGI no diretório que seja mais conveniente, com o nome que se desejar. Basta dar um nome para o script, terminado em .cgi (por exemplo, MSW.cgi) para acessá-los sem problema;

- Uma porção de dados, os quais são armazenados pelo servidor, que o script CGI possa ler (e, em alguns casos, precisa ler) para que possa processar dados. O servidor web armazena esses dados nas denominadas variáveis de contexto.

Outras plataformas para a execução de programas na web, mais novas, introduzidas por empresas comerciais como a Netscape ou a Microsoft, fazem cada vez mais concorrência à CGI. São exemplos a plataforma API[23] da Netscape e a ISAPI[24] da Microsoft, ambas otimizadas para seus próprios softwares servidores. Desta forma, essas empresas conseguiram um aumento de performance apreciável em relação à CGI. As principais vantagens da CGI, a exemplo do HTML, continuam sendo sua independência comercial e sua padronização universal.

Tanto o CGI quanto o HTML comunicam-se em ambas as direções. Isso significa dizer que, é possível chamar um script CGI através de uma página HTML que esteja ativa, ou que um script CGI pode gerar código HTML e enviá-lo ao navegador, que mostrará a página correspondente.

O CGI também pode armazenar e ler dados num servidor. É assim que funcionam livros de visita ou Bulletin-Boards. Um usuário pode entrar com dados num formulário de página HTML. Ao enviar o formulário, o script CGI é chamado e armazena os dados enviados. Um segundo script CGI ou outra chamada ao mesmo script pode gerar um código HTML contendo as entradas gravadas e enviá-lo para um navegador. Um script CGI pode ser chamado por um arquivo HTML de diversas maneiras:

- Através de um formulário, logo na tag inicial `<form>` (por exemplo, `<form action="/cgi-bin/livrovisitas.pl" method="get">`). A chamada ocorre após o envio do formulário. Os dados digitados ou assinalados pelo usuário ficam à disposição do script para serem processados. É dessa forma que funcionam mecanismos de busca, livros de visita ou compras eletrônicas;

- Através de links. Basta indicar o endereço do script CGI que deve ser executado (por exemplo, `<a href = "/cgi-bin/estatistica.pl"> Estatística </a>`). Esta é uma forma lógica de chamar um script CGI sem enviar parâmetros (dados), como, por exemplo, um contador de acessos;

- Através de uma referência a um gráfico. Também basta indicar o script como

[23] Application Program Interface (Interface para Programa Aplicativo) - conjunto de rotinas ou funções, ou a biblioteca geral de funções e rotinas internas, que o sistema operacional coloca à disposição doa aplicativos que requisitarem seus serviços

[24] Internet Server API - conjunto de chamadas desenvolvidas pela Microsoft para acesso ao seu servidor web para aplicativos de back-end

endereço URL na tag `<img>` do gráfico (por exemplo `<img src="/cgi-bin/contador.pl">`). Neste caso, o script CGI precisa mandar como resposta um gráfico no formato GIF ou JPEG. A maioria dos contadores gráficos baseiam-se neste princípio;

- Através de uma diretiva Server Side Include (SSI) de um arquivo HTML, por exemplo, com o comando `<!-- #exec cgi="/cgi-bin/contador.pl -->`. Isto é muito prático para incluir informação dinâmica em forma de texto num arquivo HTML com a ajuda de um script CGI. Esta forma é adequada, por exemplo, para contadores de acesso baseados em texto;

- Através do carregamento automático do script CGI/programa CGI que deve ser executado. Para isto inclui-se na tag `<meta>` o endereço do script CGI (por exemplo `<meta http-equiv = "refresh" content="0"; URL = /cgi-bin/benvindo.pl">`).

5.2.3.1. Variáveis de Contexto CGI

As variáveis de contexto CGI existem independentemente das variáveis que forem definidas pelo programador num programa CGI. Nas variáveis de contexto CGI estão armazenadas informações como, por exemplo, o nome do servidor web ou o tipo de navegador que chamou o script. Os valores dessas variáveis ficam à disposição do script enquanto ele estiver sendo executado.

Se o script foi chamado através do método GET (melhor especificado nas próximas seções) de um formulário, os dados enviados pelo formulário também ficam numa variável de contexto. A Tabela 12 (Anexo 6) traz o nome de cada uma das variáveis de contexto CGI e suas devidas descrições.

Para poder acessar as variáveis de contexto, o script CGI precisa utilizar as técnicas de leitura de variáveis de contexto proporcionadas pela linguagem de programação utilizada. Em Perl, por exemplo, as variáveis de contexto podem ser alcançadas através de `$ENV{'CONTEXTO_NOMEDAVARIÁVEL'};`. Com um comando como `print $ENV{'SERVER_NAME'};` pode-se, por exemplo, determinar o nome do servidor web onde se encontra o script. Como as variáveis de contexto, sob o ponto de vista Perl, estão armazenadas em Hash[25], pode-se listar todas as variáveis com um comando como `print %ENV`.

[25] Hash - Listas associativas; em criptografia de dados pela Internet, uma função hash é um algoritmo disponível publicamente, aplicado aos dados apropriados, para produzir um valor de integridade estatisticamente único, por sua vez, chamado de valor hash

5.2.3.2. Métodos GET e POST

O Hyper Text Transfer Protocol (Protocolo de Transferência de Hipertexto - HTTP) é o protocolo de comunicação utilizado para a troca de dados entre um navegador e um servidor web. E, para tanto, existem os métodos HTTP. Dois desses métodos, associados à transferência de dados de formulários, é muito importante: o método GET e o método POST.

A importância de se conhecer os métodos reside no fato de que, quando se deseja fazer uso de um script CGI pronto, precisa-se saber por qual dos dois métodos o script espera receber dados. Normalmente isto vem documentado pelo autor do script. Alguns scripts mais inteligentes testam ambos os métodos: neste caso, não importa o método de transferência de dados utilizado, ambos vão funcionar

O GET, um dos métodos do HTTP, é acionado por meio de um formulário HTML através da diretiva `method=get` incluída na tag `<form>`. Por meio desse método, os dados constantes no formulário são primeiramente transmitidos ao software servidor e este, por sua vez, armazena os dados temporariamente numa variável de contexto denominada `QUERY_STRING`. Um script CGI, chamado através da diretiva `action=` incluída na tag inicial do formulário, precisa extrair os dados dessa variável de contexto para poder obter os dados que lhe foram enviados.

Usando Perl, por exemplo, é possível extrair esses dados com `$dados_form = $ENV{'QUERY_STRING'};`. Quando um formulário HTML utiliza o método GET, o fluxo de dados é separado do endereço URL que chama o CGI através de um ponto de interrogação (?). Esta forma de endereçamento e separação pode ser observada no campo de endereços do navegador do usuário, logo após o formulário ter sido enviado.
4.2.1.1. O Método POST

O POST, também um método do HTTP, é acionado por meio de um formulário HTML através da diretiva `method=post` incluída na tag `<form>`. Este método faz com que os dados do formulário sejam diretamente transmitidos ao endereço que constar da diretiva `action=`. Um script CGI, chamado por `action=`, precisa extrair os dados através da entrada padrão, para poder obter os dados transmitidos pelo formulário.

Pode-se, por exemplo, usar Perl e indicar `read (STDIN, $Dados, $ENV{'CONTENT_LENGTH'});`. Observe que o programa precisa obter o valor da variável de contexto `CONTENT_LENGTH` para saber quantos caracteres precisam ser lidos através da entrada padrão. Isto é necessário porque não existe um caracter separador no fluxo de dados.

5.2.4. Perl

Pratical Extraction and Report Language - criada por Larry Wall em 1987, linguagem de programação simples, de macros ou de scripts, mas poderosa e bastante flexível, com algumas características de C++, popular entre usuários do sistema UNIX, utilizada por servidores na Internet; utilizada também em scripts CGI. Sua filosofia é a da praticidade e facilidade de utilização, sendo ao mesmo tempo eficiente.

Uma característica desta linguagem é o fato de ser interpretada e não compilada como C++, ou DELPHI, por exemplo. Por isso a performance de programas Perl é baixa, devendo ser utilizada somente em trabalhos simples.

A Perl é ainda a linguagem de programação mais utilizada para scripts CGI. O motivo é que a Perl possui funções poderosas, por exemplo, para tratar strings ou para a leitura e gravação de dados. O interpretador Perl, necessário para rodar um script em Perl, está à disposição como freeware para quase todos os sistemas operacionais, além de estar instalado na maioria dos servidores da Internet.

A Perl passou a ser mais difundida e respeitada a partir de sua versão 4.0 (março de 1991) e embora seu desenvolvimento ainda seja coordenado por seu criador, este conta com o auxílio de muitos outros programadores por meio da Internet. Atualmente, a Perl tem seu interpretador executado em mais de 20 milhões de web sites por todo o mundo.

6.0. - Aplicações Práticas

Este capítulo apresenta os resultados relacionados com o projeto e a implementação final do sistema objeto desta dissertação.

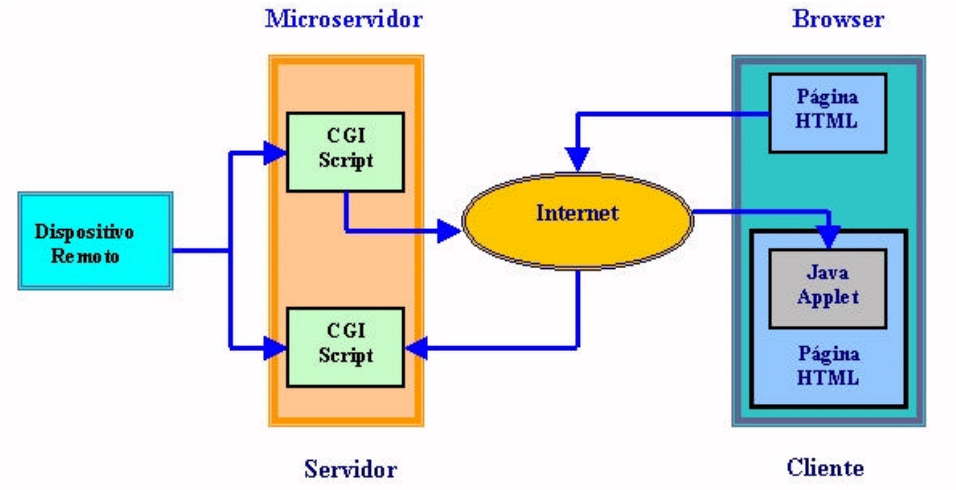
6.1. Descrição Básica das Implementações

Uma avaliação simples do servidor e componentes do cliente que incluem o sistema geral foi descrita na Introdução e nas seções 2 e 3 o servidor MSW dispõe de uma RS232 interface serial pela qual se comunica com outros equipamentos para fins de monitoramento e controle. O MSW também apresenta uma segunda interface, uma ethernet 10Base-T pela qual pode conectar-se à Internet. O cliente simplesmente é qualquer browser que também possa ser conectado à Internet, tipicamente este será um PC com um browser para Internet.

O cliente provê alguns meios para monitorar e controlar os dispositivos, as páginas HTML proporcionam a informação e a navegação básica para estes dispositivos, porém, são as Java Applets embutidas nestas páginas WEB que habilitam a comunicação direta com o servidor e assim o monitorando indireto e controle do dispositivo. Esta interação com o servidor é implementada por scripts CGI executados no servidor WEB.

Uma interação típica entre o cliente e um dispositivo remoto pode ser então dividida em três partes. Para o cliente fazer um pedido primeiramente envia ao equipamento um script CGI apropriado que reside no MSW, Este script CGI em particular pode então enviar o pedido ao equipamento através de sua interface serial. Analogamente para que os dados fluam na direção oposta, um script CGI pode receber dados da interface serial do equipamento e então pode servir estes dados a uma applet Java através de uma conexão com o script CGI ou uma chamada de uma página WEB. Este processo de comunicação é ilustrado na Figura 34.

FIGURA 34 – INTERCOMUNICAÇÃO ENTRE SERVIDOR-CLIENTE



6.2. Estruturando uma Implementação

Uma vez montado o hardware do MSW e definido o software, pode-se passar então aos trabalhos de implementação lógica do mesmo. Para configuração de partes específicas do microcontrolador AT90S8515, como por exemplo, a área ocupada para a inicialização da memória SRAM, os dois blocos da EEPROM (tanto interna como externa ao mesmo), bem como o armazenamento do programa “flash-based” (baseado no programa executado na memória flash), são definidos parâmetros que posteriormente serão inseridos no arquivo de projeto da aplicação, a ser implementado no MSW, denominado “Arquivo de Projeto”.

