Marco Antonio Santuci Carvalho
Orientador: Claudionor Nunes Coelho Jr.

Um Sistema de Monitoramento Remoto de Pacientes usando Rede sem Fio

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Novembro, 2005

Belo Horizonte

Resumo

Este trabalho implementa um sistema de monitoramento remoto de pacientes através da comunicação pela Internet utilizando o protocolo TCP/IP e uma rede sem fio. A aplicação principal do trabalho é voltada para a área biomédica onde o paciente pode ter alguns dados fisiológicos monitorados 24 horas por dia ou envia-los para o seu médico em caso de emergência.

Para a implementação do trabalho foi utilizado um circuito de supervisão de eletrocardiograma (*ECG*) que teve de ser adaptado para fazer a comunicação por rede sem fio através de um driver de dispositivo (*Device Driver*). Duas aplicações que executam em cima da pilha TCP/IP foram adaptadas e testadas para funcionamento com fins de monitoramento de pacientes de forma diferenciada. A implementação dos trabalhos considerou aspectos tais como custo, gerenciamento de energia e tolerância à falhas, fatores indispensáveis em um equipamento de monitoramento remoto na área médica.

Os grandes desafios do projeto foram a implementação do driver do dispositivo de comunicação sem fio e a adaptação do circuito *ECG* para comunicação com a interface que implementa a rede sem fio. A grandes restrições de projeto foram a elaboração do software considerando os escassos recursos de um microcontrolador de oito bits e também a adaptação do projeto a partir de um circuito já existente.

Durante o desenvolvimento dos trabalhos para a implementação do sistema de comunicação sem fio, foi identificado a necessidade de alteração do hardware inicial. Devido a limitada disponibilidade de recursos de hardware, algumas funcionalidades originais da placa tiveram de ser substituidas em troca do funcionamento correto do sistema de comunicação. Como solução para este problema é proposto um novo projeto que adiciona pouca modificação no circuito original e mantém a proposta de baixo custo.

Aplicações práticas do projeto foram testadas e comprovaram que ele pode ser utilizado não somente para aplicações médicas, mas também em outras que necessitem de comunicação utilizando rede sem fio.

Abstract

This Work implements a remote system to monitor medical patients by using Internet communication through TCP/IP protocol and wireless network. The main application is in the biomedical area where the patient can have some physiological data monitored 24 hours per day or send it to his doctor in an emergency situation.

The system was developed using a circuit already done that measure *ECG* data. It was modified to support wireless network using a *Device Driver*. Two applications runs in the top of the TCP/IP stack were developed and adapted to monitor patients in different ways. The project implementation considered some issues like cost, energy management and fault tolerance.

The main project challenges were the *Device Driver* implementation and the circuit modification to support wireless network. The biggest difficulties were the software implementation considering the poor 8-bit microcontroller resources and project adaptation from the circuit already existed.

Some practical project applications were tested and proved that it is not limited to be used only in the medical area but also in others that needed communication using wireless network.

Agradecimentos

Agradeço primeiramente a Deus que me deu forças e me ajudou a continuar na trajetória do término desta jornada apesar dos momentos mais difíceis quando cheguei a pensar em desistir.

Aos meus pais, Antônio e Cristina que me deram apoio e condições de vida para que eu tenha chegado até aqui. Aos meus irmãos, Júlio e Patrícia que mesmo distantes sempre tinham uma palavra de incentivo.

À Cíntia que sempre esteve ao meu lado me suportando no que eu precisava e entendendo as noites e fins de semana de incansável trabalho.

Ao meu orientador, Claudionor que me deu liberdade no desenvolvimento dos trabalhos de acordo com as minhas habilidades e conseguiu me motivar a buscar novos conhecimentos.

Aos colegas de trabalho da Jabil que me incentivaram e apoiaram desde o inicio para que eu conseguisse frequentar as aulas e possibilitaram que eu pudesse realizar este trabalho.

A todas as outras pessoas que me auxiliaram nos trabalhos seja direta ou indiretamente.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 A Necessidade do Monitoramento Remoto de Pacientes

As pessoas, em geral as mais idosas, são carentes de novas tecnologias que facilitem o acesso delas a um atendimento médico mais adequado com a sua condição física, que muitas vezes debilitada, impede a sua locomoção até os hospitais. Os hospitais na maioria das vezes não oferecem o atendimento adequado devido a ausência de recursos financeiros, disponibilidade de profissionais, quantidade de leitos disponíveis dentre outros. Além disso o custo de internação de um paciente, principalmente de um idoso, é muito alto.

Geralmente as pessoas visitam o médico com a frequência de no mínimo uma vez por ano para uma avaliação geral ou quando apresentam um quadro clínico de uma doença que as incomoda. O médico por sua vez não consegue diagnosticar certos sintomas por falta de um acompanhamento mais contínuo da vida do paciente. Em uma pessoa idosa, de acordo com a natureza humana, é bem mais provável que maiores problemas de saúde possam ocorrer e por isto um acompanhamento mais frequente pode evitar doenças e até mesmo a morte.

O monitoramento remoto de pacientes permite que vários destes problemas sejam amenizados. Sensores acoplados a um paciente podem coletar dados fisiológicos e os transmitirem para uma central médica remota para que sejam analisados ou para um posterior diagnóstico. Isto evita por exemplo que uma pessoa idosa tenha que se locomover até um hospital ou ao consultório de seu médico para consulta eventual ou de emergência.

O monitoramento remoto possibilita também diminuição dos gastos dos estabelecimentos médicos pois reduz a quantidade de consultas. O monitoramento remoto possibilita ao médico condições de dar um diagnóstico mais preciso porque é possível acompanhar os dados históricos do paciente e não somente aqueles coletados no momento da consulta. O monitoramento de dados à distancia surge então como solução para diversos problemas do cotidiano de pacientes médicos, representados principalmente pelas pessoas idosas

conforme discutido. A comunidade científica é atenta à necessidade de tratamento desta situação tem feito várias pesquisas nesta área que ainda é aberta a vários questionamentos e alternativas.

Diversos aspectos dividem os pesquisadores em relação ao monitoramento remoto de pacientes. Um deles diz respeito ao tipo de controle da medicação. Uma medicação próativa de forma que o paciente possa ter todos os dados em mãos e controlar seu estado de saúde é uma alternativa de acompanhamento médico à distância. Esta forma de acompanhamento não pode ser generalizada com sucesso no caso de idosos porque pode requerer algum conhecimento médico ou a habilidade para lidar com números, dados e/ou gráficos ou sinais luminosos ou sonoros que podem gerar confusão podendo ao invés de ajudar, atrapalhar com a possibilidade de algum diagnóstico errado. O ideal seria que o paciente pudesse coletar seus dados fisiológicos tais como pressão, temperatura, etc e os pudesse repassar remotamente via um sistema de comunicação. Por outro lado, estes dados seriam recebidos por uma central computacional que faria a análise deles. Baseado em dados históricos do paciente, informações de sintomas previamente fornecidas ou até mesmo modelos matemáticos, a central remota faria algum diagnóstico e enviaria para o paciente. O sistema poderia também processar os dados recebidos do paciente e envia-los para o médico quando algum dado saísse fora do padrão esperado cabendo então o médico enviar ou contactar o paciente para lhe dar o diagnóstico e a devida medicação. Uma idéia desta ultima alternativa é mostrada na figura 1.1.

1.2 Uma alternativa de sistema para o Monitotamento Remoto de Pacientes

Os primeiros equipamentos de monitoramento contínuo de pacientes que ainda são utilizados hoje em dia tais como o *Holter* e *MAPA* utilizam o conceito *off-line* onde o paciente tem os dados dos sensores fisiológicos armazenados em algum tipo de memória e depois devem descarrega-los em um outro equipamento que possibilita futura análise. Equipamentos deste tipo tem utilidade limitada sendo aplicáveis em situações onde os dados dos sensores podem ser analisados no final de um dia, não necessitando de acompanhamento em curtos intervalos, de hora em hora por exemplo. Outro inconveniente é o fato de o paciente ter que se dirigir ao centro médico onde se localiza um equipamento para descarregar os dados.

A monitoração dos dados em tempo real (*on-line*) tem revolucionado a medicina através do surgimento de dispositivos embutidos, que incorporam diversos sensores capazes de monitorar várias atividades fisiológicas dos pacientes. Estes dispositivos são dotados de sistemas de comunicação capazes de enviar os dados para serem analisados nos centros médicos especializados. Estes sistemas de comunicação utilizam tecnologia sem fio, o que garante ao paciente mobilidade na sua região de cobertura.

Seguindo este raciocínio, pesquisadores tem concentrado seus estudos em duas áreas principais: tornar os dispositivos embutidos cada vez mais completos em termos de sensores e desenvolver aplicações para análise dos dados pelo próprio dispositivo. Pouco se tem

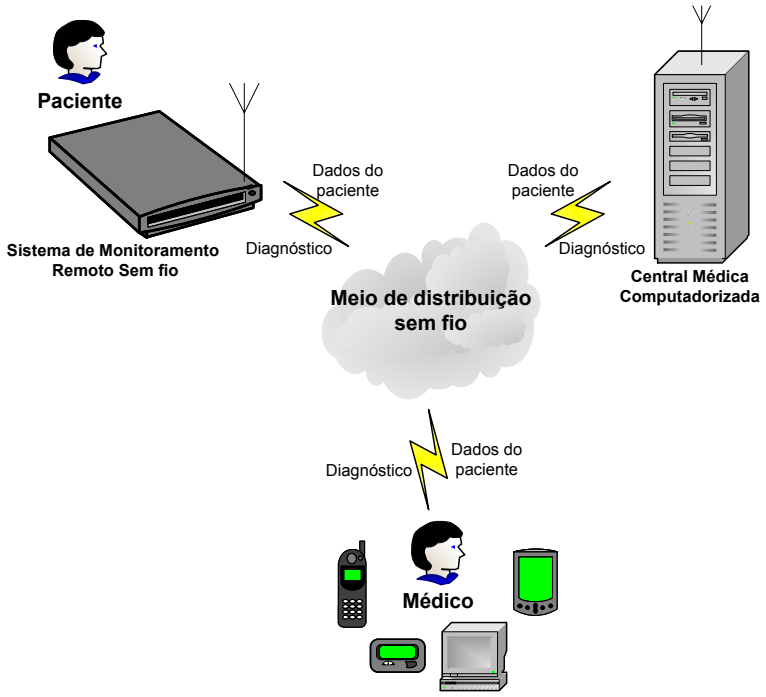


Figura 1.1: Sistema de Monitoramento Remoto com análise e diagnóstico dos dados do paciente feitos remotamente

preocupado com a compactação do hardware do dispositivo, que na maioria dos casos implementados, utiliza mini PCs ou PDAs. Esta solução não sendo customizada acaba encarecendo o dispositivo o que é economicamente inviável em se falando em distribuição em larga escala. Outros fatores que também pesam na utilização destes dispositivos com o hardware genérico é o seu peso comparado a uma solução que poderia ser mais compacta e o consumo com a utilização de vários sensores.