O Arquivo de Projeto especifica exatamente quais arquivos HTML e/ou de imagens serão carregados no sistema de arquivos da EEPROM serial do MSW, o código fonte do firmware e quais serão os arquivos objetos incluídos como parte do projeto MSW. Tanto o Arquivo de Projeto como seus arquivos de saída resultantes, são processados em múltiplos estágios, incluindo a utilização de um macro processador baseado na linguagem C, um pré-processador pcode, um assembler para as instruções do microcontrolador AVR, um linker de arquivos objeto e finalmente, mas não menos importante, um loader do firmware.

O limite para o tamanho de arquivos de imagens ou código HTML é de 64000 bytes. Arquivos maiores que o considerado limite serão truncados no sistema no momento em que dados e/ou imagens do sistema de arquivos é preparados. As imagens

maiores que 64 Kbytes podem ser mostradas na página web, quebradas em múltiplas partes (frames), sendo estes mostradas como uma única imagem.

6.2.1. Arquivo de Projeto

O arquivo de projeto do MSW traz as informações necessárias para a implementação, como o firmware e as páginas web que contêm a apresentação das aplicações. As definições do pré-processador, existentes no arquivo de projeto, são utilizadas para assinalar alguns parâmetros existentes, na construção de processos do firmware do MSW. A Tabela 4 traz definições consideradas padrão do pré-processador.

Tabela 4 - Definições do Pré-processador MSW

Definição do Pré-processador	Valor Default	Descrição
`BAUD_RATE`	19200	Define a taxa de transmissão da porta serial. É importante observar que nem todas as taxas de transmissão serão possíveis, elas dependem da freqüência de clock do microcontrolador. A diretiva `#define CLOCK` deverá ser corretamente setada
`CLOCK`	7372000	Define a freqüência de clock do microcontrolador Atmel (Hz). É também utilizada para definir a taxa de transmissão da UART e para outros cálculos
`DEBUGGER`	Não definido	Se definido, inclui o firmware debugger da porta serial como parte da construção de processos do projeto
`EEPROM_IP`	Não definido	Se definido, especifica que o endereço IP do MSW está armazenado na EEPROM on-chip (o endereço será proveniente do arquivo texto `ip`)
`ENABLE_WATCHDOG`	Não definido	
`NET_CONFIG_IP`	Não definido	Definido para permitir que o endereço IP do MSW seja modificado através da Ethernet utilizando o programa `set ip`
`NO_ETHER_ADDR_ON_BOOT`	Não definido	Definido para inibir impressão do endereço Ethernet do MSW para a porta serial
`PKT_WATCHDOG_MAX`	250	Pacote de rede do Watchdog Timer limitado pelo slow idle loop (alcance recomendado: 100-254)
`SEEPROM_IP`	Não definido	Definição do endereço EEPROM serial que possui o endereço IP do MSW. Define, se desejado, o endereço IP armazenado em uma memória EEPROM serial
`STATIC_IP_LSB` `STATIC_IP_MSB`	Não definido	Definição de duas constantes de 16-bit, no caso de se desejar que o endereço IP do MSW seja codificado dentro da memória de programa (define MSBs e LSBs do endereço IP 32-bit)
`USE_BOOTP`	Não definido	Utiliza o protocolo BOOTP para obter o endereço IP do MSW

Fonte: Lightner Engineering

As declarações dessas definições do pré-processador são indicadas por "`#define`" e estão localizadas em um arquivo cujo nome é o próprio nome do arquivo de projeto, seguido da extensão "`.h`". A seguir temos um exemplo do mesmo:

```
#define EEPROM_IP             /*  utiliza  arquivo  "ip"  para
                              configuração do endereço IP */
#define NET_CONFIG_IP         /*  permite  reconfiguração  do
                              endereço IP via net */
#define ENABLE_WATCHDOG /* utiliza o WatchDog Timer Atmel */
#define DEBUGGER              /* inclui o firmware debugger*/
#define CLOCK 7372000         /*    pulso    de    clock    do
                              microcontrolador (em HZ) */
#define BAUD_RATE 19200       /*  taxa  de  transmissão  da  porta
                              serial */
```

O arquivo de projeto apresenta uma lista das páginas web que estarão inclusas no projeto. Isso significa que existem arquivos que contêm código HTML e/ou imagens, e serão armazenados na memória SEEPROM do MSW como parte integrante de seu sistema de arquivos. Quaisquer dos arquivos listados devem apresentar uma das extensões abaixo:

`.htm` – código HTML (arquivo texto)
`.html` – código HTML (arquivo texto)
`.txt` – arquivo texto ASCII
`.jpg` – imagem JPEG
`.gif` – imagem GIF
`.png` – imagem PNG
`.js` – Javascript (arquivo texto)
`.cla` – Java applet (byte-codes)
`.class` – Java applet (byte-codes)

As rotinas CGI Públicas para Páginas Web são apresentadas em uma lista, fornecidas pelo usuário e que serão utilizadas pelas páginas web. São listados ainda, os labels dos chamados "ponteiros de entrada pcode" públicos, dedicados às rotinas pcode que são, normalmente, fornecidas nas seções de código seguintes do arquivo de projeto. A extensão "`.cgi`" deve ser anexada a cada label do ponteiro de entrada pcode listado nesta seção.

```
//
 // rotinas CGI de aplicações específicas
 //
 temperatura.cgi   //(retorna a temperatura ASCII (graus C))
```

O exemplo acima indica que uma rotina pcode CGI chamada "`temperature`" existe em algum lugar no firmware do MSW. Pelo fato desta rotina estar listada no

arquivo de projeto, ela pode então ser acessada externamente com uma solicitação HTTP GET utilizando a URL a seguir:

`http://x.y.z.w/temperatura.cgi`

onde *x.y.z.w* é o endereço IP designado para o MSW.

Igualmente, sempre que a string "`temperatura.cgi`" for encontrada em quaisquer dos arquivos de código fonte HTML como parte do projeto, uma tag especial será inserida logicamente nestes arquivos. A mesma fará com que a rotina pcode "`temperature`" seja chamada naquele ponto do código HTML sempre que um arquivo do mesmo tipo contendo aquela tag seja recuperado. É importante perceber que, o processamento da tag aplica-se a qualquer label de subrotina pcode (global), não apenas àquelas rotinas aqui mostradas. [pcode]

Concluída a descrição dos componentes do arquivo de projeto inicia-se o código HTML que é uma importante parte do processo de configuração do MSW. O que diferencia este arquivo de um documento qualquer, é a existência de uma sintaxe não usual apresentando tokens[26] cujas tags são precedidas de apóstrofos (`). É o caso, por exemplo, da tag especial "`temperatura.cgi`", em outras palavras, ela diz ao MSW para chamar a rotina "`temperatura`" exatamente no ponto do código HTML onde ela está localizada; a rotina pcode então, escreve a temperatura corrente (lida como um texto ASCII) na saída. Como resultado, a string aparecerá na página web exatamente naquele ponto, e a temperatura corrente lida será mostrada pelo browser.

6.2.2. Construindo um Projeto

Para que se dê início aos trabalhos de construção do arquivo de projeto de uma aplicação no MSW, dois arquivos texto precisam estar já definidos:

- ip : deve conter o endereço IP designado para o MSW;
- ether : deve conter o endereço Ethernet designado para o MSW, no caso 0.1.2.3.4.5.

[26] Tokens: qualquer caracter enviado que não seja dígito ou ´-´. Desta forma é possível fazer uma convenção, indicando por exemplo, ´O´ para início de arquivo e outras letras para os demais códigos necessários.

Uma vez definidos estes arquivos, o arquivo de projeto pode então, ser compilado para download dentro do microservidor, executado para o nosso exemplo com a linha de comando a seguir:

```
C:>pwbuild exemplo
```

O comando pwbuild é o responsável pela geração de uma série de arquivos, inclusive um arquivo em linguagem assembly (*exemplo.lst*) e um arquivo linker de mapeamento (*exemplo.map*), bem como três arquivos de dados necessários para o download do firmware e páginas web no hardware do MSW:

- exemplo.rom : memória de programa do microcontrolador (arquivo ROM);
- exemplo.ep : programa da EEPROM on-chip do microcontrolador Atmel;
- exemplo.el : "sistema de arquivos" do MicroServidor e dados binários pcode CGI que são carregados na memória SEEPROM externa.

Quadro 1 - Execução do Comando Pwbuild

```
C:>pwbuild teste
Apagando arquivos de saída
Configurando endereço IP
10.7.1.21
Configurando endereço Ethernet
0.1.2.3.4.5
Criando tabelas de comando.
Processando project file teste.pwp
Versão: v1.35
Total de 4813 bytes de código HTML e imagens
Assembling webled.asm
avr-as -W -alms=teste.lst -o teste.o teste.s
Lincando projeto teste.elf
Criando arquivos de inicialização
3695 words na flash (90% used)
12 bytes na EEPROM (2% used)
5686 bytes na SEEPROM (38% used)
Finalizado.
```

O Quadro 2 mostra os resultados obtidos quando da compilação do sistema de desenvolvimento do MSW, utilizando no arquivo de projeto exemplo, a partir da execução do comando pwduild. Se algum erro for encontrado durante o processo de construção, o assembler AVR emitirá linhas de erro com o nome do arquivo fonte

afetado, seguido pelo número de linhas que possui. A seguir um exemplo de uma linha de erro:

```
webled.pwp:143(webled.s:1745) Error: unknow opcode
```

Os arquivos assembler responsáveis por criar módulos objeto e linkeditá-los ao projeto podem ser montados utilizando-se do comando a seguir:

```
gas meuasm
```

Aqui, meuasm especifica que o arquivo “meuasm.asm” seria processado e montado para produzir um módulo objeto chamado “meuasm.o” (e um arquivo em linguagem assembly chamado “meuasm.lst”). Para que o arquivo “meuasm.o” seja linkeditado com o projeto, basta adicionar a linha link add meuasm.o ao arquivo de projeto.

6.2.3. Fazendo o Download do Projeto

Inicialmente um programador de nome AVR.EXE predefinido fará uma pesquisa de todas as possibilidades de portas paralelas do PC para um cabo de programação do microservidor (LPT0, LPT1 e LPT2, por exemplo). O software de programação localiza o cabo buscando por um fio “loop-back” no cabo de programação. Caso não se deseje que esta pesquisa aconteça, pode-se especificar a porta paralela a ser utilizada, adicionando-se o parâmetro “-lptN” à linha de comando (onde N é 0, 1 ou 2), após o nome do arquivo de projeto, nas linhas de comando PWAVRLD e PWLOAD.

O download do firmware do MSW é um processo que está dividido em duas etapas. A primeira etapa é a responsável pelo download de dados no microcontrolador Atmel utilizando um cabo de programação ligado à porta paralela do PC. A segunda, utiliza-se da rede Ethernet para fazer o download das páginas web e do pcode na SEEPROM do microservidor. Passamos a seguir, a uma descrição mais detalhada das duas etapas. Na primeira etapa, os arquivos exemplo.rom e exemplo.ep anteriormente gerados, podem ser carregados no hardware do MSW utilizando-se da porta paralela do PC, via ligação do cabo de programação ao conector no MSW. Pode-se então entrar com a próxima linha de comando:

C:>pwavrld exemplo

A execução desta linha de comando apagará a memória EEPROM do microcontrolador Atmel, e fará o download dos novos firmware e dados no chip. Caso o projeto faça uso de algum pino de I/O da porta B que são compartilhados com o cabo de programação do microservidor, então este deverá ser desconectado após o processo de download estar concluído.

Se até então o MSW estiver conectado à rede, estará agora ativo na rede Ethernet, podendo-se inclusive, efetuar testes com o microservidor a partir do sistema de desenvolvimento do PC (com o programa ping, por exemplo), utilizando-se do endereço IP designado para o MSW. Na segunda e última etapa de compilação do sistema, as páginas web e rotinas pcode CGI armazenadas no arquivo `exemplo.el` criado, precisam ser carregadas no MSW. Essa tarefa pode ser executada a partir da linha de comando a seguir:

C:>pwnetld exemplo

Uma vez executada essa linha torna-se responsável pelo download dos dados destes arquivos na rede, utilizando a placa de rede do PC. Os resultados apresentados durante as etapas de download, são apresentadas no Quadro 4.

Quadro 2 - Download do Projeto

```
C:>pwavrld exemplo
Carregando a Memória de Programa com 'exemplo.rom'
Lendo firmware atual do MSW
Encontrou LPT1 (I/O base 0x378)
Arquivo exemplo.crm da memória de programa gravado (0-3735)
Arquivo exemplo.cee EEPROM gravado (0-511)
3651 alocações modificadas no arquivo exemplo.rom (@0x0001,0xc369->0xc311)
C:\pwdev\bin\avr -ce -lp exemplo.rom
Encontrou LPT1 (I/O base 0x378)
Enviando o comando chip ERASE
Gravando a memória Flash (0-3694).................
Carregando a memória EEPROM com 'exemplo.ep'
C:\pwdev\bin\avr -le exemplo.ep
Encontrou LPT1 (I/O base 0x378)
Gravando memória EEPROM (0-11) ......
Habilitando MSW
Encontrou LPT1 (I/O base 0x378)
Carga do firmware AVR do MSW completa.
C:>pwnetld exemplo
Configurando EEPROM 8515 1FF para FF.
Carregando Pcode CGI na EEPROM via rede
carregando.............................
lendo..............................
Reinicializando EEPROM 8515 1FF para 0.
Carga da rede completa.
```

6.2.4. Visualizando Páginas Web

A função principal do MSW é retornar páginas web e imagens em resposta às solicitações HTTP GET endereçadas pelas URLs, tendo como objetivo seu servidor HTTP. Assim, uma vez compilados todos os arquivos necessários ao funcionamento do microservidor, este pode ser acessado através da rede como um servidor web tradicional, uma vez que as URLs, direcionadas à porta TCP80, são respondidas da maneira convencional.

6.2.5. Debugger

A biblioteca do firmware do MSW apresenta um debugger, simples, que apresenta *dumps* de memória, alterações da memória EEPROM, pcode e controle de traçado da rede, dentre outras coisas. Esse debugger pode ser incluído na construção do MSW, adicionando-se no início do arquivo de projeto a linha "`#define DEBUGGER`". Seus comandos podem ter sua entrada via porta serial ou via rede. Esta última utilizando-se do browser e fornecendo uma URL que referencia uma porta TCP especial: a porta 911. Tais comandos podem ser encontrados na Tabela 12. O formato de um comando debugger URL é o seguinte:

`http://x.y.z.w:911/command[[parameter1]+parameter2]`

Os caracteres *x.y.z.w* representam como já mencionado, o endereço IP previamente designado para o MSW; os parâmetros adicionados (`parameter1` e `parameter 2`) são opcionais, dependendo apenas do comando (`command`) que os precede.

Novos comandos podem ser facilmente adicionados (ou eliminados, para que se possa salvar espaço de programa). Uma lista de comandos debugger ativos é mantida na biblioteca do microservidor, em um arquivo chamado `cmd.txt`. O código fonte do firmware para a maioria destes comandos pode ser encontrado no diretório onde estão as bibliotecas, em arquivos de linguagem assembly (`.asm`).

Tabela 5 - Comandos de Debug do MSW

Comando	**Descrição**
`dm` *XXXX nn*	Dump SRAM no espaço de *XXXX...XXXX+nn-1*
`de` *XXXX nn*	Dump EEPROM on-chip no espaço de *XXXX...XXXX+nn-1*
`ds` *XXXX nn*	Dump EEPROM serial no espaço de *XXXX...XXXX+nn-1*
`wm` *XXXX YY*	Grava o endereço da SRAM *XXXX* com o byte *YY*
Comando	**Descrição**
`we` *XXXX YY*	Grava o endereço da EEPROM on-chip *XXXX* com o byte *YY*
`ws` *XXXX YY*	Grava o endereço da EEPROM serial *XXXX* com o byte *YY*
`L`	Toggle TCP packet logging on/off
`pd` *n*	Set pcode debug trace flag to n (0=off; 1=on)
`pc` *XXXX*	Chama rotina pcode no endereço *XXXX*
`R`	Reinicializa processador (não retorna página web)
Comando	**Descrição**
`^C` (control-C)	Reinicializa processador (apenas porta serial)

Fonte: Lightner Engineering

6.2.6. Linguagem Assembly para MSW

Tabela 6 - Macros Predefinidos Assembly AVR

Macros	**Operandos**	**Descrição**	**Operação**	**Flags**
`Addw`	Wd, Wr	Adiciona palavras sem Carry	Wd ← Wd + Wr	Z, C, N, V, H
`Addwi`	Wd, K	Adiciona imediato à palavra	Wd ← Wd + K	Z, C, N, V
`andwi`	Wd, K	Logical AND Word with Immediate	Wd ← Wd - K	Z, N, V
`clrw`	wd	Limpa Palavra	Wd ← 0	Nenhum
`cmpw`	Wd, Wr	Compara Palavras sem Carry	Wd - Wr	Z, C, N, V, H
`cmpwi`	Wd, K	Compara Palavra com Imediato	Wd - K	Z, C, N, V, H
`decw`	wd	Decrementa Palavra	Wd ← Wd - 1	Z, C, N, V, H
`incw`	wd	Incrementa Palavra	Wd ← Wd + 1	Z, C, N, V, H
`ldsbw`	wr	Carrega Byte a partir da SRAM na Palavra	Wd ← (k)	Nenhum
`ldsw`	Wd, k	Carrega Palavra Direto da SRAM	Wd ← (k+1,k)	Nenhum
`movw`	Wd, Wr	Move Palavras	Wd ← Wr	Nenhum
`Movwi`	Wd, K	Move Immediate to Word	Wd ← K	Nenhum
`Popw`	Wd	Desempilha Palavra	Wd ← STACK	Nenhum
`Pushw`	Wr	Empilha Palavra	STACK ← Wr	Nenhum
`Shlw`	Wd	Troca lógica da palavra à esquerda	Wd ← Wd << 1	Z, C, N, V, H
`Shrw`	Wd	Troca lógica da palavra à direita	Wd ← Wd >> 1	Z, C, N, V, H
`Stsw`	Wd, k	Armazena Palavra Direto na SRAM	(k+1,k) ← Wd	Nenhum
`Subw`	Wd, Wr	Subtrai Palavras sem Carry	Wd ← Wd - Wr	Z, C, N, V, H
`subwi`	Wd, K	Subtrai Imediato da Palavra	Wd ← Wd - K	Z, C, N, V

Fonte: Lightner Engineering

O firmware do microservidor é escrito utilizando-se de um misto de linguagem assembly AVR padrão (nada mais que um tipo de linguagem assembly, interpretada por uma máquina virtual 16-bit). Há ainda a utilização de um número predefinido de macros da linguagem assembly AVR por parte do firmware do MSW. Tais macros são apresentados na Tabela 6, sendo os mesmos utilizados para permitir a manipulação conveniente dos dados.

6.2.6.1. Assembler e Linker

Para o desenvolvimento do MSW fez-se o uso de ferramentas de software GNU adaptadas visando o código de máquinas AVR da Atmel. Tem-se então, uma versão especial do assembler GNU (avr-as) que produz arquivos objeto (.o) e listagens assembly (.lst), a ferramenta de bibliotecas (arquivos) GNU (avr-ar), que gerencia bibliotecas de arquivos objeto (.a), e o linker GNU (avr-ld) que produz arquivos binários ELF (.elf) e mapeadores (.map).

Os arquivos binários ELF são criados pela ferramenta `avr-ld` a partir do módulo objeto, dentro do controle de um script linker. O resultante (o arquivo ELF), é então pós-processado para produzir os arquivos de dados necessários para o download do firmware no MSW. O sistema de desenvolvimento utilizado no microservidor também faz uso do "utilitário make" GNU (`avr-make`), responsável por gerenciar os vários módulos objeto no diretório biblioteca.

O ambiente de desenvolvimento do MSW utiliza-se de várias seções complexas do linker GNU para controlar a localização do código e dados nos vários tipos de memória presentes no hardware do microservidor. Isso inclui a memória SRAM (seções .bas e .data), memória flash de programa (seções .text e .cseg) e EEPROM (seção .section eeprom), todas internas ao microcontrolador Atmel; bem como a memória SEEPROM externa (seções .cseg e .section eeprom).

A nomenclatura e ordem destas muitas seções do linker são bastante delicadas, sendo não recomendadas a adição de novas seções e/ou modificações do chamado "master linker control script" (script de controle do linker mestre), sem que haja um entendimento completo das questões envolvidas.

A Tabela 7 lista as diretivas assembler comuns `.section`, que são utilizadas para controlar a localização de código e dados no MSW:

Tabela 7 - Diretivas Assembler para MSW

Memória de Programa Flash On-Chip (8 Kbytes)	
Diretiva Assembler	*Significado*
.text	***Código de programação AVR e dados***
.section reset	***Seção #avr_reset do MSW***
.section fast	***Seção #avr_fast do MSW***
.section slow	***Seção #avr_slow do MSW***
.section strings	***Strings .ascii e .ascz na memória de programa***
SRAM On-Chip (512 bytes)	
Diretiva Assembler	*Significado*
.data	***Dado inicializado***
.section .bss	***Dado não inicializado (zerado na reinicialização)***
.section COMMON	***Os chamados blocos common***
.section uninitialized_data	***Dados não inicializados (não zerados na reinicialização***
.dseg	***Mapeado para ".data" pelo processador***
EEPROM On-Chip (512 bytes)	
Diretiva Assembler	*Significado*
.section eeprom*	***Dado inicializado na EEPROM***
.section .eeprom*	***Dado inicializado na EEPROM***
EEPROM Serial Externa (16-32 Kbytes)	
Diretiva Assembler	*Significado*
.section seeprom	***Pcode e dados na EEPROM serial***
.section seeprom_string	***Strings na EEPROM serial***
.section seeprom_cgi	***Pcode e dados na EEPROM serial (linkados)***
.section seeprom_cgi_strings	***Strings na EEPROM serial (lincadas)***
.eseg	***Mapeado para ".section seeprom_cgi*" no Project File (erro elsewhere)***

Fonte: Lightner Engineering

6.2.6.2. Compilador GNU C

A versão do Compilador GNU C para os processadores AVR (por exemplo, o avr-gcc) é capaz de produzir módulos objeto que podem ser interligados utilizando a avr-ld, incluindo-os como parte do Arquivo de Projeto do MSW. O código de máquina AVR que o avr-gcc produz, pode utilizar quaisquer dos 32 registradores 8-bits do processador Atmel. Assim sendo, passos especiais devem ser seguidos quando da utilização simultânea do código gcc produzido com a linguagem assembly AVR, ou com pcode. Se uma rotina gcc é chamada de dentro de uma instrução pcode, os ponteiros da instrução pcode (r4 e r5) devem ser preservados. Da mesma forma, qualquer rotina assembly AVR que é chamada a partir de uma rotina C produzida pelo gcc, deve salvar quaisquer registradores que utilizem (não aplicado, entretanto para r0 e r1). Desta forma se uma rotina chamada pelo código C precisa executar o pcode, então todos os registradores do processador (com exceção de r0 e r1), precisarão ser preservados. Isso porque rotinas pcode são livres para utilizar quaisquer dos registradores do processador, exceto os ponteiros da instrução pcode (r4 e r5), como já mencionado.

A barreira maior encontrada na sua utilização e que deve ser observada, é o fato de esta versão do gcc produzir apenas arquivos binários para o código de máquina AVR "nativo". Isso ocorre porque tipicamente, há menos que 2 Kbytes de espaço disponível no MSW para que seja utilizado por códigos de aplicação fornecidos pelo usuário, limitando o gcc. Assim, para que se possa conservar o espaço de código (e SRAM) do microcontrolador Atmel, o uso de variáveis unsigned byte no lugar de int é recomendado.

6.2.6.3. Pré-processador

Tabela 8 - Mapeamentos do Pré-processador

Diretiva Assembler	*Mapeamento do Pré-processador*
`.devide`	*(a linha é apagada)*
`.listmac`	*(a linha é apagada)*
`.dseg`	`.data`
`.cseg`	`.text`
`.eseg`	`.section cgi_seeprom` *(noArquivo de Projeto apenas, de outra forma erro)*
`Eseg string`	`.section seeprom_cgi_string`
`.dw`	`.word`
`.db`	`.byte`
`.que X=y`	`.equ X,y, .equ x,y, .equ X,Y (`*mapas para maiúsculas e minúsculas)*
`.byte` *n**	*Não é modificado*
`.word` *n**	*Não é modificado*

Fonte: Lightner Engineering

Um pré-processador baseado em scripts Perl,é utilizado para preparar instruções pcode para assembly. Este pré-processador automaticamente mapeará diretivas assembler Atmel antes que o fonte da linguagem assembly seja enviado para o assembler GNU. Esses mapeamentos automáticos do pré-processador são listados na Tabela 8.