Como exemplo de um sistema de monitoramento remoto, pode-se citar dois sistemas desenvolvidos no DCC/UFMG. O primeiro deles é o Monitor de Sinais Vitais Multiparamétrico Vestível [Con01] que foi implementado para monitoração de parâmetros vitais tais como a suspervisão do eletrocardiograma (ECG) e nível de oxigenação do sangue. Neste projeto foram implementados o hardware para aquisição de sinais e o software para aquisição e transmissão dos dados em rede. Este projeto se caracterizou também pelo desenvolvimento de uma vestimenta apropriada de forma a carregar todo o hardware, permitindo mobilidade ao paciente. O projeto possibilitou uma integração de diversas tecnologias de hardware e software mas teve como ponto crítico o peso dos equipamentos e a autonomia de energia.

O Outro sistema desenvolvido no DCC/UFMG foi o monitor cardíaco [FLCJ03]. Este trabalho apresentou uma nova proposta para o desenvolvimento de um sistema embutido para monitoramento de dados fisiológicos de seres humanos através da utilização de um

hardware com componentes eletrônicos com maior escala de integração, conseguindo-se com isto uma redução do tamanho do hardware. Este projeto, ao contrário do anterior, foi projetado a príncipio para comunicação sem fio a partir de uma rede celular embora não tenha sido implementado nenhuma aplicação para teste deste sistema. Este projeto foi caracterizado pelo esforço na redução do hardware anterior mantendo algumas de suas funções. A parte de software de aplicação e do sistema de comunicação do projeto não foram muito exploradas.

Dando continuidade de pesquisa na área biomédica e desenvolvimento de sistemas embutidos na área de monitoramento remoto de pacientes, este trabalho desenvolve uma aplicação focando na implementação de um sistema de comunicação sem fio utilizando o mesmo hardware do projeto Monitor Cardíaco. Além disso, o projeto pretende ser mais eficiente no consumo de energia comparado com o Monitor Multiparamétrico Vestível.

Pretende-se também mostrar com este trabalho a possibilidade de ter um sistema dedicado para monitoramento de pacientes à distância através de um sistema de comunicação sem fio, mesmo que as características de hardware sejam limitadas. As restrições de hardware também provocaram a busca de soluções e utilização de técnicas que serão mostradas na apresentação do trabalho. Os limites das aplicações serão testados e verificados e será identificado o que é factível de ser implementado com o hardware proposto.

A contribuição esperada para o projeto é a implementação de um sistema remoto de monitoramento de pacientes utilizando uma rede de comunicação sem fio priorizando baixo custo e consumo.

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta as necessidades e adequações do projeto de acordo com o hardware existente; no Capítulo 3 são analisados trabalhos relacionados com as decisões de projeto do capítulo anterior. O Capítulo 4 apresenta o desenvolvimento do sistema de comunicação utilizado, com as devidas considerações para integração com rede sem fio e também o desenvolvimento das aplicações incluindo suporte para redução do consumo de energia e tolerância a falhas. O Capítulo 5 apresenta os testes, resultados, principais dificuldades e, finalmente, o Capítulo 6 trata da nova versão do projeto, das conclusões e dos trabalhos futuros.

Capítulo 2

Necessidades e Decisões de Projeto

O objetivo deste capítulo é apresentar as considerações, necessidades e decisões para o projeto de um sistema de monitoramento médico remoto com sistema de comunicação sem fio. O hardware para o trabalho foi previamente definido com base nos pré requisitos que serão apresentados e portanto não foi necessário a alteração ou melhoria desta parte do circuito cabendo portanto definições, decisões e modificações no que diz respeito ao sistema de comunicação cujo hardware não tinha sido originamente projetado. Em relação ao software, são levantadas algumas questões importantes para o projeto tal como o gerenciamento de energia do sistema.

As necessidades para o projeto serão enfatizadas e algumas soluções para o projeto serão apresentadas. Logo em seguida, baseando-se nestas necessidades e nas restrições de projeto, as decisões de maneira geral serão apresentadas. Decisões e implementações mais específicas do projeto serão mostradas e discutidas nos capítulos seguintes.

2.1 Pré-Requisitos de Projeto

Primeiramente, para ser um monitor de dados fisiológicos, obviamente o projeto precisa ter a capacidade de monitorar dados fisológicos dos pacientes. O objetivo aqui não é monitorar diferentes tipos de dados *multiparamétrico* conforme outros projetos mas sim conseguir monitorar remotamente um ou mais dados. Através deste princípio, o conceito do monitoramento pode ser extensível facilmente, apenas ampliando a quantidade de hardware e processamento de dados via software.

É necessário também que o monitor remoto seja leve para que possa ser utilizado pelo paciente como um acessório qualquer tal como uma carteira ou um telefone celular sem que isto incomode ou atrapalhe seus movimentos.

O sistema precisa ter comunicação sem fio para que possa ser usado como um monitor remoto de forma a permitir mobilidade não ficando o paciente preso a fios. Além disso, este sistema de comunicação deve ser implementado de forma a eliminar ou minimizar problemas de ruído, recepção errada de dados além de ser acessado pela Internet. Deve

ser veloz em situações de envio e recepção de dados permitindo rapidez no diagnóstico e possibilitar a medicação de pacientes em estado de emergência.

Em termos de aplicação, é necessário que o paciente consiga abrir uma conexão a qualquer momento que desejar para o envio de dados e também que o médico possa acessar os dados remotamente a qualquer momento conseguindo abrir uma conexão com o monitor cardíaco.

É necessário também que o sistema tenha um baixo consumo para que o monitor remoto possa ser utilizado pelo paciente permitindo grandes deslocamentos sem necessidade de recarga de bateria, possibilitando um monitoramento mais prolongado.

De maneira geral é importante que o código de programa que implemente o sistema de comunicação e as aplicações execute rapidamente, ou seja, possua um código eficiente para que o tempo de execução dele seja rápido. Um outro aspecto a ser considerado é o tamanho do código. O espaço ocupado pelo código não pode ocupar grande parte da memória de programa, para poder ser capaz de dar liberdade para o desenvolvimento ou melhoria de aplicações e funcionalidades.

2.2 Análise de algumas soluções para o projeto

2.2.1 O projeto existente : Monitor Cardíaco

O monitor cardíaco[FLCJ03] foi projetado com a finalidade de monitoramento de dados fisiológicos de seres humanos, auxiliando na detecção de problemas relativos a saúde e bem estar, mais especificamente, através de uma avaliação da condição cardíaca e de oximetria do paciente. Este projeto procurou otimizar o tamanho do hardware através da compactação de circuitos utilizados no tratamento dos sinais analógicos vindos dos sensores de medição de eletrocardiograma (ECG).

O monitor cardíaco portanto atende ao primeiro pré-requisito para o projeto que seria o monitoramento de pelo menos um dado fisiológico do paciente. Devido a integração do hardware, este sistema ficou com o peso reduzido que incluindo 3 pilhas de tamanho AA ficou em torno de 170g. Com isto pode-se dizer que o monitor cardíaco também atende ao segundo pré requisito de projeto relacionado com o peso do produto.

Foi pensado, na primeira concepção do projeto, em se monitorar os dados remotamente utilizando-se uma rede de telefonia celular no padrão GSM. O formato da placa foi preparado para ser acoplado a um celular. Embora o sistema tenha sido preparado para suportar tal tipo de comunicação, não existem registros e relatos que comprovem a implementação de um sistema para comunicação remota sem fio no sistema em questão. Portanto, pode-se dizer que com as suas características originais, o sistema não atendia ao terceiro pré-requisito que é a comunicação sem fio.

Em relação ao consumo de energia, não foi feita também nenhuma análise do projeto em relação a este pré-requisito. Consequentemente, não é possível afirmar se o projeto atendia ou não tal pré-requisito.

2.2.1.1 Características do projeto Monitor Cardíaco

Para um melhor entendimento das características do projeto Monitor Cardíaco, o Diagrama em blocos dele é ilustrado na figura 2.1.

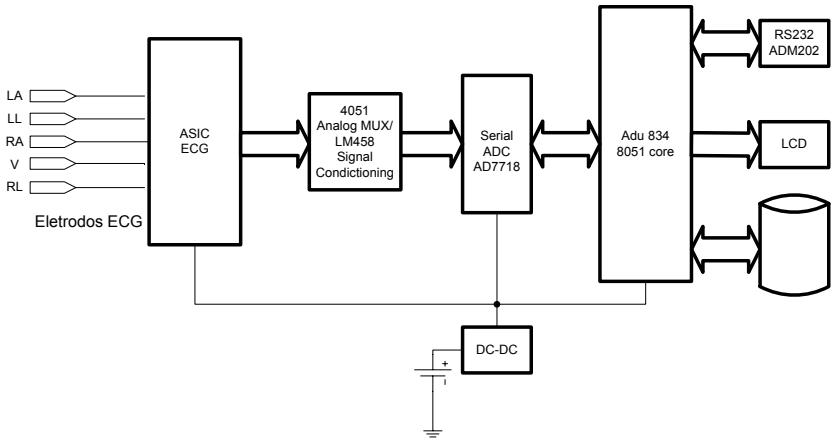


Figura 2.1: Diagrama em Blocos do Monitor Cardíaco

O componente responsável pelo controle das funcionalidades da placa é o microcontrolador de 8 bits ADu834[Dev03] da Analog Devices. Este microcontrolador possui um núcleo básico do popular 8051[Cor98] e é bastante versátil possuindo um conjunto de características bastante completo. Dentre elas pode-se destacar:

- 62Kbytes de Memória Flash de Programa;
- 4Kbytes de memória Flash de dados;
- 2304 bytes de memória RAM;
- 26 pinos de I/O programáveis;
- 11 fontes de interrupção;
- Portas seriais nos padrões UART*(Universal Asynchronous Receiver-Transmitter)*, SPI*(Serial Peripheral Interface)* e I2C*(Inter-Integrated Circuit)*;
- Temporizador *Watch-Dog*;
- Monitor de alimentação (PSM);
- Consumo da ordem de 2,3mA no modo normal e 20uA no modo de economia de energia;

Com o intuito de compactação do circuito foi utilizado um circuito integrado *ASIC (Application Specific Integrated Circuit)* ECG ASIC da Welch Allyn[Tec01] que implementa uma solução completa de ECG. Como componentes complementares foi utilizado um conversor Analógico-Digital (ADC) de baixo ruído para converter as amostras analógicas das saídas do ASIC, um conversor DC-DC compacto para geração das tensões de operação dos componentes, um driver padrão para comunicação RS-232 e uma unidade de memória de massa baseado na interface padrão CompactFlash.

2.2.2 O padrão Compact Flash

O padrão Compact Flash[Com03] ou PC-card foi introduzido em 1994, inicialmente para funcionar como um dispositivo armazenador de dados de acordo com o padrão *IDE (Integrated Drive Eletronic)*. Este modo de funcionamento iremos chamar de agora em diante de modo memória. O padrão Compact Flash é uma versão reduzida de 50 pinos do padrão da *PCMCIA (Personal Computer Memory Card International Association)* [PCMb] que possui 68 pinos. Além do modo memória, o padrão Compact Flash pode ser utilizado no que chamaremos de modo I/O onde dispositivos de entrada e saída tais como modems, cartões ethernet, portas seriais, interfaces sem fio Bluetooth e WI-FI são implementados.

Os cartões Compact Flash são utilizados em dispositivos embutidos no modo memória para armazenamento de dados de câmeras fotográficas digitais por exemplo. Já no modo I/O, os cartões Compact Flash podem ser utilizados como interface de rede sem fio WI-FI em Assistentes Digitais Pessoais (*PDAs*).

A pinagem do padrão Compact Flash utilizada pelo Monitor Cardíaco é mostrada na tabela 2.2. Devido a limitação da quantidade de pinos de Entrada/Saída do microcontrolador do projeto monitor cardíaco, este foi configurado para interfacear com o padrão Compact Flash utilizando um barramento de oito bits embora o padrão suporte um barramento de 16 bits. Desta forma trataremos neste trabalho somente da configuração do padrão no modo 8 bits.

Desta maneira, uma possível alternativa de projeto utilizando-se o monitor cardíaco seria a utilização da interface Compact Flash no modo I/O com um cartão de interface de rede sem fio WI-FI, por exemplo. Com isso, o hardware da interface sem fio seria facilmente integrado ao projeto, possibilitando com isto a manutenção do tamanho original do circuito.

2.2.3 Protocolo de interface de rede 802.11 - WI-FI

O protocolo de interface de rede IEEE 802.11 [IEE99] surgiu da necessidade de se conectar dispositivos à internet no modo *Wireless* de modo a possibilitar completa mobilidade a eles. Nos dias de hoje, a rede de distribuição dos pontos de acesso(*APs*) para acesso às redes sem fio 802.11 é bem diversificada podendo ser encontrada em restaurantes, redes de Fast-Food, livrarias, cafeterias, hotéis, aeroportos, shoppings, universidades, estádios de futebol e em muitos outros. A área de cobertura destas redes padrão 802.11 pode ir desde

Nome	**Pino(s)**	**Função**
A3:A0	17, 18, 19, 20	Linhas de Endereço
D7:D0	6, 5, 4, 3, 2, 1, 23, 22, 21	Linhas de dados (8 bits)
-CE1	7	habilitação do Cartão (8 bits)
-OE(RD)	9	Habilitação de leitura de memória Comum e de atributos do cartão.
-WR	36	Habilitação de escrita em memória Comum e de atributos do cartão
CD1	26	Detecta se cartão está inserido corretamente no conector Compact Flash

Tabela 2.1: Pinagem do padrão Compact Flash com pinos usados pelo projeto monitor cardíaco

pequenos escritórios até uma cidade inteira como é o caso de Amsterdã que foi a primeira capital européia a ser coberta por uma rede sem fio[For05].

Uma tendência que favorece a expansão deste protocolo é a sua aceitação pela industria mundial. Isto contribui para o desenvolvimento de novas tecnologias e produtos relacionados com ele. É o caso de alguns notebooks atualmente que já dispensam placas de rede sem fio e já embutem o hardware que implementa o protocolo diretamente. Periféricos tais como inpressoras, webcams e câmeras digitais também estão aderindo a utilização do protocolo. Sistemas de som, televisores, aparelhos de DVD, telefone IP são apenas alguns eletroeletrônicos onde também é possível encontrar acesso pelo protocolo 802.11.

O protocolo 802.11 também é conhecido por WI-FI, ou *Wireless Fidelity* . Ele foi projetado, a príncipio, para operar em três meios físicos diferentes sendo dois deles baseados em Espalhamento de frequência de rádio(*Spread Spectrum*) e o outro baseado em infra vermelho. Hoje em dia a tecnologia por espalhamento de frequência de rádio usando sequência direta (*Direct Sequence*) é a mais utilizada.

Com a utilização de uma pilha de protocolos no padrão TCP/IP que inclui o WI-FI como camada de interface de rede é possível a conexão do projeto à Internet. Portanto, o padrão 802.11 pode ser utilizado embutido em um hardware de um cartão Compact Flash no modo I/O para atender o pré requisito de conexão do monitor cardíaco à Internet e permitir que isto seja feito sem o uso de fios atendendo a outra necessidade de projeto.

2.2.3.1 Versões do Protocolo

O protocolo 802.11 é descrito como sendo uma combinação de um protocolo de Controle de Acesso ao Meio (*MAC*) e Controle da Camada de Ligação (*LLC*). Hoje em dia existem versões do protocolo 802.11 que são representadas principalmente pela versão 802.11b criada em 1999 e 802.11g criada em 2003. Existe também o padrão 802.11a criado em 1999 que comercialmente não foi bem aceito devido ao alcance limitado, falta de compatibilidade com o 802.11b e custo. Na versão 802.11b pode-se atingir velocidades em teoria de até 11Mbps e na 802.11g a especificação para a velocidade máxima de comunicação é de 54Mbps. Esta velocidade diminui com a distância conforme mostrado na figura 2.2 encontrada em [Cor03].

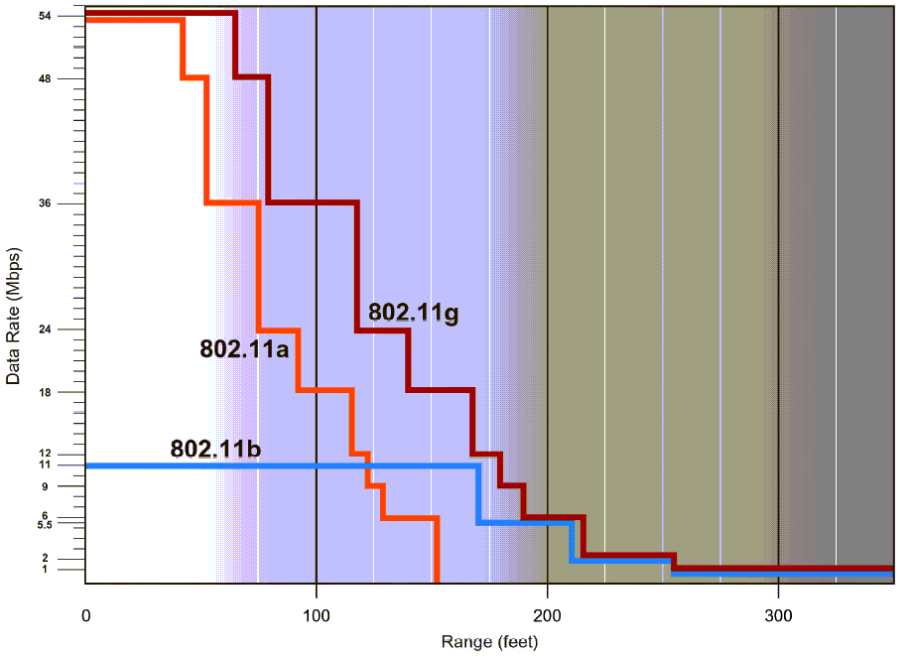


Figura 2.2: Taxas de comunicação dos padrões 802.11 em função da distância do ponto de acesso

Outra característica dos padrões são as frequências de operação. O 802.11b e 802.11g operam na faixa de frequência em torno dos 2.4GHz (banda S ISM) enquanto que o 802.11a opera em 5GHz. Estas e outras características podem ser visualizadas na tabela 2.2.

Embora o padrão 802.11g seja mais veloz, como o 802.11b surgiu primeiro, ele hoje possui maior representatividade que em 2003 era de 95%. Mais detalhes sobre os tipos de redes sem fio e sobre o padrão 802.11 podem ser encontrados no Anexo B.

Característica	**802.11b**	**802.11a**	**802.11g**
Aprovação do padrão	Julho 1999	Julho 1999	Junho 2003
Taxa de comunicação Máxima	11Mbps	54Mbps	54Mbps
Modulação	CCK	OFDM	OFDM e CCK
Taxa de dados	1, 2, 5,5, 11Mbps	6, 9, 12, 18, 24, 36, 48, 54 MBps	CCK: 1, 2, 5,5, 11Mbps OFDM: 6, 9, 12, 18, 24, 36, 48, 54 MBps
Frequências	2.4-2.497GHz	5.15-5.35GHz, 5.425-5.675GHz, 5.725-5.875GHz	2.4-2.497GHz

Tabela 2.2: Características das versões do Protocolo 802.11

2.2.4 A Pilha TCP/IP

O protocolo TCP/IP é o padrão mais usado na atualidade para a comunicação em redes de computadores. Ele ganhou dimensão mundial porque foi projetado para prover comunicação entre diferentes tipos de sistemas computacionais e cada um deles podendo rodar um sistema operacional distinto.

Ele Surgiu, na década de 1960, através de pesquisas sobre redes comutadas por pacotes feitas ARPA(Advanced Research projects Agency), uma das agências de pesquisa e desenvolvimento do Departamento de defesa dos Estados Unidos. No inicio era um padrão fechado mas com a abertura para utilização por universidades ganhou uma maior popularidade.

O protocolo TCP/IP na verdade é o núcleo principal de protocolos que integram o que podemos chamar de arquitetura da Internet. Esta arquitetura em alguns casos é chamada de pilha TCP/IP por possuir os dois principais protocolos, TCP (Transport Control Protocol) e IP (Internet Protocol), e outros divididos em camadas. Estas camadas situam-se uma em cima da outra como se estivessem empilhadas conforme figura 2.3. Neste texto iremos referenciar pilha TCP/IP como sendo o conjunto de protocolos que integram a arquitetura de uma rede Internet.

A figura 2.3 sugere que cada nível da pilha faça interação com o outro não sendo restrito a cada protocolo. Por exemplo, as funções executadas pelo protocolo TCP podem ocasionalmente executar funções do protocolo IP e vice versa. Estas interações são importantes na otimização do código de programa.