6.2.6.4. Vetores de Interrupção

Por padrão, cada entrada na Tabela dos Vetores de Interrupção do microcontrolador Atmel, localizada na posição 0 da memória de programa, aponta para uma rotina dummy[27] que consiste de uma única instrução "ret". Isso é feito utilizando a

[27] Um caractere ou outro item de informação que se dá entrada no computador apenas para encontrar condições prescritas, como o comprimento de uma palavra, e que não tem efeito nas operações; uma variável que não contém dados úteis, mas reserva espaço para uma variável verdadeira a ser usada

diretiva global “.weak” do assembler. À seguir é mostrada a Tabela dos Vetores de Interrupção do kernel do MSW na posição 0:

```
.weak VEC_reset
.weak VEC_int0_isr
.weak VEC_int1_isr
.weak VEC_timerlcap_isr
.weak VEC_timerlcompa_isr
.weak VEC_timerlcompb_isr
.weak VEC_timerlovf_isr
.weak VEC_timer0ovf_isr
.weak VEC_spistc_isr
.weak VEC_rx_isr
.weak VEC_udre_isr
.weak VEC_tx_isr
.weak VEC_ana_comp_isr
```

6.2.6.5. Endereços Ethernet para o MSW

O endereço MAC (endereço Ethernet, por exemplo) do microservidor é setado utilizando o arquivo texto “ether” como parte do processo de construção do projeto. Por default, este endereço 6-byte é armazenado ma memória EEPROM do microcontrolador Atmel. Tecnicamente, este endereço seria único, um valor 6-byte diferente para cada dispositivo Ethernet no mundo, na prática, este endereço precisa apenas ser único dentro de uma rede (local) de dispositivos Ethernet, incluindo a(s) primeira(s) rota(s) da rede. Além da primeira rota, o endereço MAC não precisa ser único. O valor preciso não importa, desde que o primeiro byte seja par. As chances de compartilhar o mesmo endereço MAC designado aleatoriamente com outro cartão Ethernet em sua rede são extremamente pequenas. No arquivo de projeto do MSW, o arquivo “ether” seria um endereço decimal no formato 0.48.162.x.y.z, onde x, y, e z são números decimais no espaço de 0 a 255.

6.2.6.6. Endereços IP para o MSW

O endereço IP do MSW é normalmente configurando utilizando o arquivo texto “ip” como parte do processo de construção do projeto. O ideal seria que, para o microservidor, fosse designado um endereço IP na mesma sub-rede do PC utilizado para o desenvolvimento e download do MSW. Por exemplo, se o endereço IP designado para o PC é 10.1.2.3 e a máscara de rede é 255.255.255.0, então qualquer endereço IP no espaço de 10.1.2.1 a 10.1.2.254 está na mesma sub-rede que o mesmo.

6.3. – Instalando uma WebCam no MSW

Nesta seção será demonstrada a flexibilidade do MSW, quanto às aplicações. Neste caso específico, através da conexão serial do mesmo a um dispositivo serial (uma câmera digital), tornando possível a aquisição de imagens, transferindo-as a um browser para que sejam apresentadas.

6.3.1. - A Câmera Digital

Para a implementação dessa aplicação para o MSW, fez-se necessário encontrar uma câmera de baixo consumo, baixo custo, proporcionando um mínimo de qualidade de imagens, além é claro de ser facilmente adaptada ao MSW. Este último quesito torna-se de prioridade superior aos demais, uma vez que estamos trabalhando com um microcontrolador de memória reduzida; desta forma, é imprescindível que a câmera possua uma interface simples e que seja compatível com o microservidor.

A grande maioria das câmeras disponíveis no mercado atualmente possuem porta de comunicação paralela ou USB[28], adequadas para se adicionar vídeo aos PCs. Em um primeiro momento, são perfeitas, preenchendo todos os quesitos assinalados; entretanto, o maior problema apresentado é o fato de que seus protocolos não são simples para se trabalhar com o firmware do microservidor. Isso descarta completamente a sua utilização.

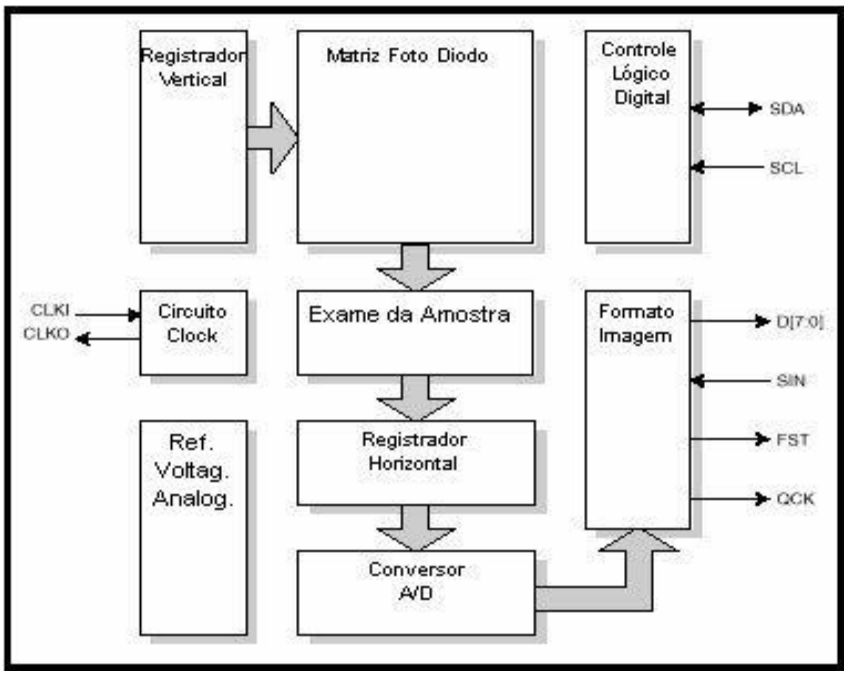
Assim, a opção pela utilização de uma câmera digital mais acessível. As encontradas câmeras com baixa resolução (160 x 120 pixels), com interface serial. A escolhida internamente, é baseada no sensor de saída digital VVL300 da STMicroelectronics. O chip utilizado pela câmera é o VV6301, um sensor de câmeras coloridas altamente integrado. Seu diagrama em blocos é apresentado na Figura 36. Esse tipo utiliza um dispositivo CMOS, melhor do que o sensor CCD (charge-coupled device) comumente utilizado. A vantagem dos sensores CMOS reside no fato de que, um único processo de silício pode ser utilizado para fabricar o chip e todos os seus

[28] USB: Universal Serial Bus, ou barramento serial universal é um padrão desenvolvido em conjunto por empresas de informática e de telecomunicações. Traz as facilidades do Plug and Play para periféricos que

auxiliares lógicos. A câmera, a qual chamamos de WebCam MSW, utiliza um microprocessador MCS 51 da família Intel e memória RAM estática para o armazenamento da imagem.

Uma câmera como a WebCam MSW, envia um total de 20 KB de dados brutos por foto. Quando convertidos em uma imagem JPEG comprimida, esta mesma foto é apenas um décimo deste tamanho. A conexão da câmera ao MSW torna-se uma tarefa relativamente simples, utiliza-se um cabo para a conexão possuindo fios TX, RX e GND. As imagens são armazenadas na câmera em sua RAM. O MSW foi programado para ler a porta serial da webcam, uma vez por segundo com uma taxa de 57.6 Kbps (derivados do clock de 7.372 MHz), necessários para a câmera.

FIGURA 36 - DIAGRAMA EM BLOCOS DO CHIP VV6301



Fonte: VVL300 Digital Output Sensor

6.3.2. - Software para WebCam MSW

O formato do comando para controle da câmera é simples, apresentando caracteres maiúsculos seguido por um parâmetro opcional. Assim, o comando e os

---

estão fora do micro, por exemplo, os aparelhos de som. O USB transfere dados à taxa máxima de 12 Mbits por segundo, o que é suficiente para conectar os dispositivos tradicionais.

caracteres do parâmetro são precedidos por STX (0x02) e seguidos por ETX (0x03). Em resposta a cada seqüência de comandos, a WebCam MSW envia-lhes, ou ACK (0x06) ou NAK (0x15). Se uma resposta a um comando é autorizada, a mesma é precedida por STX (0x02) e terminada por ETX (0x03).**[36]**

As respostas são tipicamente a repetição de uma seqüência de comandos de quatro caracteres, depois há a modificação do caracter do comando para minúsculo e a reposição do parâmetro com um código de status/erro. A Tabela 9 mostra os comandos da câmera utilizados neste projeto.

Tabela 9 - Comandos Utilizados pela WebCam MSW

Comando	Char	Parâmetro	Descrição
Seta imagem index	'A'	Index	Seta a imagem index atual como uma de 6 (seis) imagens
Tira uma foto	'G'	Delay	Tira uma foto e armazena como imagem atual
Carrega uma foto	'U'	0	Envia imagem atual para a porta RS-232

Fonte: VVL300 Digital Output Sensor

São utilizados ainda, comandos responsáveis por configurar uma imagem index antes de tirar uma foto (armazenado-a como primeira), e por carregar a foto. Essa seqüência de comandos é mostrada na Quadro 3.

Quadro 3 - Comandos para Tirar e Carregar Fotos

```
1. Seta a imagem index para zero:
STX + 'A' + 0x00 + ETX espera por ACK seguido de STX + 'a' +
0x00 + ETX

2. Tira uma foto sem o timer delay:
STX + 'G' + 0x00 + ETX então espera por ACK seguido de STX +
'g' + 0x00 + ETX

3. Seta a imagem index para zero novamente:
STX + 'A' + 0x00 + ETX então espera por ACK seguido de STX +
'a' + 0x00 + ETX

4. Carrega a foto:
STX + 'U' + 0x00 + ETX então espera por ACK seguido dos
dados da imagem: STX + 'u' + N 1 +N + N + N + D + D + ... D
+ ETX

N 2 = 2 (número de linhas pretas)
N 3 = 124 (número de linhas visíveis)
N 4 = 16 (número de status de bytes)
D + D + ... D são os bytes de dados da imagem (20,680 bytes)
```

Os dados da imagem adquirida são retornados em um fluxo contínuo, sem qualquer controle, tomando cerca de 4s para transferir os 20.680 bytes completos de dados da imagem. Isso significa que o MSW deve receber, guardar em memória, e transmitir os dados para o socket TCP/IP aberto sem diminuir o taxa de 57.6 kbps. O objetivo maior da WebCam MSW é fazê-la trabalhar como outras câmeras web: acessando-se um web site (preferencialmente a home page do MSW) uma foto tirada pela câmera deve ser apresentada. Uma vez que o MSW está acessível para a Internet, então fotos da câmera podem ser visualizadas de qualquer lugar do mundo.

Para tanto, deve ser armazenada uma página HTML na memória EEPROM serial do microservidor. Este código HTML é retornado quando a URL da home page é referenciada; por sua vez, a página referencia uma Java applet armazenada no MSW, que é enviada ao browser iniciando seu interpretador e a execução do programa Java propriamente dita. O interpretador então, nos apresenta uma janela gráfica no web site na qual, mais tarde, são apresentadas as fotos. Finalmente, a Java applet faz uma conexão TCP/IP retornar ao MSW para que páginas HTML especiais contendo referências de rotinas pcode CGI embutidas sejam recuperadas, fazendo com que rotinas pcode associadas sejam executadas no microservidor.

Em resposta à solicitação da Java applet, o MSW diz à câmera para disparar sua última foto através da sua porta serial (comando upload). O servidor envia este dado bruto de volta para a Java applet através da conexão TCP/IP aberta como uma página web. O dado recebido é processado, tornando-se uma imagem apresentável.

6.3.2.1. - Java Applets

Uma Java applet foi necessária para este projeto porque, os dados da imagem retornada pela WebCam MSW precisam ser processados antes de sua apresentação. Por si só, a câmera não armazena imagens em um formato que possam ser diretamente apresentadas pelo browser (imagens JPEG ou GIF, por exemplo). Em vez disso, a câmera envia dados brutos para o computador na forma de um modelo de cores baseado na matriz de Bayer.

Os pixels são apresentados em uma matriz Bayer 2x2 vermelho-verde-azul-verde como mostrado na Figura 37. Cada pixel no chip sensor de imagens é coberto por um filtro colorido de acordo com o modelo de Bayer mostrado. Há dois pixels verdes para cada pixel vermelho e azul. É necessário fornecer um pixel vermelho, verde e azul para toda possível localização de pixel, a fim de derivar uma imagem real. Se isso não for feito, o resultado é uma imagem esverdeada e de baixa resolução como mostrado. Para que isso não ocorra, os pixels coloridos são observados em toda a sua vizinhança, fazendo-se uma análise sobre a provável cor e intensidade de luz que refletiu em cada pixel no momento em que a foto foi tirada.