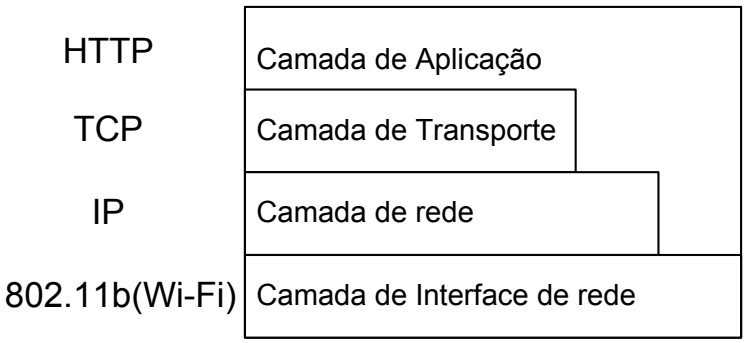


Figura 2.3: Pilha TCP/IP com alguns de seus protocolos

Fazendo um breve resumo da função de cada camada pode-se citar:

- Camada de Aplicação: Provê uma aplicação final para o usuário. No caso do HTTP, permite que páginas em HTML com dados sejam visualizadas em *Browsers*.
- Camada de Transporte: Identifica qual é o tipo de aplicação a qual os dados se destinam. No caso do TCP, implementa um mecanismo de transferência de dados confiável através de transmissão orientada por conexão, onde cada mensagem é identificada, verificada em relação a erros, reconhecida e caso necessário retransmitida para garantir confiabilidade na troca de informações.
- Camada de rede: Faz o gerenciamento dos dados dentro da rede orientada por pacotes. No caso do IP Implementa endereçamento e roteamento dos datagramas e permite que redes físicas diferentes sejam interligadas. Não implementa um mecanismo de transferência de dados confiável para as camadas superiores.
- Camada de Interface de rede: Como o nome sugere, faz a interface física entre os níveis superiores e nível de rede. No caso do protocolo ARP mapeia endereços de rede em endereços de internet. No caso do 802.11b implementa mecanismo de autenticação, associação e comunicação com outros nós da rede. Tudo isto feito sem o uso de fios (Wireless).

Cada protocolo possui um formato de organização de dados semelhante ao outro sendo iniciado por um cabeçalho com campos variando de bits a 6 ou mais bytes seguido dos dados. Dependendo da camada, os dados de uma delas serão o cabeçalho e dados do nível superior conforme mostrado na figura 2.4.

Mais informações sobre a pilha de protocolos TCP/IP podem ser obtidas no anexo A.

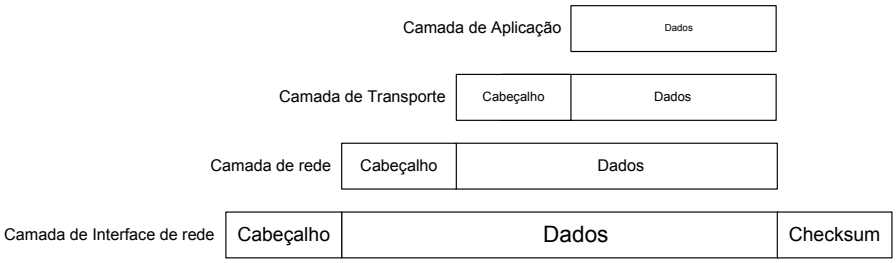


Figura 2.4: Estrutura dos dados em cada nível da pilha de protocolos TCP/IP

2.3 Decisões de Projeto

A primeira decisão conforme pode-se ter percebido ao longo deste capítulo, foi utilizar como hardware para o monitoramento remoto de pacientes, o monitor cardíaco. O principal motivo foi o fato de ser um circuito compacto que reune funcionalidade para monitoramento de sinais vitais de pacientes e também por ser um circuito com peso e tamanho razoáveis de serem carregados por um ser humano sem incomodar. Outros fatores também contribuiram para a escolha tais como ter sido projetado no DCC/UFMG permitindo continuidade de pesquisa, ferramentas de desenvolvimento facilmente disponíveis, etc. Com esta decisão de projeto os primeiros dois pré-requisitos de projeto podem ser atendidos.

Com o projeto do circuito já definido, não foi possível escolher muito em relação ao que usar como interface de rede sem fio. Um projeto que envolve comunicação sem fio, possui um projeto de hardware complexo e está fora do escopo desta dissertação. Portanto seria necessário a utilização de uma alternativa pronta e desta forma decidiu-se pela interface de rede sem fio WI-FI devido a disponibilidade de um conector padrão Compact Flash no projeto do monitor cardíaco podendo ser utilizado no modo I/O e a possibilidade de interligação do projeto à internet.

O protocolo *Bluetooth* foi descartado devido ao seu reduzido alcance e alto consumo. A velocidade de comunicação da aplicação não foi um fator de decisão e portanto embora ela seja maior no *Bluetooth* que o padrão 802.11b.

Para interligação do dispositivo à internet foi escolhido a pilha de protocolos TCP/IP por sua grande difusão, popularidade e adequação ao projeto conforme será visto no capítulo 3.

Através da escolha do hardware e do protocolo de comunicação sem fio será necessário a adoção de técnicas para implementação da pilha TCP/IP, do driver de dispositivo sem fio e também para o gerenciamento de energia do sistema conforme será visto nos capítulos seguintes.

Capítulo 3

Trabalhos relacionados

O presente capítulo relaciona os príncipais trabalhos pesquisados relacionados com o tema da dissertação.

Três grandes áreas relacionadas com o trabalho foram pesquisadas para se verificar o estado da arte e a partir disso verificar os pontos em aberto, questões não tratadas, implementações futuras ainda não realizadas e outros de forma que pudessem ser tratados no presente trabalho. O objetivo com isto é enriquecer ou colaborar com a comunidade científica com informações relevantes sobre o tema pesquisado.

A primeira área que diz respeito ao trabalho são as aplicações de sistemas embutidos na área biomédica. Serão apresentados trabalhos que implementam sistemas semelhantes ou com a mesma idéia da plataforma original do projeto previamente escolhida. Esta área nos auxiliará a entender melhor alguns tipos de sistemas de monitoramento remoto que existem hoje na literatura e também a extrair informações que podem ser aproveitadas para a implementação do trabalho. Além disso, os trabalhos pesquisados serão analisados e na medida do possível comparados com a implementação proposta.

A outra área pesquisada está relacionada com as implementações de pilhas TCP/IP para sistemas embutidos. Em decisão de projeto feita no capítulo 2 optou-se por fazer a comunicação do sistema de monitoramento remoto sem fio através do padrão 802.11b e para poder fazer com que o sistema seja integrado à internet é necessário a implementação ou desenvolvimento de uma pilha TCP/IP para a plataforma de hardware escolhida. Serão analisadas as pilhas TCP/IPs existentes na literatura relacionadas com sistemas embutidos.

A ultima área pretende verificar técnicas utilizadas no gerenciamento de energia de sistemas embutidos. É necessário que a bateria que alimenta o sistema dure tempo suficiente para uma monitoração mais contínua do paciente. Para isto é necessário verificar quais são os métodos de hardware e software adotados para prolongar o uso de uma bateria.

3.1 Sistemas de monitoramento remoto para aplicações médicas

Este trabalho teve como base de pesquisa dois projetos desenvolvidos no DCC/UFMG. O primeiro deles foi o projeto monitor cardíaco [FLCJ03] descrito no capítulo 2. O segundo foi Monitor Paramétrico Vestível*(MMV)*[Con01] que utiliza o conceito de computadores vestíveis[Man98] e é capaz de fazer a supervisão do eletrocardiograma *ECG*, do nível de oxigenação do sangue (oximetria) e da pressão arterial não invasiva.

O *MMV* caracteriza-se por ser um sistema embutido que possui duas placas com processamento próprio, cada uma com o seu respectivo firmware. Uma delas é baseada em um microcontrolador de 8 bits e possui ainda 16 MBytes de RAM e 16 Mbytes de memória *Flash*. Ela executa a coleta, conversão e disponibilização dos sinais analógicos. A outra é baseada em um microprocessador AMD 586 133MHz de alto desempenho. Ela é controlada por uma versão do *Linux* conhecida como *Coyote* e é especializada no empacotamento e disponibilização das amostras digitais provenientes da primeira *CPU* para uma rede de computadores. Através da utilização do *linux* foi possível utilizar a pilha TCP/IP embutida neste sistema operacional. O projeto ainda possui uma interface de rede local sem fio através de um cartão PCMCIA no padrão *Wavelan*. Todos estes circuitos são alimentados por uma bateria recarregável com autonomia de 4 horas.

Através da exploração do conceito dos computadores vestíveis, a equipe de pesquisadores do Media Lab no MIT tem desenvolvido um conjunto de aplicações voltadas para a área médica denominadas de *Healthwear*[Pen04]. Como hardware básico para a maioria destas aplicações foi desenvolvido o MIThril 2003 [RD03]. Trata-se de um *PDA* Zaurus 5500 da Sharp que roda Linux. Este *PDA* possui uma porta serial RS232 que é ligado a um outro hardware denominado *Hoarder*[Ger03] que é baseada em um microcontrolador PIC 16F877 responsável por coletar informações de sensores capazes de monitorar o sinal *ECG*, transcondutância da pele (*Galvanic Skin Response - GSR*) e temperatura. O *Hoarder* possui uma interface Compact flash, operando apenas no modo memória, para armazenar os dados dos sensores e ainda um módulo de rádio FM half duplex e um relógio de tempo real (*Real Time Clock - RTC*). O PDA Zaurus do projeto MIThril 2003 possui uma interface Compact Flash capaz de interfacear com um cartão de rede sem fio no padrão 802.11.

O projeto *CodeBlue*[KL04, DM04] da Universidade de Harvard utiliza uma rede de sensores sem fios baseada no conhecido hardware *MICA*[xbo]. Este hardware fica acoplado ao paciente através de um sensor de eletrocardiograma[TRFFJW04] e os dados coletados por cada nó sem fio são enviados para um PDA iPAQ rodando Windows CE que recebe os sinais enviados e envia pra uma central médica ou através de uma aplicação permitindo que as informações sejam analisadas pelo próprio PDA de um médico que recebeu os sinais da rede de sensores. Devido à limitada largura de banda e capacidade computacional, o MICA não pode ser utilizado com protocolos baseados em Internet tais como TCP/IP e ARP.

Mais recentemente, com o advento da programação em Java voltada para dispositivos móveis celulares, aplicações tem sido desenvolvidas para a área médica. Em [Fer05] é

descrito um exemplo de aplicação que usa um arquivo com dados adquiridos a partir de um ECG e estes são enviados para um servidor por meio de uma conexão HTTP POST. A grande dificuldade neste caso é que ainda as operadoras de telefonia ainda não estão completamente adequadas para suportar tais tipos de aplicação e além do mais, cada conjunto de dados enviado pelo paciente é cobrado pela operadora.

3.2 Pilhas TCP/IP para Sistemas Embutidos

A implementação de pilha TCP/IP mais conhecida, difundida e documentada atualmente é a BSD da Universidade de Berkeley. Ela possui código aberto e rapidamente ganhou popularidade na comunidade científica. Ela foi desenvolvida para computadores pessoais ou estações de trabalho que em sua maioria usam arquitetura 32 bits.