FIGURA 37 - MATRIZ DE BAYER

Existe uma série de algoritmos utilizados para interpolação de pixels à disposição, apresentando variações de complexidade e escala de solicitações computacionais. Particularmente a Java applet aqui utilizada, executa um algoritmo de interpolação bilinear simples e rápido, baseado no vizinho-próximo, provendo uma imagem full-color da matriz de pixels brutos. A imagem resultante pode se tornar mais nítida, utilizando-se uma função de convolução antes de ser apresentada.

O principio de funcionamento da convolução é baseado na combinação de um pixel fonte e seus vizinhos para que se possa determinar a cor do pixel que será futuramente apresentado. Essa combinação é especificada através da utilização de um operador linear que determina a proporção da coloração de cada pixel fonte, para que se possa calcular a coloração do pixel de “destino”.

Deve-se imaginar este operador como se fosse um “template” que se sobrepõe à imagem para realizar a convolução em um pixel de cada vez. Como cada pixel sofre os efeitos desta função, o template é movido para o próximo pixel da imagem, e o processo torna a se repetir. Esse é um artificio que os softwares que as empresas fabricantes deste tipo de câmera utilizam, a fim de que fotos fora de foco pareçam melhores**.[35]**

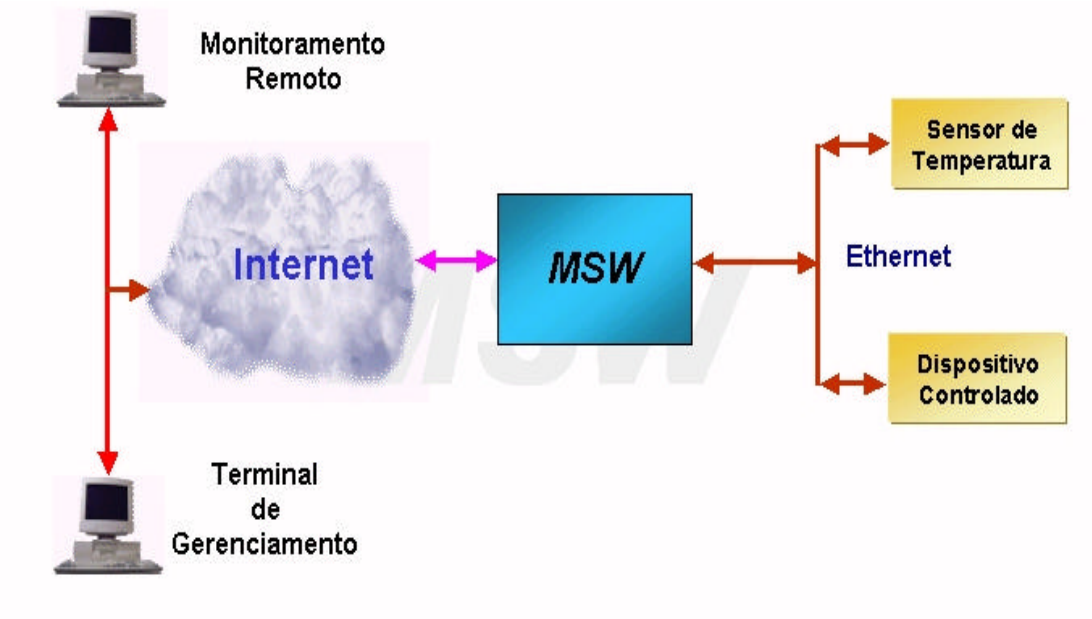
FIGURA 38 – IMAGEM CAPTURADA PALA CÂMERA SERIAL

6.4. – Monitorando e Controlando Temperatura, via Web, com o MSW

Esta aplicação pretende mostrar uma implementação do MSW com vistas a adquirir uma temperatura e controlá-la remotamente via Web. As aplicações desta implementação incluem controles termostáticos, industrial, sistemas, produtos domésticos, termômetros, ou qualquer sistema sensível térmico.

A figura 39 apresenta o modelo implementado para monitoramento e controle de temperatura, para a aquisição foi utilizado o circuito integrado DS 1621, fabricado pela Dallas Semicondutores e que será abordado de forma mais abrangente nesta seção, no que se refere dispositivo controlado, a natureza deste reside no modelo da aplicação, se esta determina que seja necessário refrigerar ou aquecer, a utilização de qualquer uma delas não altera substancialmente o modelo apresentado.

FIGURA 39 - MODELO APLICAÇÃO PARA CONTROLE DE TEMPERATURA



6.4.1. – O modelo da aplicação

A aplicação para aquisição, monitoramento e controle de temperatura está estruturada sob uma página Web construída com código html e cinco scripts CGI, conforme descritos a seguir:

- temperatura.cgi: mostra o valor atual da temperatura;
- temp_t.cgi: determina o valor máximo da escala do termômetro;
- temp_b.cgi: determina o valor mínimo do termômetro;
- set_point.cgi: determina o ponto de ajuste da temperatura;
- setpoint_f.cgi: posiciona o cursor no termômetro.

Para a aquisição da temperatura são utilizadas as portas PD2 e PD7 do microcontrolador e o dispositivo controlado é ativado e desativado através da porta PD4 do AT90S8515, através do conector DB25. A temperatura captada pelo chip sensor de temperatura Dallas DS1621, é apresentada em uma página web semelhante a aqui apresentada na Figura 40, tendo um mecanismo de refresh a cada 60 s, o que possibilita retorno da temperatura ambiente real, online, aos usuários.

FIGURA 40 - PÁGINA WEB DISPARADA PARA APRESENTAÇÃO DA TEMPERATURA

6.4.2. O sensor de temperatura

O sensor de temperatura, mostrado na figura 41, escolhido foi o DS1621, da Dallas Semiconducors que apresenta as características básicas a seguir:

- Mede temperaturas de –55°C a +125°C com incrementos de 0.5°C;
- A temperatura é lida como um valor de 9–bits;
- Converte a temperatura em uma palavra digital em 1 segundo;
- Os dados são lidos e escritos através de um barramento I2C.

FIGURA 41 – SENSOR DE TEMPERATURA DS 1621

Fonte: Dallas DS1621 – Data Sheet

A descrição dos pinos do sensor de temperatura é mostrada na Tabela 10 e a conexão destes ao MSW é mostrada na Figura 42. **[34][35]**

Tabela 10 – Descrição dos Sinais do DS1621

Nome do Sinal	Descrição
SDA	entrada e saída de dados serial
SCL	sinal de clock
GND	Ground
T OUT	sinal de saída de referência/termostato
A0	entrada de endereçamento do chip
A1	entrada de endereçamento do chip
A2	entrada de endereçamento do chip
VDD	tensão de alimentação da fonte

7.0. - Conclusões e Perspectivas Futuras

7.1. Trabalhos Futuros

Este capítulo discute possíveis caminhos de futuros desenvolvimentos que podem ser feitos a partir do sistema objeto desta dissertação. Em particular, de discutir a necessidade de agregar e suportar outros tipos de dispositivos, as possibilidades e requirementos para conexão à Internet através de outros meios, diferentes da Ethernet, e o assunto acerca da segurança. Finalmente, este capítulo discute as possibilidades em longo prazo de futuros desenvolvimentos que este sistema poderá oferecer. Isto inclui a capcidade para atualização do firmware remotamente e muito importante à viabilidade comercial.

7.1.1. Suporte para Outros Dispositivos e Vendedores

Uma limitação do sistema implementado nesta dissertação é que o dispositivo é somente possível quando o MSW é conectado para um dispositivo específico. Isto se dá porque o software cliente do msw foi projetado para as especificações da interface serial.

Uma das primeiras implementações futuras para o sistema deverá ser prover suporte para a conexão com diferentes modelos de diferentes fornecedores, através de uma interface USB. Consequentemente, o sistema atual será provido de interface largamente utilizada, permitindo, então, o suporte a diversos dispositivos existentes no mercado.

8.1.2. Diferentes Meios de Conexão na Internet

A presente implementação possibilita o uso do dispositivo na web através de uma conexão Ethernet 10Base-T, limitando o uso do dispositivo remoto dentro de uma LAN, não podendo fazer assim uso de suas características na ausência de tal infra-estrutura. Isto traz um problema, ou seja, prover a possibilidade de conexão do dispositivo remoto em um local onde não está disponível a conexão através de uma LAN. Este é obviamente o caso com da maioria das residencias, então, uma das metas iniciais do sistema de prover acesso ao dispositivo remoto para residências torna-se dificultada.

Isto sugere então, que seria apropriado para um desenvolvimento adicional do sistema à possibilidade de acesso via Web através de um meio diferente da LAN, como conexões baseadas em modem, interface wireless, ou até mesmo comunicação baseada em satélite.

A conexão mais óbvia seria a baseada em modem, pois este é um dos meios primários para acesso à Internet e disponível na maioria das residencias. Nesta implementação, o servidor Web embutido no MSW disporia de uma conexão discada para um Provedor de Serviços de Internet (ISP) para poder acessar a Internet. Naturalmente, alguns problemas de acesso à rede deveriam ser superados para a efetiva utilização do sistema.

O MSW não dispõe de um meio direto para implementar uma conexão baseada em modem; então, a alternativa seria trabalhar a partir da porta serial disponível para atingir este objetivo. Uma possível solução seria incorporar a PPP na pilha TCP/IP e também ajustar a interface serial para conexão com o modem. Enquanto conexões discadas são o principal meio para conexões de Internet na maioria das regiões, em alguns locais, principalmente nas áreas rurais somente a comunicação por satélite é possível. Portanto, para utilização do dispositivo remoto nestes locais, desenvolvimentos adicionais precisariam ser feitos.

As conexões via satélite são grandes, ou seja, demanda muita memória, em comparação ao tamanho do sistema implementado, argumento que torna uma conexão

para Internet via satélite, para os dispositivos remotos difícil. Contra argumentando, poderia ser proposto um sistema, envolvendo um veículo que agiria como uma estação base e proveria acesso à Internet através de uma conexão via satélite montada nele. Tendo vários dispositivos remotos conectados wireless com a estação base no veículo (algumas placas de LAN sem fio: IEEE 802.11a, b poderiam ser implementadas), o veículo poderia executar a coleta de dados dos dispositivos remotos em cidades rurais. Cada unidade se comunicaria com a estação base dentro do mesmo método como o sistema presente, e a estação base proveria a ligação com o satélite para a Internet na qual profissionais em outras cidades ou países poderiam monitorar e controlar dados.

7.1.3. Segurança

A informação médica, por exemplo, é um assunto sensível à maioria das pessoas, e as implicações de prover informação on-line podem ser bastante sérias. Na implementação atual do sistema, o cliente e o servidor estão situados dentro de uma LAN que é protegida do mundo externo através de firewalls. Portanto, a transferencia dos dados de um paciente não é um problema tão grande uma vez que as pessoas sem autorização não estão dentro da infra-estrutura da LAN.

Porém, problemas poderão surgir se o sistema for ampliado de forma que o cliente e o servidor possam funcionar em diversas redes, inclusive com conexões de dialup. Em tal situação, a capacidade para assegurar a transferência e acesso de dados de um paciente e/ou acionar um dispositivo, é essencial. Para assegurar estas transações e autorizações, esquemas de criptografia serão necessários. Por exemplo, seria necessário o servidor Web prover alguma forma de autenticação, ou contra-senha, para permitir que somente as pessoas autorizadas tenham acesso a informação ou poder para controlar os dispositivos.

Então, assegurar que a transferência de dados pela Internet esteja segura, técnicas de criptografia precisariam ser empregadas. Felizmente, existem muitas disponíveis atualmente, quer comercial, quer de domínio público, especificamente para a Internet, como os desenvolvidos pela RSA **[20].**

7.1.4. Atualização do Firmware Remotamente

A habilidade para ter acesso a um dispositivo remotamente pela Internet introduz muitos questionamentos foram discutidas ao longo desta dissertação. Porém, é importante lembrar que o acesso à Internet não só é limitado ao dispositivo, mas também se estende ao servidor Web embutido que provê a conectividade. É possível implementar atualizações do firmware no servidor Web embutido, remotamente pela Internet.

Uma possível vantagem de tal implementação origina-se na discussão de prover suporte para múltiplos modelos ou tipos de instrumentos. Em vez de dispor de todas as rotinas para conectar muitos dispositivos diferentes, o servidor Web poderia atualizar as rotinas remotamente pela Internet.

Este procedimento poderia ser feito manualmente a partir de uma opção dentro do software do cliente onde o instrumento particular e modelo são escolhidos de uma lista e a atualização apropriada é carregada então no servidor Web. Semelhantemente, o procedimento também poderia ser feito automaticamente pelo próprio MSW que poderia carregar atualizações de um servidor central. Esta técnica seria particularmente útil para assegurar atualizações de firmware que poderiam alcançar todas as unidades.

7.1.5. Viabilidade Comercial

Há vários pontos neste documento que permitem discutir o caso que uma versão futura do sistema, seria comercialmente viável. Primeiramente, foi discutida implementação e o custo de implantação, plenamente viável e posteriormente foram mostradas aplicações práticas que estimulam a continuidade do projeto e sua exploração comercial.

7.2. Conclusão

É certo que, uma área tão abrangente e complexa como a Ciência da Computação pode ser em sua íntegra representada pelo provérbio que diz “o homem é o lobo do homem”; uma vez que seu avanço é contínuo, e o que é novo hoje, pode num

piscar de olhos, estar fadado ao esquecimento amanhã.

Baseado nisso, é que os estudos dedicados à construção e implementação do MSW visaram sempre a busca de inovações no que diz respeito, especialmente ao hardware dos computadores.

Buscou-se sempre o desenvolvimento de um dispositivo que, apresentando características próprias, com algumas características distintas do que se vê atualmente, realizasse os mesmos trabalhos, com mesma - ou melhor- eficiência, e nem por isso, custasse mais caro ao usuário final.

A utilização de um microcontrolador como coração de um servidor web ao invés de um microprocessador denota uma das características primeiras do projeto. Sem dúvidas tornou-se claro que, a mais marcante vantagem dentre as existentes é o fato de o primeiro reunir em um único chip vários elementos (memórias, timers, contadores, portas de I/O, entre outros), ao passo que em sistemas microprocessados, os mesmos são desempenhados por chips independentes.

Este fato evidencia como sendo o uso do microcontrolador a proposta mais atraente, provando que lhe cabe o pseudônimo “microcomputador de um só chip”. Assim, quando o mesmo foi conectado a outros elementos de hardware e um pequeno código de programação, concretizou-se um servidor de tamanho extremamente reduzido, cuja funcionalidade inicial é voltada ao conveniente provimento de acesso a web, sem que se faça necessária a utilização de um PC.

Todavia, sua funcionalidade não está restrita somente a isso, estando o MSW apto a ser utilizado nas mais variadas aplicações. Dentre estas, duas mereceram atenção especial , e foram adotadas para este projeto.

A primeira delas foi a utilização de um chip sensor de temperatura. Demonstrou-se que, por intermédio deste chip, é possível ao MSW captar a temperatura de qualquer ambiente, disponibilizá-la online e, como conseqüência disto acionar ou chavear qualquer dispositivo que dependa desta informação, através da comunicação serial entre o MSW e o mencionado dispositivo.

A outra aplicação, envolve a utilização de uma câmera web responsável por captar imagens e, por intermédio da interface com o MSW, apresentá-las via Internet para qualquer lugar do mundo. O desafio maior, além de se conseguir uma câmera serial, foi sem dúvidas aumentar a performance da WebCam quando da aquisição de imagens de maneira contínua, sem a utilização do mouse.

Buscou-se da maneira mais adequada possível modificar seu software - uma Java applet - para que se conseguisse tal feito de forma viável, sem prejudicar seu perfeito funcionamento.

Em suma, a construção do MSW, vem provar não só a viabilidade de sua arquitetura embasada com a tecnologia dos microcontroladores, mas a possibilidade de produzir dispositivos flexíveis, de fácil adaptação às necessidades que se apresentam, de grande qualidade, proporcionando consequentemente maximização dos benefícios a baixos custos.

Referências Bibliográficas

1. NETBURNER. "**Netburner Standard Hardware Plataforms**". Janeiro 2000. Disponível em <http://netburner.com/hardware_platforms.html>. Acesso em 10 Fev. 2001.
2. NETBURNER. "**Pricing Information**". Janeiro 2000. Disponível em <http://netburner.com/pricing.html>. Acesso em 10 Fev. 2001.
3. LIGHTNER Engineering, "**PicoWeb** Home Page . Janeiro 2000. Disponível em <http://www.picoweb.net/toc.html>. Acesso em 10 jan. 2001.
4. CONNECTONE. "**Connect One Home Page**", Fevereiro 2001. Disponível em <http://www.connectone.com/>. Acesso em 02 fe. 2002.
5. RABBIT Semiconductor. **"Rabbit2000 TCP/IP Development Kit**". Janeiro 2001. Disponíivel em <http://rabbitsemiconductor.com/products/rab20_tcpip/rab20_tcpip_devkit.html >. Acesso em 3 de fev. 2002.
6. HEWLETT Packard. "**HP Brings the Power of the Internet to Printing**". Dezembro 2000. Disponível em <http://www.hp.com/hpinfo/newsroom/press/20mar01a.html>. Acesso em 20 de mar. 2001.
7. FREYER Steve. "**A $25 Web Server**". Circuit Cellar Online. julho de 1999. Disponível em <http://www.chipcenter.com/circuitcellar/july99/c79b11.html>. Acesso em 21 de dez. 2001.
8. LIGHTNER Bruce. "**Look Ma, No PC! – A $55 Webcam.** Circuit Cellar Online. nº 121. Agosto de 2000.Disponível em http://www.chipcenter.com/circuitcellar/august00/c0800bl.html. Acesso em 14 de mar. 2002.

9. ATMEL**.** "**AVR Microcontroller with 8K Bytes In-System Programmable Flash (AT90S8515)**", Atmel Corporation. Setembro de 2001. Disponível em <http://www.atmel.com/atmel/acrobat/doc0841.pdf>. Acesso em 30 de set. 2001.
10. INTEL**. 8-bit Embedded Controller Handbook**. Intel Corporation, Santa Clara, 1989.
11. SCHMIDT, G. **"Beginner´s introduction to AVR assembly language".** Outubro 2001. Disponível em <http://www.dg4fac.de/avr/avr_em/>. Acesso 8 de nov. 2001.
12. REALTEK**. RTL8019AS - Realtek Full-Duplex Ethernet Controller with Plug and Play Function (RealPNP**). REALTEK SEMI-CONDUCTOR CO., LTD, TAIWAN, R.O.C. 2001.
13. BARNETT, Richard H. **The 8051 Family of Microcontrollers**. 1ª ed. São Paulo. Prentice Hall. 2000.
14. BÔAS, Alexey Antônio Villas; SANTOS, Leandro Farina dos; [et al]. **CGI com Perl – Conceitos, Programação e Aplicação**. 1ª ed. São Paulo: Érica, 2001.
15. DAMASCENO JÚNIOR, Américo. **Aprendendo Perl com Arquivos Txt**.Julho 2000. Disponível em <http://www.foster.com.br>. Acesso em 14 set. 2001.
16. PROVIDELO, Celso Renato G. **Desenvolvimento do AVRworkstation**. http://july,sel.eesc.usp.br/documents/uisp, 20 de setembro de 2000.
17. PROVIDELO, Celso Renato G. **Estudo Básico da Arquitetura AVR**. Setembro 2000. disponível em <http://july,sel.eesc.usp.br/documents/uisp>. Acesso em 20 de nov. de 2000.
18. QUONG, Russel. **Perl in 20 pages. A guide to Perl 5 for C/C++, awk, and shell programmers**. Fevereiro 2001. Disponível em <http://best.com/~quong/perlin20>. Acesso em 19 de fev. de 2001.
19. SILVA JUNIOR, Vidal Pereira da. **Aplicações Práticas do Microcontrolador 8051**. 8ª Edição. São Paulo; Érica, 1999. 270p.
20. STANISLAV, G. Adam. **Tutorial de CGI**. Fevereiro de 1999. Disponível em <http://www.aesa.com.br/informatica/cgi>. Acesso em 29 de fevereiro de 2000.
21. PROVIDELO, Francisco. **Crystal Board** .Fevereiro 2000. Disponível em http://july.sel.eesc.sc.usp.br>. maio 2000. Acesso em 12 de mar. 2000.

22. MAXIM. +5-Powered, Multichanel RS232 Drivers/Receivers. Novembro de 1997. Disponível em <http://www.maxim-ic.com/>. Acesso em 21 de nov. de 2000.
23. NATIONAL instruments. DP83905 Data Sheet. Janeiro 2000. Disponível em <http://ni.com/> Acesso em 22 de novembro 2001.
24. PLATISE, Uros. **Assembler Ava**. Abril 2001.Disponível em <http://medo.fov.uni-mb.si/mapp/uTools/index.html>. Acesso em 10 de dez. 2001.
25. DOMOINFO.**Introducción a Domótica-Que es**. Fevereiro de 2000. Disponível em < http://www.domotica.net/2-1-1.htm>. 12 de maio de 2002.
26. The I 2 C-bus and how to use it, Philips Semiconductors, 1995 update April 1995
27. BOTHHAND. **LF1S022 – Single RJ-45 Connector Module with Integrated 10 base T magnetics & Filters**. Abril 2000. Disponível em <www.bothhand.com/products>. Acesso em jun. 2002.
28. HUDSO, Dave. **Liquorice Project.** Outubro 2000. Disponível em <http://liquorice.sourceforge.net/>. Acesso em 15 nov. 2000.
29. KYLE Jason. **E-AVR Project**.Janeiro 2002. Disponível em <http://avr.jpk.co.nz/>. Acesso em 27 de mar. 2002.
30. PLATISE Uros. **Ultimate Realtime Operating System I**. Janeiro 2002. Disponível em <http://medo.fov.uni-mb.si/mapp/uros/>. Acesso 31 de Jan. 2002.
31. CIRRUS. **CS8900A Data Sheet**.Abril de 2001. Disponível em <http://www.cirrus.com/design/products/overview>. Acesso em 12 Abril 2001.
32. CIRRUS .**AN181 – Using The Crystal 8900A in 8-bit Model**. Janeiro 2000. Disponível em <http://www.cirrus.com/design/products/overview>. Acesso em 12 Abril 2001
33. DALLAS Semiconductor**. DS1621 – Digital Thermometer and Thermostat**. Outubro 1999. Disponível em http://dbserv.maxim-ic.com/. Acesso em 12 julho 2000
34. DALLAS Semiconductor. **App Note 176: Using the DS1631 in DS1621 Applications**. Outubro 1999. Disponível em <http://dbserv.maxim-ic.com/>. Acesso em 12 julho 2000

35. WELCH David. **NickCam Connecting**. Junho 2001. Disponível em <http://www.dwelch.com/nickcam>. Acesso em 06 julho 2002
36. SOURCEFORGE. **EVK LoRes Serial Communications Protocol**. Janeiro 2002. Disponível em <http://webcam.sourceforge.net/>. Acesso em 30 Janeiro 2002.
37. SCHUNK, Leonardo Maurício. **Microcontroladores AVR – Teoria e Aplicações Práticas**. 1ª Edição. São Paulo. Editora Érica. 2001.
38. STEWART, James W. The 8051 Microcontroller – Hardware, Software and Interfacing. 2ª Edição. New Jersey – USA. Prentice Hall. 1999.
39. PREDKO, Mike. Programing and Customizing the 8051 microcontroller. 1ª Edição. New York – USA. MCGraw Hill. 1999.
40. ST Microeletronics. L7800 Series Data Sheet. Julho 2000. Disponível em <http://www.st.com>. Acesso em 30 de maio 2002.