Existem várias pilhas TCP/IPs implementadas para sistemas embutidos. A maioria delas são variações do código original de Berkeley. Estas pilhas variam de acordo com a arquitetura utilizada e tamanho do código. Estes dois fatores permitem que funcionalidades do padrão sejam implementadas ou não.

Algumas pilhas usam um modelo simplificado e fazem suposições ou restrições relacionados com o ambiente onde o sistema será instalado. As suposições mais comuns são que o sistema embutido irá comunicar com um computador ou outro sistema que irá rodar uma pilha TCP/IP completa e de acordo com os padrões. Implementações que usam este modelo irão até comunicar de certa forma com outros sistemas nesta condição mas não irão comunicar ou irão ter a performance prejudicada quando comunicarem com sistemas semelhantes.

3.2.1 Simplificações utilizadas nas implementações

Uma das simplificações utilizadas no projeto de pilhas TCP/IP é definir apenas uma aplicação na camada de aplicação. Na maioria dos casos esta aplicação é um servidor web[Bra02]. Desta forma consegue-se reduzir o trabalho e complexidade a serem executados pela pilha implementando apenas o que é necessário para a aplicação funcionar. Não são implementados por exemplo suporte para dado urgente e não é necessário implementar abertura de conexão uma vez que não é necessário abrir conexão com outros servidores.

A implementação da aplicação tipo servidor web é muito utilizada em sistemas embutidos porque possibilita monitorar à distância o status de funcionamento do sistema ou verificar o estado de sensores acoplados aos microcontroladores.

As menores implementações conhecidas em relação ao tamanho do código e memória RAM, conhecidas são muito particularizadas para servidores web e são caracterizadas por não armazenar nenhum estado da conexão. Nestas implementações, retransmissões não podem ser feitas porque nada se sabe sobre o estado das conexões ativas. Para atingir transferências confiáveis, é necessário confiar no servidor remoto que é responsável pelas retransmissões. Outra simplificação destas implementações é não manter a conexão ativa

por muito tempo, ou seja, a partir de uma requisição, atende-la rapidamente e fechar a conexão. Dentre estas implementações pode-se citar o servidor *Web* baseado no microcontrolador PIC[Ben02],

Em relação a interface de rede implementada para estas pilhas TCP/IPs é comum aproveitar a UART existente nos microcontroladores e adotar o protocolo SLIP para comunicação serial com outro servidor. Um exemplo de uma pilha que implementa o protocolo SLIP*(Serial in Line Internet Protocol)* é a descrita em [Shr]. O protocolo SLIP é fácil de ser implementado porque apenas insere códigos simples de caracteres para sinalizar o limite de um bloco de dados.

Outro protocolo ao nível de rede que vem sendo bastante utilizado em conjunto com pilhas TCP/IP para sistemas embutidos é o PPP *(Point to Point Protocol)*[Req94]. Ele vem ganhando espaço porque alguns provedores de serviço previnem o acesso do protocolo SLIP. O protocolo PPP também é atrativo para os provedores de acesso porque implementa um protocolo de autenticação *(PAP)* com ID e senha além de outros relacionados com compressão *(CCP)*, por exemplo. Também tem a vantagem de monitorar a qualidade da linha. Por outro lado, por possuir uma extensa especificação com os vários protocolos descritos, a sua implementação é mais complexa e também o tamanho de código é maior quando comparado com o protocolo SLIP.

Em conjunto com o protocolo SLIP, algumas pilhas TCP/IPs implementam a emulação de um modem através de comandos AT. Esta implementação se faz necessária porque a maioria dos PCs utilizam um modem para comunicar com um provedor de serviços de Internet. Desta forma o dispositivo embutido deve ser capaz de responder a comandos de modem AT. O que se faz então é emular no dispositivo embutido fazendo com que a aplicação, como por exemplo um *Browser*, pense estar comunicando com um modem.

Apesar de a maioria das pilhas TCP/IP utilizarem como aplicação um servidor WEB, existem alguns projetos que não seguem a mesma regra e implementam protocolos não muito convencionais tais como a pilha desenvolvida por [Loe99b] que utiliza PPP como protocolo de rede, UDP como protocolo de transporte e TFTP como protocolo de aplicação.

Outras aplicações tais como a pilha TCP/IP da Atmel [Ben03] não implementam certos mecanismos vitais do TCP com o objetivo de redução de código. Elas deixam de implementar mecanismos de controle de congestionamento que é usado para reduzir a taxa de envio de dados quando a rede está sobrecarregada. Não implementando esta funcionalidade, o dispositivo embutido funcionaria perfeitamente quando conectado a apenas um ponto de rede mas enfrentaria problemas se fosse conectado a mais servidores.

A pilha TCP/IP para o microcontrolador MSP430 da Texas Instruments [Dan04] usa outra simplificação onde é possível manipular apenas uma conexão em um dado momento. Esta simplificação contraria um dos príncipios de funcionalidade do protocolo TCP que é prover conectividade a príncipio ilimitada para todos os nós de uma rede. Outra simplificações desta pilha incluem ausência de cálculo do checksum nos dados de entrada, falta de suporte para as opções TCP e segurança e tipo de serviço(TOS) IP.

Uma boa implementação de pilhas TCP/IPs é a da Microchip [Raj02] . Ela utiliza o conceito de Multitarefa cooperativa onde existe mais de uma tarefa, sendo que cada uma executa o seu trabalho e retorna o seu controle para que a próxima tarefa possa

executar o trabalho dela. Com isto a implementação da pilha não fica tão amarrada a aplicação o que pode ser interessante em casos onde muitas aplicações diferentes rodam ao mesmo tempo. Ela comporta de 2 a 253 conexões que são limitadas pelo compilador e pelo microcontrolador usado. Cada socket de conexão consome 36 bytes de RAM.

A pilha CMX-Micronet, desenvolvida pela empresa CMX, foi implementada para uso em microcontroladores com pequena quantidade de memória de dados e de programa. Ela suporta a maioria dos protocolos de Internet e ainda 127 soquetes UDP ou TCP [Ead04].

A tabela 3.1 mostra a quantidade de memória RAM de alguns das principais famílias de microcontroladores de 8 bits utilizados na atualidade:

Modelo	**Fabricante**	**RAM(bytes)**
COP8SBR9	National	1024
PIC18F4515	Microchip	3968
MC68HC908AP64	Motorola/Freescale	2048
ATmega128	Atmel	4096
ADUc834	Analog Devices	2304

Tabela 3.1: Quantidade de memória RAM de alguns Microcontroladores de 8 bits

Percebe-se que em alguns microcontroladores fica quase impossível receber um segmento ethernet (1500 bytes) ou um segmento 802.11(2314 bytes). Por isso algumas implementações descartam um segmento se o tamanho dele for maior que determinado limite. Isto funciona em casos onde o tamanho dos dados em certos tipos de aplicações não seja grande. De qualquer forma, quando receber os dados, quem enviou, na teoria, não saberá desta restrição e tentará retransmitir o dado novamente gerando um maior fluxo na rede. Outro problema também seria que alguns frames poderiam não serem recebidos devido a esta limitação na recepção dos dados.

Outra restrição imposta por algumas implementações é não suportar fragmentação de pacotes IP. Novamente, se esta funcionalidade não é implementada corre-se o risco de caso algum pacote destinado ao dispositivo embutido estiver fragmentado, a aplicação não será capaz de interpreta-lo e com isso de forma semelhante ao parágrafo anterior pode causar aumento de tráfego por causa de retransmissões e em alguns casos até perda do pacote. Implementações TCP/IP que são capazes de reconstruir pacotes IP fragmentados tais como [kad] são grandes em termos de tamanho de código e necessidades de RAM que se tornam impraticáveis para sistemas de 8 bits.

Adam Dunkels foi talvez um dos pesquisadores que mais estudou sobre as pilhas TCP/IPs para arquiteturas de 8 bits[Dun03]. Ele implementou duas pilhas TCP/IPs com diferentes abordagens e direcionadas para sistemas diferenciados. A estas pilhas ele deu o nome de *lwIP (Lightweight IP)* e *uIP (micro IP)*.

A *lwIP* é uma implementação TCP/IP completa mas simplificada que inclui os protocolos IP, ICMP,UDP e TCP e é modular o suficiente para ser facilmente extendida

para outros protocolos. A lwIP suporta multiplas interfaces de rede e tem opções de configuração flexíveis o que a torna compatível com a maioria dos dispositivos. O código desta implementação em linguagem C é todo estruturado, o que permite organização e fácil entendimento.

A implementação *uIP* foi projetada para ter apenas o conjunto mínimo de características necessárias para uma pilha TCP/IP completa. Ela pode manipular apenas uma interface de rede e não implementa UDP, ficando desta forma focada nos protocolos IP, ICMP e TCP.

As implementações de Dunkels procuraram seguir todas as necessidades da RFC1122 [Req89] que afetam a comunicação host para host (*host-to-host communication*), exceto certos mecanismos de interface entre a aplicação e a pilha tais como o report de erros e bits de tipo de serviço configurados dinamicamente no protocolo TCP. Uma vez que poucas aplicações fazem o uso destas características não implementadas, elas podem ser removidas sem comprometer o propósito da implementação.

As funcionalidades dos protocolos TCP/IP que ambas pilhas projetadas por Dumkels implementam são mostradas na tabela 3.2.

Característica	**uIP**	**lwIP**
Checksum IP e TCP	X	X
Fragmentação/Reconstrução IP	X	X
Opções IP		
Múltiplas Interfaces		X
UDP		X
Multiplas conexões TCP	X	X
Opções TCP	X	X
MSS variável TCP	X	X
Estimação de RTT	X	X
Controle de Fluxo TCP	X	X
Janela deslizante		X
Controle de Congestionamento	Não necessário	X
Dados TCP fora de sequência		X
Dado urgente TCP	X	X
Dado armazenado para retransmissão		X

Tabela 3.2: Funcionalidades Implementadas nas pilhas TCP/IP desenvolvidas em [Dun03]

3.2.2 Gerenciamento de memória

Devido as arquiteturas de 8 bits terem como fator limitante, quantidade de memória RAM conforme mostrado anteriormente na tabela 3.1 geralmente algumas soluções para gerenciamento da memória são adotadas.

A implementação *lwIP*, por exemplo, possui um buffer dinâmico e um mecanismo de alocação de memória onde a memória para armazenar o estado da conexão e os pacotes são dinamicamente alocados de um grupo de blocos de memória disponíveis. Pacotes são armazenados em um ou mais buffers de tamanho fixo alocados dinamicamente.