Anexos

Anexo 1 - Diagrama Elétrico do MSW

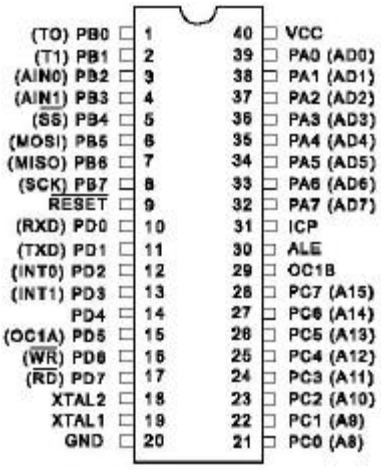
Anexo 2 – Registradores do AT90S8515

Tabela 2 – Registradores de I/O do AVR AT90S8515

Endereço	**Nome**	Função
0x3f	SREG	Registrador de Status
0x3e	SPH	Ponteiro de Pilha alto
0x3d	SPL	Ponteiro de Pilha baixo
0x3b	GIMSK	Máscara de Interrupção geral
0x3a	GIFR	Flags de Interrupção
0x39	TIMSK	Máscara de Interrupções do Timer
0x38	GIFR	Flags do Timer
Endereço	**Nome**	**Função**
0x35	MCUCR	Controle geral do MCU
0x33	TCCR0	Controles do Timer 0
0x32	TCNT0	Timer 0
0x2f	TCCR1A	Controles do Timer 1
0x2e	TCCR1B	Controles do Timer 1
0x2d	TCNT1H	Timer 1 alto
0x2c	TCNT1L	Timer 1 baixo
0x2b	OCR1AH	Timer 1 Output Compare A alto
0x2a	OCR1AL	Timer 1 Output Compare A baixo
0x29	OCR1BH	Timer 1 Output Compare B alto
0x28	OCR1BL	Timer 1 Output Compare B baixo
0x25	ICR1H	Timer 1 Input Capture alto
0x24	ICR1L	Timer 1 Input Capture baixo
0x21	WDTCR	WatchDog Controls
0x1f	EEARH	Endereço EEPROM alto
0x1e	EEARL	Endereço EEPROM baixo
0x1d	EEDR	Dado EEPROM
0x1c	EECR	Controles EEPROM
0x1b	PORTA	Latch de saída
0x1a	DDRA	Direção do Dado
0x19	PINA	Leitura
0x18	PORTB	Latch de saída
0x17	DDRB	Direção do Dado
0x16	PINB	Leitura
0x15	PORTC	Latch de saída
0x14	DDRC	Direção do Dado
0x13	PINC	Leitura
0x12	PORTD	Latch de saída
0x11	DDRD	Direção do Dado
0x10	PIND	Leitura
0x0f	SPDR	Dado SPI
0x0e	SPSR	Status SPI
0x0d	SPCR	Controles SPI
0x0c	UDR	Dado UART
0x0b	USR	Status UART
0x0a	UCR	Controles da UART
0x09	UBRR	Baud Rate UART
0x08	ACSR	Status Comparador Analógico

Fonte: Lightner Engineering

Anexo 3 – Descrição dos Pinos do AT90S8515



VCC : Alimentação;
GND : Terra;
Porta A (PA7..PA0) : A Porta A é uma porta 8-bit bidirecional de I/O. Seus pinos apresentam resistores de pull-up[29] (selecionados para cada bit); pode suportar até 20 mA de corrente, o suficiente para acionar diretamente displays de LEDs. Quando os pinos PA0 a PA7 são utilizados como entradas e são pulled low[30] (nível baixo é forçado) externamente, só se tornarão fonte de corrente se os resistores de pull-up estiverem ativos. A Porta A serve como uma entrada/saída Multiplexada à parte menos significativa de Dados e Endereços quando utilizando a SRAM externa. Serve ainda como saída dos bytes de instrução, durante a verificação dos programas, sendo necessária a colocação de pull ups externos para esta operação;
Porta B (PB7..PB0) : A Porta B é uma porta 8-bit bidirecional de I/O, com resistores de pull-up. A exemplo da Porta A, seus buffers podem suportar até 20 mA de corrente. Como entradas, os pinos da Porta B são pulled low externamente e originarão corrente se os resistores de pull-up estiverem ativos. Durante a verificação de programas, serve como entrada da parte menos significativa de endereços. Esta porta também apresenta funções próprias do AT90S8515;
Porta C (PC7..PC0) : A Porta C é uma porta 8-bit bidirecional de I/O, com resistores de pull-up. Seus buffers podem suportar até 20 mA de corrente. Como entradas, os pinos da Porta C são pulled low externamente e originarão corrente se os resistores de pull-up estiverem ativos. Serve também como saída da parte mais significativa de Endereços quando utilizando a SRAM externa, ou verificação da programação;
Porta D (PD7..PD0) : A Porta D é uma porta 8-bit bidirecional de I/O, com resistores de pull-up. Seus buffers podem suportar até 20 mA de corrente. Como entradas, os pinos da Porta D são pulled low externamente e originarão corrente se os resistores de pull-up estiverem ativos. A exemplo da Porta B, também esta apresenta funções específicas do AT90S8515;
RESET[31] : Entrada de Reset. Um baixo nível sobre este pino por mais de 50 ns irá gerar um reset, até mesmo se o clock não estiver trabalhando. Pulsos menores não são garantia para geração de reset;
XTAL1 : Entrada para o amplificador inversor do oscilador e entrada para o circuito operacional interno de clock;
XTAL2 : Saída do amplificador inversor do oscilador;
ICP : Pino de entrada para a função Timer/Counter1 Input Capture;
OC1B : Pino de saída para a função Timer/Counter1 Output CompareB;

[29] Um resistor de pull-up representa um processo pelo qual é garantido que um certo ponto num circuito lógico ficará num nível lógico fixo, não flutuando aleatóriamente. Neste caso, é garantido o nível 1. (SILVA JUNIOR, 1999)
[30] Exatamente como pull up, apenas aqui o nível é 0. (SILVA JUNIOR, 1999)
[31] Os pinos sucedidos por "\", indicam complemento, ou seja, que o mesmo é ativo em nível lógico 0

ALE : (Address Latch Enable) – Saída habilitadora do latch[32] de endereços, é utilizado quando a Memória Externa está habilitada. O strobe ALE é utilizado para manter o endereço baixo (8 bits) em um endereço de armazenamento durante o primeiro ciclo de acesso, e os pinos AD0-7 são utilizados pelos dados durante o segundo ciclo de acesso.

Os pinos que exercem funções especiais são descritos na seqüência:

RXD : Receptor da porta serial assíncrona ou entrada e saída de dados síncronos;
TXD : Saída de transmissão da porta serial assíncrona, ou saída de clock para os registradores de deslocamento;
INT0 : Interrupção externa número 0, ou Bit de controle para o Timer/Counter0;
INT1 : Interrupção externa número 1, ou Bit de controle para o Timer/Counter1;
T0 : Entrada externa para o Timer/Counter0;
T1 : Entrada externa para o Timer/Counter1;
WR : Strobe de escrita na memória SRAM externa;
RD : Strobe de leitura na memória SRAM externa.

As descrições de cada um dos pinos foram baseadas no Data Sheet AT90S8515 da Atmel, procurando tornar a leitura deste documento mais acessível foi efetuada a tradução do Data Sheet do Atmel AT90S8515 (ANEXO 3).

[32] Funciona como uma porta: estando ativa, deixa passar a informação presente em suas entradas, e se inativa, faz com que a informação na saída não se altere, independente das alterações na entrada. (SILVA JUNIOR, 1999)

Anexo 4 – Memória EEPROM Serial

Descrição dos Pinos

A descrição dos pinos é listada na Tabela 2-1.

Tabela 3: Função dos Pinos

Name	PDIP	SOIC	Function
A0	1	1	User Configurable Chip Select
A1	2	2	User Configurable Chip Select
A2	3	3	Non-Configurable Chip Select This pin must be hard wired to logical 1 state (Vcc). Device will not operate with this pin left or held to to logical 0 (Vss).
Vss	4	4	Ground
DAS	5	5	Serial Data
SCL	6	6	Serial Clock
WP	7	7	Write Protect Input
Vcc	8	8	1.8 to 5.5V (24AA515) 2.5 to 5.5V (24LC515) 4.5 to 5.5V (24FC515)

Fonte: Microchip 24xx515 Data Sheet

Entradas de endereço A0 e A1

As entradas A0 e A1 são usadas para múltiplas operações do 24XX515. São comaparados os níveis nestas estradas para determinar o dispositivo endereçado. O chip é selecionado se o compare for verdadeiro. Até quatro dispositivos podem ser conectados no mesmo barramento usando diferentes combinações das linhas de endereço. .

Entrada de endereçamento A2

A entrada A2 não é configurada pelo chip select. Estes pino pode ser conectado ao VCC para tornar o dispositivo operacional.

Serial Data (SDA)

Este é um pino de comunicação bi-directional pin usado para transferir endereçamento e dados para um outro dispositivo. Ele é um terminal do tipo dreno-aberto, então, o barramento SDA requere um resistor de pull-up para VCC (típico 10 k**W** para 100kHz, 2 k**W** para 400kHz e 1MHz). Para transferência de dados normal o pino SDA é permite a mudança somente durante o período em que SCL estiver em nível

baixo. As mudanças durante o períod o em que SCL estiver alto são reservadas para indicar as condições de START e STOP.

Serial Clock (SCL)

Esta entrada é usada para sincronizar a transferência de dados com o dispositivo.

Write Protect (WP)

Este pino pode ser conectado em VSS, VCC ou ficar flutuando. Um resistor interno de pull-down neste pino manterá o nível se este ficar em estado flutuante. Se o pino for conectado ao VSS ou ficar flutuante, a operação normal de memória estará habilitada (leitura/escrita nos endereços de memória 0000h-FFFFh). Se for conectado em VCC, a operação de escrita estará habilitada. As operações de leitura não serão afetadas.

Anexo 5 – Instruções Pcode

Instruções de uso Geral Pcode para MSW

Opcode	P1	P2	P3	Descrição	Operação	Flags
Paddwi	a	imm		Adição imediata à palavra	*a ← *a+imm	Z, C, N
Pandn	a	B	n	And n-bytes integers	For (i=0; j<n; ++i) a[i] & b[i]	
Pandwi	a	imm		AND lógico palavra imediata	*a ← a & imm	Z, N
pbegin				Inicia a execução do pcode	Rcall pcode	
pbitwi	a	imm		Bit de teste da palavra imediata	*a & imm	Z
Pclrw	a			Limpa a palavra	*a ← 0	
pcmpbi	a	imm		Compara o byte imediato	a[0] – low(imm)	Z, C, N
pcmpn	a	b	n	Compara dois buffers na SRAM	For (i=0; i<n; ++i) a[i] – b[i]	Z
pcmpwi	a	b		Compara palavra imediata	*a – imm	Z, C, N
pcomw	a			Complemento de um da palavra	*a ← 0xFFFF - *a	Z, C, N
pdecw	a			Decrementa palavra	*a ← *a – 1	Z, C, N
Pdiv	a	b	c	Divisão unsigned 16-bit, resultado 32-bit	*a = *b / *c; *(a+2) = *b % *c	
pee2s	a	e	n	Copia bytes da EEPROM para a SRAM	For (i=0; i<n; ++i) a[i] ← eeprom[e + n];	
Pend				Pára a execução do pcode	.dw 0	
pincw	a			Incrementa palavra	*a ← *a +1	Z, C, N
pjump	addr			Jump incondicional dentro do pcode	Ppc ← addr	
pjumpeq	addr			Jump dentro do pcode se igual (if equal)	If (Z == 1)ppc ← addr	
pjumphis	addr			Jump dentro do pcode se maior/igual (if high/same)	If (C == 0)ppc ← addr	
pjumplo	addr			Jump dentro do pcode se menor	If (C == 1)ppc ← addr	
pjumpne	addr			Jump dentro do pcode se não igual	If (Z == 0)ppc ← addr	
pmemcpy	a	b	n	Cópia da memória SRAM	For (i=0; i<n; ++i) a[i] ← b[i];	
pmovb	a	b		Move byte	a[0] ← b[0]	
pmovbi	a	imm		Move byte imediato	a[0] ← low[imm]	
pmovw	a	b		Move palavra	*a ← *b	
pmovwi	a	imm		Move palavra imediata	*a ← imm	
pmul	a	i1	I2	Multiplicação unsigned 16-bit, resultado 32-bit	*a ← (i1 x i2) & 0xFFFF; *(a+2) ← (i1 x i2) >> 16	
pnegw	a			Complemento de dois da palavra	*a ← 0 - *a	Z, C, N
porwi	a	imm		OR lógico palavra imediata	*a ← *a | imm	Z, N
ps2ee	e	a	n	Copia bytes da SRAM para a EEPROM	For (i=0; i<n; ++i) eeprom[e + i] ← a[i];	
ps2see	e	a	n	Copia bytes da SRAM para SEEPROM externa	For (i=0; i<n; ++i) seeprom[e+i] ← a[i];	
psee2s	a	e	n	Copia bytes da EEPROM externa para a SRAM	For (i=0; i<n; ++i) a[i] ← seeprom[e+i]];	
pshnw	a	n		Logical shift 16-bit palavra de n bits	If (n>0) *a ← (*a << low(n)) Else *a ← (*a >> low(n));	Z, C, N
psubwi	a	imm		Subtrai palavra imediata	*a ← *a – imm	Z, C, N
pwdr				“Reseta” Watchdog Timer	Wdr	
pwreebi	e	imm		Grava byte na EEPROM	Eeprom(e)← low(imm)	

pwrseebi	e	imm		Grava byte na SEEPROM externa	Seeprom(e)←low(imm)	
pxorwi	a	imm		OR exclusivo palavra imediata	*a ← *a ^ imm, set flags with low(*a)	Z, N

Fonte: PicoWeb Pcode

Notas:

- `push(x)` é uma forma de empilhar o byte `x` para o topo da pilha AVR e `pop(a)` é uma forma de desempilhar o byte a do topo da pilha no endereço a da SRAM.
- Uma transferência de instruções de controle pcode (`pcall`, `pjump`, etc.) **deve** ter como seu endereço alvo uma outra instrução pcode;
- Pcode deve ter sua entrada na linguagem assembly AVR utilizando uma instrução `pbegin`. O retorno à linguagem assembly AVR **deve** ser feito utilizando uma instrução `pend`.