Na recepção dos dados, o driver do dispositivo de rede aloca buffers quando um pacote chega. Se o pacote recebido é maior que um buffer mais buffers são alocados e o pacote é dividido entre os buffers.

Na transmissão dos dados, a aplicação passa o comprimento e um ponteiro do dado para a pilha e também um flag dizendo se o dado é volátil ou não. A pilha TCP/IP então aloca buffers de tamanho apropriado ao tamanho e dependendo do flag de volatilidade, copia o dado para os buffers ou referências ao dado através dos ponteiros. Os buffers alocados para a pilha podem alocar além dos dados, os cabeçalhos TCP/IP e camada de rede. Depois que os cabeçalhos são escritos, a pilha passa os buffers para o driver do dispositivo de rede. Os buffers não são desalocados quando o driver do dispositivo termina de enviar os dados ficando estes em uma fila de retransmissão. Se os dados são perdidos na rede e tem que ser retransmitidos, os buffers da fila de retransmissão são retransmitidos. Os buffers são desalocados quando os dados forem recebidos pelo outro ponto ou quando a conexão é abortada seja pela aplicação local ou remota.

A implementação *uIP* não usa alocação dinâmica de memória. Ela usa um único buffer global denominado *uip_buf* para armazenar os pacotes e tem uma tabela fixa para armazenar o estado da conexão. O buffer global de pacotes é grande o suficiente para armazenar um pacote de tamanho máximo.

Quando o pacote chega da rede, o driver do dispositivo o coloca no buffer global e chama a pilha TCP/IP. Uma vez que o buffer será sobrescrito pelo próximo pacote de entrada, a aplicação deve processar rapidamente o dado ou copia-lo para um buffer secundário para posterior processamento. O buffer de pacotes não será sobre escrito por novos pacotes antes que a aplicação tenha processado o dado. Pacotes que chegam quando a aplicação está processando o dado devem ser enfileiradas pelo dispositivo de rede ou pelo driver do dispositivo. Se o buffer global estiver cheio, o pacote recebido é descartado. Isto causa degradação de performance mas apenas quando multiplas conexões estão rodando em paralelo. Isto é porque a implementação *uIP* anuncia uma pequena janela para receber os dados o que significa que apenas um único segmento segmento TCP estará na rede por conexão.

Na transmissão dos dados, o buffer global também é usado. Para enviar o dado, a aplicação passa um ponteiro para o dado e também o tamanho do dado para a pilha. Os cabeçalhos TCP/IP são escritos no buffer global e juntamente com o dado a ser enviado são transmitidos pela rede pelo driver do dispositivo. O dado não é enfileirado para retransmissão e neste caso a aplicação deve regerar o dado novamente se a retransmissão for

necessária.

A pilha criada por Jeremy Bentham[Ben02] restringiu ainda mais o uso de mémória RAM da pilha TCP/IP. O microcontrolador utilizado por ele, o PIC 16C76, possuia pouca memória RAM e foi necessário decodificar os cabeçalhos TCP/IP em tempo de execução quando recebidos, e prepara-los também em tempo de execução quando transmitido. Para armazenar os dados são utilizados dois buffers sendo um de transmissão e outro de recepção. A vantagem desta abordagem é de que os buffers serem independentes, ou seja, dados de transmissão e recepção podem ser tratados diferentemente e não é necessário esperar receber um pacote de recepção para enviar um de transmissão. Em compensação eles acabam dividindo o espaço total disponível para buffers fazendo com que capacidade do buffer de entrada e saída sejam menores do que a de um buffer global. Consequentemente, para microcontroladores com pouca memória a capacidade para receber um pacote de dados de tamanho máximo fica consideravelmente limitada.

Seguindo a mesma idéia da solução anterior, a pilha da Texas[Dan04] utiliza 3 buffers sendo um buffer para recepção e dois buffers para transmissão. Dos dois buffers para transmissão, um deles armazena os dados a serem enviados e todos os cabeçalhos necessários (TCP, IP e ethernet) e o outro além de armazenar o mesmo conteúdo anterior também armazena quadros dos protocolos ARP e ICMP. Embora possuindo 3 tipos de buffers apenas um buffer de transmissão e um de recepção podem ser mantidos ao mesmo tempo. Além disso, a pilha precisa esperar por um reconhecimento (ACK) do outro lado TCP antes de sobrescrever o conteúdo do buffer e permitir que novos dados sejam trocados.

Em alguns casos quando é possível, algumas implementações em microcontroladores utilizam a memória RAM de outros dispositivos tais como o buffer de dados do controlador de rede (NIC) para ajudar no armazenamento do cálculo do checksum, por exemplo.

3.2.3 Implementações de pilhas TCP/IP em Hardware

Algumas empresas implementam as soluções de uma pilha TCP/IP no próprio hardware. O primeiro Chip que apareceu com esta característica foi o Wiznet w3100A[Can01, Can02]. Ele possui integrado uma pilha TCP/IP que suporta também ICMP, UDP e ARP além de 24KB de memória RAM e um controlador de acesso ao meio (MAC) para interface de rede ethernet. Uma das vantagem deste tipo de abordagem é que a solução está pronta e na maioria das vezes fácil de ser utilizada pois algumas utilizam acesso às funcionalidades da pilha por meio de API´s.

Outra solução de pilha integrada ao hardware é o chip DS80C400[Max04]. Ele é um chip derivado da família 8051 que suporta uma pilha TCP/IP e um MAC para ethernet 10/100. A pilha suporta 32 conexões TCP simultâneas e pode fazer transferências de até 5MBps através do MAC ethernet. Além dos protocolos TCP, suporta o protocolo IP em suas duas versões(IPv4 e IPv6) e também UDP, DHCP, ICMP e IGMP.

Uma outra vantagem das pilhas TCP/IPs implementadas em hardware é que o trabalho de decodificação dos pacotes de dados fica para o chip que implementa a pilha enquanto que o microcontrolador fica somente encarregado de processar os dados ou implementar

protocolos para a aplicação final. Este é o caso da aplicação descrita em [RR00]. Ela utiliza um microcontrolador PIC 16F877 para processar os dados providos pela pilha implementada pelo chip S-7600A da Seiko Instruments. Ele integra uma pilha TCP/IP, 10Kbytes de RAM, interface com microcontrolador e uma UART em um único chip. Uma vez configurado, este chip age como se fosse um buffer de dados. Dados a serem transimitidos, até 1024 bytes, são armazenados em um buffer interno de RAM e a pilha TCP/IP acrescenta os cabeçalhos e checksums. Ela transmite o pacote pela UART. Quando o pacote é recebido, a pilha verifica se o endereço IP e a porta casam com o que foi configurado, calcula e verifica o checksum e transfere o conteúdo de dado do pacote para o buffer. Depois, o chip avisa ao microcontrolador que o pacote chegou e pode ser processado através de linhas de interrupção. A utilização de interface UART como meio físico sugere a conexão de um modem, que na aplicação é feita com uma solução integrada pelos chips Si2400/Si3015 da Silicon Laboratories.

As desvantagens de se ter as implementações da pilha em hardware é a falta de flexibilidade caso alguma mudança ocorra no padrão ou seja necessário a inclusão de certas funcionalidades. Outro ponto a se destacar é que pode-se ficar preso a um ou outro fabricante no projeto de um sistema embutido, uma vez que as implementações das pilhas TCP/IPs são particulares de cada fabricante de chip.

Uma previsão feita em [Can01] diz que assim como quase todo microcontrolador hoje em dia possui embutido uma UART é bem provável que no futuro, os chips já venham com uma pilha TCP/IP embutida. Isto porque assim como outros protocolos, a tendência é ter o projeto do protocolo TCP/IP congelado podendo com isso ser integrado em uma pastilha de silício.

3.2.4 Tamanho de Código

O tamanho dos códigos implementados pelas pilhas TCP/IPs dependem de vários fatores. Dentre estes fatores pode-se citar o compilador utilizado para geração do código. Outro fator é também a quantidade de protocolos utilizados. Algumas implementações não incluem o protocolo UDP ou o protocolo ARP.

A solução implementada pela Texas Instruments[Dan04] ocupa 4,2KB de memória de programa, 100bytes EEPROM (constantes) e 700 bytes RAM. A implementação de [Loe99a] não incluiu o protocolo TCP por causa da necessidade de ROM e RAM. Esta implementação gastou 145 bytes de RAM e 2170 palavras de ROM.

A implementação da Microchip[Raj02] mostrada na tabela 3.3 dá uma idéia do tamanho do código utilizado para alguns protocolos. Já a implementação de [Bra02] possui tamanho de código para alguns protocolos de acordo com a tabela 3.4.

A tabela 3.5 mostra o tamanho de código e quantidade de memória RAM usada pela implementação *uIP*[Dum03] O código foi compilado para um microcontrolador Atmel de 8 bits com arquitetura AVR usando o compilador *gcc* versão 3.3 com a opção de otimização de código ligada.

O total de memória RAM usada pode depender de quantas conexões TCP são alocadas,

Módulo	**Memória de Programa(KBytes)**	**Memória de Dados(Bytes)**
MAC	1,8	5(1)
SLIP	1,56	12(2)
ARP	0,78	0
IP	0,79	2
ICMP	0,64	0
TCP	6,6	42
HTTP	2,9	10
Servidor FTP	2,1	35

Tabela 3.3: Tamanho do código para cada implementação conforme [Raj02]

Módulo	**Memória de Programa(KBytes)**
MAC	1,0
ARP	2,5
TCP/IP	9,5
HTTP	3,8
UDP	1,4

Tabela 3.4: Tamanho do código para cada implementação conforme [Bra02]

quantas entradas são alocadas na tabela ARP e do tamanho do buffer de pacotes. Para a pilha *uIP* por exemplo, cada porta TCP em espera por conexão usa 2 bytes de RAM conforme mostrado na tabela 3.5. Um exemplo de configuração com 1 conexão TCP em estado de escuta, 10 conexões TCP em processamento, 10 entradas na tabela ARP e um tamanho de pacote de 400 bytes em uma aplicação simples de servidor HTTP irá ter as quantidades de memória mostradas na tabela 3.6 considerando o mesmo microcontrolador e compilador usados na tabela 3.5.

Para fazer uso de pouca quantidade de memória RAM, a pilha *uIP* também usa pouca memória alocada para chamadas de função que ficam armazenadas na pilha. Existe pouca profundidade de código entre a função principal *main* e as funções de aplicação. Além disto a maioria das funções da pilha *uIP* que são usadas pelos programas de aplicação são implementadas como macros da linguagem *C* e desta forma não usam memória alocada para pilha.

De acordo com o mostrado e também conforme estudado para outras pilhas TCP/IPs, o tamanho médio das implementações é em torno de 10 a 20Kbytes de ROM e 1Kbyte

Módulo	**Tamando do código**	**RAM estática**	**RAM Dinâmica**
Buffer de pacotes	0	100-1500	0
IP/ICMP/TCP	3304	10	35
Conexões TCP	646	2	0
UDP	720	0	8
Web server	994	0	11
Checksums	636	0	0
ARP	1324	8	11
Total	7624	1520	65

Tabela 3.5: Tamanho do código e uso da RAM em bytes para a pilha uIP no microcontrolador Atmel com plataforma AVR

Módulo	**Tamanho do código**	**RAM**
ARP	1324	118
IP/ICMP/TCP	3304	360
HTTP	994	110
Checksum	636	0
Buffer de pacotes	0	400
Total	6258	400

Tabela 3.6: Exemplo de Tamanho do código(Kbytes) para a implementação uIP

de RAM. Desta forma microcontroladores de 8 bits com tamanho de memória ROM de 32Kbytes e 2K de RAM são suficientes para rodar o protocolo TCP/IP. Microcontroladores com mais capacidade de memória, logicamente, poderão rodar além da pilha, outras implementações de protocolos relacionados e também outras aplicações não relacionadas com a pilha tais como um sistema operacional, por exemplo.

Existem outras implementações de pilhas TCP/IP que são vendidas no mercado conforme mostrado na coletânea de soluções descrita em [Pea05] mas todas elas são implementadas para microcontroladores de 16 bits ou maiores o que foge ao escopo deste trabalho.

3.2.5 Implementações TCP/IP para redes sem fio 802.11 em Sistemas embutidos

O desenvolvimento de um circuito para comunicação pelo protocolo 802.11 é muito complexo. O uso de uma frequência de rádio na ordem de 2,4GHz requer atenção especial devido as preocupações com interferência, ruído, compatibilidade eletromagnética e outros. Além disto o projeto do circuito para o nível físico e nível de interface de rede precisa incorporar todas as características discutidas no capítulo anterior e outras que não foram detalhadas. O desenvolvimento de um protótipo de rádio para comunicação com o protocolo 802.11 fica inviável dentro das necessidades exigidas.

Para suprir esta carência dos circuitos, a industria de semicondutores criou *chipsets* que agrupam circuitos que implementam ambos os níves físico e de interface de rede. Estes *chipsets* tem sido utilizados pela industria de fabricantes de suprimentos para redes sem fio 802.11 que desenvolvem os circuitos utilizando estes chips provendo interfaces de comunicação padrões para o mercado de computadores pessoais, notebooks e PDAs. As principais interfaces de comunicação para os adaptadores, produzidas pelos fabricantes de suprimentos de rede são nos padrões PCI, USB, PCMCIA e Compact Flash. Em [Tou04] são citados os principais fabricantes, tipos de chipsets e também fabricantes de equipamentos para redes 802.11. Segundo este artigo um dos principais *chipsets* utilizados pelos fabricantes é o *Prism*.

De forma a evitar o desenvolvimento complexo de um sistema de rádio compatível com o padrão 802.11b uma alternativa encontrada por alguns projetistas é a utilização de soluções baseadas em adaptadores de rede prontos. Pensando-se em quantidade de pinos alocados para a interface e lembrando que para um microcontrolador de oito bits isto deve ser plenamente levado em consideração a opção mais adequada neste caso seria a utilização da interface serial USB que necessita apenas de 2 pinos de I/O para comunicação. O problema da interface USB é que hoje quase não existem microcontroladores com interface USB embutida e por isso a interface requer mais um chip de interface o que foge do objetivo do trabalho onde o propósito é a utilização de apenas o microcontrolador sem grande modificação do hardware inicial para fazer o trabalho da pilha TCP/IP e controle da interface de rede.

Ainda pensando-se em quantidade de pinos, a interface com menos pinos depois da USB é a interface Compact Flash [Com03] que possui 50 pinos incluindo os pinos de alimentação. Através da implementação do modo de memória e de I/O, necessários para o funcionamento de interfaces de rede 802.11, pode-se implementar uma interface do microcontrolador com o padrão Compact Flash utilizando cerca de no mínimo 28 pinos. Com esta quantidade de pinos seriam necessários microcontroladores que geralmente possuem 4 portas de 8 bits ou 32 pinos.

Na implementação descrita em [Ead05] foi elaborado uma placa que comporta uma interface de rede sem fio 802.11b no padrão Compact Flash utilizando 28 pinos de um microcontrolador. Para este projeto foi utilizado um cartão Compact Flash com o *chipset Prism*.

Outra aplicação interessante que também utiliza o *chipset Prism* é a de um sistema que implementa um coletor de dados que armazena a radiação da luz solar[Cyl04]. Pelo propósito do projeto é necessário que o coletor solar fique em cima do teto de uma casa e portanto o projetista decidiu que o projeto fosse alimentado por luz solar. Para não ter a necessidade de ficar com fios saindo da janela e indo para o teto foi utilizado rede sem fio no padrão 802.11b para envio dos dados coletados para o computador do projetista.

O projeto do coletor solar teve como base um microcontrolador RCM3400 da Rabbit que possui uma pilha TCP/IP fornecida pelo fabricante e também foi implementado um driver de dispositivo para ser utilizado pela pilha. Como interface de rede sem fio foi utilizado um cartão Compact Flash no padrão 802.11b do fabricante Linksys. Devido ao alto consumo dos transmissores WI-FI, na camada de aplicação foi utilizado o protocolo SMTP para envio de emails com os dados coletados durante um dia.

3.3 O Gerenciamento de energia em sistemas embutidos

A utilização do protocolo de rede sem fio 802.11b através de um cartão Compact Flash para o desenvolvimento dos trabalhos com o sistema de monitoramento remoto consome cerca de 300mA quando está transmitindo [Cyl04]. Sendo assim é necessário a implementação de técnicas para gerenciamento de energia para redução do consumo de energia no projeto Monitor Cardíaco. Estudos mostram que com o gerencimento de energia é possível economizar até 50% de energia [YHL00].

Na literatura muito tem-se pesquisado em relação a sistemas de gerenciamento de energia para sistemas embutidos principalmente porque a maioria deles opera com alimentação proveniente de baterias. Em relação ao hardware, cada vez mais tem sido pesquisados e testados combinações de elementos químicos que possam prolongar a vida útil de uma bateria. Uma procura grande por circuitos de baixo consumo também tem sido observada.

Como o circuito de monitoramento remoto se encontra com o hardware projetado o que nos resta a fazer em relação ao hardware são escolher baterias com boa vida útil no que diz respeito ao hardware e usufruir dos recursos e características de software do microcontrolador para se implementar um sistema de gerenciamento de energia.

Os sistemas de gerenciamento de energia podem ser classificados em estáticos e dinâmicos [YHL01]. Os sistemas estáticos de gerenciamento de energia são na maioria das vezes aplicados durante o desenvolvimento do sistema enquanto que os sistemas dinâmicos são aplicados em tempo real enquanto os sistemas estão com pouca carga de trabalho ou estão ociosos.

Os sistemas de gerenciamento de energia possuem diferentes estados de operação denominados de acordo com o seu consumo de energia. De maneira geral pode-se classificar os sistemas em relação ao seu consumo de energia de acordo com o seguinte:

1. **Ativo** - estado onde o consumo de energia é maior e o sistema opera a plena carga;

2. **Stand By** - o sistema opera com algumas funcionalidades principais tais como timers e interrupções mas não consome tanto. Possui funcionalidade limitada;

3. **Power Down ou modo Sleep** - o sistema consome pouca energia e geralmente possui somente o relógio de tempo real funcionando. Ele só é "acordado" por meio de interrupção ou reset por exemplo.

A maioria das técnicas estudadas dizem respeito ao tipo de política de gerenciamento de energia que é adotada. A política de gerenciamento é a lei de controle adotada pelo gerenciador de energia para decidir sobre o estado de operação do sistema [uBM99]. O próprio padrão 802.11 possui uma política de gerenciamento de energia embutida no protocolo. Este padrão requer que o ponto de acesso *AP* envie um sinalizador (*beacon*) a cada 100ms seguido de um mapa de indicação de tráfego (*TIM*). Cada cartão que desejar comunicar precisa ativamente escutar pelo sinalizador para saber se existe algum dado destinado a ele. Se não é necessário transmitir nem receber então o cartão pode ir para o estado de baixo consumo (*Doze*) até o próximo sinalizador.

Em [SVGM00] foram combinados novos algoritmos de gerenciamento e controle de energia para reduzir o consumo de um cartão de rede sem fio. O gerenciamento de energia reduz o consumo seletivamente colocando o cartão em estados de menor consumo quando o usuário não está ativamente comunicando com o cartão. O Controle de energia reduz o nível que o cartão transmite enquanto mantém a mesma performance. Desta forma este trabalho procurou reduzir o consumo de potência nos modos de operação ativo e ocioso do cartão.

Outra técnica também utilizada para reduzir o consumo dos dispositivos embutidos é a denominada de *DVS (Dynamic Voltage Scaling)* [SBA+01]. Esta técnica consiste em mudar a velocidade e a tensão de alimentação do processador durante o tempo de execução do programa de acordo com as necessidades da aplicação que está rodando. Para a implementação desta técnica é necessário que o processador ou microcontrolador suporte mudanças de velocidade e tensão de alimentação em tempo real. Em [IM02] são apresentados vários conceitos ligados ao gerenciamento dinâmico de energia incluindo a técnica *DVS* e também são citados alguns exemplos de aplicação baseados em diferentes políticas de gerenciamento.

Quando o tempo de transição entre os estados de operação é instantâneo, a política de gerenciamento de energia é trivial: os recursos devem ser desligados assim que se tornarem ociosos. Em alguns casos, a transição para o estado de baixo consumo é cara em termos de tempo e energia. No caso dos discos rígidos (HDD's), por exemplo, transições do estado *sleep* para o estado ativo são um processo lento e de alto consumo uma vez que o disco precisa ser acelerado até uma alta velocidade. Quando estas transições são lentas, o problema da otimização da política de gerenciamento de energia torna-se não trivial e políticas efetivas que miniminizam o consumo sem comprometer a performance são necessárias.

Na aplicação desenvolvida em [Cyl04] foi implementado um circuito para desligar a alimentação do cartão Compact Flash de uma interface de rede sem fio 802.11b quando esta não estivesse sendo utilizada. A implementação do software otimizou a economia

de energia colocando o microcontrolador em modo *Sleep* somente permitindo a coleta de dados do conversor Analógico-Digital interno. Quando é desejado o envio de email com os dados, o software verifica se existe energia suficiente para a comunicação WI-FI. Caso exista, a alimentação da interface Compact flash é ligada, o cartão é resetado e configurado para envio do email. Através de todas estas técnicas foi possível permitir que a aplicação conseguisse funcionar de maneira autônoma com a energia para alimentação do circuito proveniente apenas de paineis solares e bateria.