Instruções de Manipulação de Pilha e Chamada/Retorno Pcode

Opcode	p1	P2	p3	Descrição	Operação	Flags
pcall	addr			Chama rotina pcode	Empilha ppc sobre a pilha de retorno pcode; ppc ← addr	Z, C, N
pdropn	n			Tira n bytes da pilha	For (i=0; i<n; ++n) pop();	
penter	n			Determina nova estrutura da pilha pcode	Configura nova estrutura de pilha pcode de n bytes	Z, N
ppopn	a	n		Desempilha n bytes da pilha na memória	For (i=n-1; i>=0; --i) pop(a+1);	
ppopw	a			Desempilha palavra	Pop(a+1); pop(a);	Z
ppushn	a	n		Empilha n bytes sobre a pilha na memória	For (i=0; i<n; ++i) push(a[i])	
ppushwi	imm			Empilha palavra imediata sobre a pilha	Push(low(imm)); push(high(imm));	Z, C, N
Pret				Retorna da rotina pcode	ppc ← pop return pcode stack	Z
pretn	n			Retorna da rotina pcode e destrói a estrutura da pilha	Destrói a estrutura da pilha pcode atual de n bytes	

Fonte: PicoWeb Pcode

Notas:

- A estrutura da pilha pcode é determinada sobre a pilha do hardware do microcontrolador. A estrutura da pilha pcode pode ser acessada utilizando offsets a partir de um nome predefinido, que aponta para a base da estrutura da pilha pcode atual na memória SRAM;
- Qualquer dado empilhado **antes** da instrução `penter`, imediatamente acompanha a estrutura da pilha na memória.

Instruções de Entrada/Saída Pcode

Opcode	p1	p2	p3	Descrição	Operação	Flags
Pcrlf				Imprime CR, LF	Printf("\r\n")	
psgetcto	A	imm		Recebe o caracter da porta serial com timeout	if (caracter lido dentro de loops imm) a[0] ← getchar(), Z=0 else Z=1; // utiliza macro PSGETCTO_MSECS(msecs) para setar imm	Z
phexbi	imm			Imprime o byte hex imediato	Printf("%02x",(imm & 0Xff))	
pprint	s		n	Imprime string	Printf("%s",a)	
pprintb	s	a		Imprime bytes em hex com string	If (s !=0) printf("%s",s); For (i=0; i<n; ++i) printf("%02x",a[i]);	

pprinthwi	imm			Imprime palavra imediata 16-bit não-assinalada em hex	Printf(“%04x”,imm)	
pprintse	s	n		Imprime string EEPROM serial de tamanho n	For (i=0; i<n; ++i) printf(“%c”, s[i]);	
pprintswi	imm			Imprime palavra imediata 16-bit assinalada	Printf(“%6d”,imm)	
pprintuwi	imm			Imprime palavra imediata 16-bit não-assinalada	Printf(“%6u”,imm)	
pprintv	s	imm		Imprime string com valor hex word 16-bit com espaço entre linhas	If (s != 0) printf(“%s”, s) else putchar(‘ ‘); Printf(“%04x”,imm)	
pputc	imm			Imprime caracter imediato	Putchar(imm & 0xFF)	
pputcb	a			Imprime caracter	Putchar(a[0]);	
pser_getc	a			Verifica/recebe caracter da porta serial	If (caracter pronto) a[0] ← getchar(), Z = 0 else Z=1;	
pser_mode	bin			*Flush* e seta modo da porta serial	Flush quaisquer caracteres bufferizados, então if (bin = 0) setar modo normal; else setar pass-all (binary) modo	Z
pser_putc	imm			Escreve byte imediato para a porta serial	Esperar por uma indicação “transmit done” na porta serial, então escreva low(imm) para UART	
pspace				Imprime um espaço	Putchar(‘ ‘)	

Fonte: PicoWeb Pcode

Notas:

- A operação `putchar(ch)` comporta-se como segue: `if (putc_b == 0)`, o byte `ch` é escrito na porta serial. Caso contrário, `if (putc_b <> 0)`, o byte `ch` é enviado para que seja transmitido ou armazenado no buffer de transmissão;
- A operação `printf()` comporta-se identicamente à operação `putchar(ch)` descrita anteriormente.

Instruções de Rede Pcode para MSW

Opcode	p1	p2	p3	Descrição	Operação	Flags
paddn	a	b	n	Adiciona n-bytes inteiros (network byte ordering)	CF = 0; for (i-low(n)/2-1; i<>0; i-=1) *(a+2*i) ← swap (swap(*(a+2*i)) + swap(*(b+2*i) + CF);	
pf2x	off	func	n	Saída de bytes para o buffer de transmissão Ethernet iniciando com um byte off, com acumulador checksum de rede “chkacc”	for (i=0; i<n; i+=2) Xmit[off + i] ← func(); onde func retorna palavra 16-bit no registro X para ser armazenada. Um network checksum 16-bit é também acumulado no “chkacc”	
pi2x	off	imm		Saída da palavra 16-bit imediata para o buffer de transmissão Ethernet com byte off	Xmit[off] ← imm	
pprinta	s	eadd	n	Imprime bytes a partir do buffer de recepção Ethernet com label	If (s != 0) printf “%s=”, s); For (i=0; i<n; ++i) putchar (Recv[eadd + i];	
pprintr	s	off	n	Imprime palavras em hex a partir do buffer de recepção	If (s != 0) printf “%s=”, s); For (i=0; i<2*n; ++i) printf	

				atual com label	“%02x”, Recv[off + i];	
pprintt	s	off	n	Imprime palavras em hex a partir do buffer de transmissão atual com label	If (s != 0) printf “%s=”, s); For (i=0; i<2*n; ++i) printf “%02x”, Xmit[off + i];	
pr2f	func	off	n	Move bytes a partir do buffer de recepção Ethernet atual para a função chamada	for (i=0; i<n; i+=2) func(x); onde func é chamada com uma palavra 16-bit em i0 e i1 (i0=Recv[off + i] e i1=Recv[off + i + 1])	
pr2s	a	off	n	Move bytes a partir do buffer de recepção Ethernet para a memória	for (i=0; i<n; ++i) a[i] ← Recv[off + 1];	
pr2x	off1	off 2	n	Move bytes a partir do buffer de recepção Ethernet atual para o buffer de transmissão Ethernet	for (i=0; i<n; ++i) Xmit[off1 + 1] ← Recv[off2 + i];	
ps2x	off	a	n	Move bytes a partir da memória para o buffer de transmissão Ethernet	for (i=0; i<n; ++i) Xmit[off + i] ← a[i];	
Opcode	**P1**	**P2**	**P3**	**Descrição**	**Operação**	**Flags**
psetparm	imm			Seta parâmetros de memória global	*psetparm_parm = imm	
purl2int	a	pof f		Converte caracteres em linha URL para inteiros 16-bit (utiliza purl2s)	Sscanf(poff, “%d”, a); incrementa *poff para apontar para o próximo “não-dígito”	

purl2s	a	off	n	Move bytes a partir do buffer de recepção Ethernet atual (ou EEPROM serial) para a memória	If (off & 0x8000)
					For (i=0;i<n;++i) a[i] ← *(((char*)off)+1);
					Else for (i=0; i<n; ++i) a[i] ← Recv[off + 1];
					Z=0 se recebe buffer overrun, se não Z=1;
						Z

purl2scmp	poff	s		Compara strings *NULL-Terminated* com linha URL (utiliza purl2s)	Compara a string s iniciando no offset *poff. Se strings iguais, inteiras, Z=1 e *poff +=strlen(s). Se há qualquer diferença, Z=0 e *poff não é modificado.	Z
purlparm	poff	s		Encontra a string de parâmetro na linha URL (utiliza purl2s)	Pesquisa linha URL para string s (verificando apenas depois de cada caracter ? e &). Se encontra, Z=1 e *poff é setado para o próximo offset depois da string igual. Se não encontrado, Z=0 e *poof não é modificado.	Z
px2s	a	off	n	Move bytes a partir do buffer de transmissão Ethernet para a memória	For (i=0;i<n;++i) a[i] ← Xmit[off + i];	
pz2x	off	n		Escreve n bytes de zero para buffer de transmissão	For (i=0;i<n;++i) Xmit[off + i] ← 0;	

Fonte: PicoWeb Pcode

Notas:

- `Xmit[0]` é o primeiro byte do buffer de transmissão Ethernet e `recv[0]` é o primeiro byte do buffer de recepção Ethernet. Ambos apontam para o primeiro byte do endereço MAC Ethernet no buffer respectivo;
- A operação `printf()` comporta-se identicamente à operação `putchar(ch)` descrita anteriormente;
- Depois que uma solicitação HTTP GET é recebida e “um arquivo” do MSW conhecido é localizado na linha URL, a variável 16-bit global de memória

`url_parms`, é setada para offset no buffer de recepção Ethernet, apenas depois do nome do arquivo. Se há parâmetros seguintes ao nome do arquivo (denotado por um ? (ponto de interrogação)), então `url_parms` aponta para o primeiro parâmetro URL.

Anexo 6 – Tabela de Variáveis de contexto CGI

Variáveis de Contexto CGI- Fonte: Tutorial de CGI

Variável de Contexto	Descrição
CONTENT_LENGTH	Contém o número total de caracteres que o script CGI precisa ler como parâmetros enviados através do método POST
CONTENT_TYPE	Contém o Mime-Type dos parâmetros enviados ao script CGI através do método POST. Se o script for chamado através do método POST de um formulário, o Mime-Type típico é application/x-www-form-urlencoded
GATEWAY_INTERFACE	Contém a versão da plataforma CGI instalada no servidor web, por exemplo, CGI/1.1
HTTP_ACCEPT	Contém a lista de Mime-Types que o navegador requisitante suporta. O valor */* significa que o navegador suporta todos os tipos
HTTP_REFERER	Contém o endereço URL da página que chamou o script CGI. Nem todos os navegadores fornecem este parâmetro, portanto, esta variável nem sempre tem um valor disponível
HTTP_USER_AGENT	Contém informações do tipo e da versão do navegador requisitante. Portanto, é possível saber qual navegador o usuário está utilizando
PATH_INFO	Contém informações de path quando o script é solicitado através do método GET, sempre relativo ao diretório raiz do servidor web
PATH_TRANSLATED	Contém informações especiais de path quando o script é solicitado através do método GET. Difere do PATH_INFO porque corresponde à estrutura de diretórios do servidor
QUERY_STRING	Contém uma string de dados fornecida ao script através de uma chamada pelo método GET. Os dados seguem o padrão Mime-Type application/x-www-form-urlencoded
REMOTE_ADDR	Contém o endereço IP do servidor web através do Qual o script CGI foi chamado se este servidor não for o que contém o script. Este valor nem sempre pode ser obtido
REMOTE_HOST	Contém o endereço do Domínio do servidor web através do qual o script CGI foi chamado se este servidor não for o que contém o script. Este valor nem sempre pode ser obtido
Variável de Contexto	**Descrição**
REMOTE_IDENT	Contém informações de protocolo se no servidor web estiver ativado o protocolo ident para acessos protegidos
REMOTE_USER	Contém o nome do usuário que chamou o script CGI. Este valor só está disponível se a autenticação do servidor web estiver ativada
REQUEST_METHOD	Contém o método pelo Qual o script CGI foi chamado, ou seja, GET ou POST. Um script CGI pode, por exemplo, obter o valor dessa variável para poder decidir como irá receber os parâmetros enviados: através da entrada padrão (do método POST) ou da variável de contexto QUERY_STRING (do método GET)
SCRIPT_NAME	Contém o endereço URL do script CGI chamado relativo ao endereço de domínio, ou seja, por exemplo, /cgi-bin/busca.pl
SERVER_NAME	Contém o nome do servidor web (www.xyz.com, por exemplo) ou seu endereço IP (127.0.0.1, por exemplo)
SERVER_PORT	Contém o número da porta que foi designada para o servidor. Normalmente, para servidores web, a porta é 80
SERVER_PROTOCOL	Contém a versão do protocolo HTTP que o servidor web utiliza, por exemplo, HTTP/1.0
SERVER_SOFTWARE	Contém a definição do software instalado no servidor web, por exemplo, OmniHTTPd/2.0a1 (Win32; i386)

Anexo 7 – RTL8019 Diagramas