Capítulo 4

Implementação do Sistema de Monitoramento Remoto de Pacientes

Este capítulo apresenta as implementações que foram necessárias para que o hardware do monitor cardíaco fosse transformado para atender os pré requisitos de projeto. Estas tranformações compreenderam algumas mudanças físicas no hardware do projeto. Em relação ao software foi desenvolvido um driver de dispositivo (*Device Driver*) para o cartão Compact Flash que implementa uma interface de rede sem fio no padrão 802.11b, uma pilha TCP/IP apropriada para o microcontrolador utilizado foi adaptada e aplicações para esta pilha foram também adaptadas para atender ao requisito de comunicação com a internet. Em relação ao gerenciamento de energia, técnicas de software são mostradas e implementadas para garantir que o sistema funcione com maior autonomia permitindo maior continuidade na coleta dos dados.

4.1 Interface com o cartão Compact Flash - Modo Memória

4.1.1 Leitura e escrita em um cartão Compact Flash

O trabalho de implementação da pilha TCP/IP com interface de rede sem fio originava na validação do projeto inicial em relação a interface Compact Flash. Era necessário saber se o circuito projetado era capaz de se comunicar com algum cartão no padrão Compact Flash. Desta forma, os trabalhos se concentraram inicialmente na familiarização com o padrão Compact Flash. Isto foi feito através de um estudo aprofundado sobre este padrão sobretudo no que diz respeito a leitura e escrita em um cartão Compact Flash.

Para a leitura e escrita no cartão Compact Flash, deve-se considerar os mapeamentos existentes para o tipo de dispositivo implementado no cartão. Basicamente, um cartão

Compact Flash possui os seguintes tipos de memória:

- Memória de atributos *(CIS)*;
- Memória Comum (Modo memória);
- I/O (Modo I/O).

Todo cartão Compact Flash possui uma estrutura de informação importante que fica armazenada nos primeiros bytes da chamada *memória de atributos*. Esta estrutura é denominada *CIS (Card Information Structure)* e possui dados que indicam qual é o tipo do cartão (memória ou I/O), dados do fabricante, quantidade de memória (modo memória) e para cartões no modo I/O com interface de rede sem fio indica por exemplo, qual é o endereço de acesso ao meio *MAC Address* do cartão. Esta estrutura tem 256 bytes de endereço.

A memória Comum existe para o caso de cartão no modo memória e o mapeamento de I/O é referente aos registradores existentes no cartão que implementam uma função de entrada e saída como é o caso do cartão de rede sem fio 802.11b.

O padrão Compact Flash suporta operação com um barramento de 8 ou 16 bits. Na tabela 4.1 simplificadamente é mostrado a configuração dos pinos necessária para operação em modo 8 bits para um cartão do tipo memória ou I/O. A temporização destes sinais é mostrada na figura 4.1. A temporização dos sinais para a leitura da estrutura de configuração *CIS* é a mesma do modo memória, porém o sinal REG, deve ser ativado em nível lógico zero ao invés de permanecer em nível lógico um.

Os tempos máximos e mínimos necessários para leitura e escrita foram omitidos na figura 4.1. Maiores informações sobre eles em ambos os modos podem ser obtidas em [Com03, Inc01].

4.1.2 Validação do projeto inicial - interface com o padrão Compact Flash

Após o entendimento do procedimento para leitura e escrita no padrão Compact Flash, decidiu-se validar o projeto Monitor Cardíaco para a leitura e escrita na memória Compact Flash. Para isto foram implementadas funções de leitura e escrita em um cartão de memória com capacidade de 8MB.

Como esta implementação ainda não tinha sido testada, no momento da tentativa de escrita detectou-se que o circuito da fonte da alimentação não suportava a corrente necessária para escrever no cartão de memória. Esta corrente elevada fazia com que a tensão de alimentação do circuito fosse interrompida provocando um reset no microcontrolador. Este problema foi inicialmente contornado ligando-se uma tensão de 5VDC direto na alimentação dos circuitos e desligando o circuito do conversor DC-DC da placa.

A implementação das funções de escrita e leitura no modo memória foram feitas com base no diagrama de sinais no tempo mostrados na figura 4.1 [Com03] para operações com

-CE2	-CE1	-REG	-OE	-WE	-IORD	-IOWR	Espaço selecionado
X	0	0	X	X	0	1	Leitura dos registradores de configuração (Modo IO)
X	0	0	0	1	X	X	Leitura dos registradores de configuração (Modo memória)
1	0	1	0	1	X	X	Leitura da memória comum (D7:D0)
X	0	0	1	0	1	0	Escrita nos registradores de configuração (Modo IO)
X	0	0	1	0	X	X	Escrita nos registradores de configuração (Modo memória)
1	0	1	1	1	X	X	Escrita na memória comum (D7:D0)
X	0	0	0	1	X	X	Leitura da estrutura de informação do cartão (CIS)
1	0	0	1	0	X	X	Acesso inválido (escrita na CIS)

Tabela 4.1: Tabela de configuração dos pinos do padrão Compact Flash para acesso aos diferentes modos para funcionamento em oito bits

o barramento de dados configurado para oito bits. Nesta figura também são mostrados os sinais *REG* e *CE2* que no projeto original estavam ambos fixos com os pinos em nível lógico um. Alguns pinos tais como *WAIT* e *INPACK* não foram considerados no projeto original principalmente pela escassez dos pinos do microcontrolador.

Foi implementada uma função de escrita de dados no modo memória através da função `CFWriteAdr`. É passado como parâmetro o endereço e o dado que se deseja escrever. Uma função de leitura foi implementada através da função `CFReadAddr` onde o endereço a ser lido é passado como parâmetro.

Ainda foram implementadas as seguintes funções para uma melhor utilização da interface no modo memória:

- `CFSetAdr(unsigned char adr)` - faz a máscara endereço da Compact Flash com porta do microcontrolador;
- `CFWaitBusy(void)` - espera enquanto sinal de ocupado (Busy) estiver ativo;
- `CFWaitReady(void)` - espera enquanto sinal de Pronto (Ready) estiver ativo;

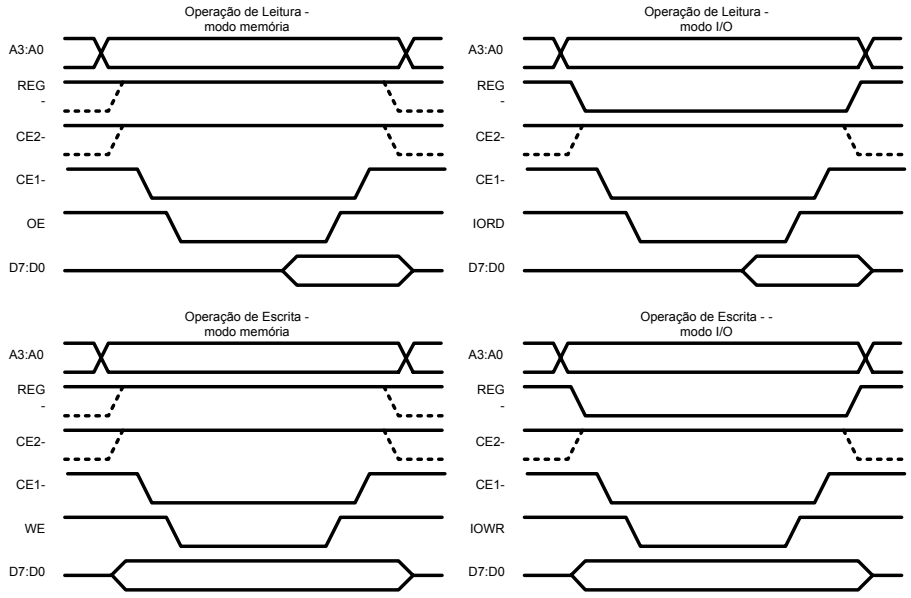


Figura 4.1: Diagrama de tempo para escrita e leitura no padrão Compact Flash no modo memoria e I/O

- `CFWriteSector(unsigned long lba, unsigned char *buf)` - escreve em um endereço `LBA` *Logical Block Address* um caractere apontado pelo ponteiro `buf`;
- `CFReadByteFromSector(unsigned long lba, unsigned char byte_to_read)` - Lê o byte `byte_to_read` do endereço `lba`.

Para a verificação do atendimento ao padrão de temporização do barramento e validação das funções escritas foi implementado um pequeno programa que escreve uma string na Compact flash em setores diferentes e depois lê o que foi gravado nestes setores. O programa foi testado e funcionou corretamente. Foi possível verificar o resultado do teste através da impressão dos dados lidos da Compact Flash na porta serial do microcontrolador ADUc834.

Este primeiro desenvolvimento permitiu que parte do circuito fosse validada principalmente a questão da interface do microcontrolador com a memória Compact Flash em relação aos sinais de escrita, leitura, dados e endereços. Outro aspecto relevante que também foi validado foi a questão da temporização dos sinais nas operações de escrita e leitura para o modo memória mostrados na figura 4.1.

4.2 Familiarização com a Pilha de Protocolos TCP/IP

Após a validação do projeto inicial em relação ao padrão Compact flash no modo memória partiu-se para uma familiarização com a pilha de protocolos TCP/IP. Como ainda não

existia uma interface de nível físico sem fio implementada, foi necessário a utilização de outro tipo de interface.

Para uma implementação rápida do protocolo foi adaptado para o projeto monitor cardíaco a pilha TCP/IP descrita em [Ben02]. Através do livro que acompanha esta implementação foi possível um rápido entendimento das necessidades e limitações do projeto. Seguindo os moldes clássicos de aplicações implementadas em sistemas embutidos foi adaptado um servidor web que quando requisitado enviava uma página com a hora local no circuito obtida por meio de um circuito de Relógio de Tempo Real (*RTC*) embutido no próprio microcontrolador. Para a interface de rede foi utilizado a USART do microcontrolador ADUc834 com a implementação do protocolo *SLIP*.

Com pouca memória para armazenamento dos pacotes de dados, a implementação conforme descrita no capítulo 3 decodifica um pacote de dados em tempo de execução. Possui um buffer de dados de entrada `rx_buff[]` que é preenchido a partir de uma interrupção de chegada de dados na porta serial. Para envio dos dados, preenche um único buffer de saída `txbuff[]`. Para esta aplicação utilizando apenas a memória interna de 256 bytes do ADUc834, os tamanhos dos buffers de entrada e saída foram setados respectivamente para 80 e 96 bytes. Como o tamanho dos cabeçalhos TCP e IP padrões é de 40 bytes sobram no máximo 40 bytes no caso de recepção e 56 bytes de transmissão para os dados das aplicações. Isto é suficiente para atender e responder uma requisição ICMP do tipo ping de até 40 bytes.

O Controle de entrada e saída de dados é feito através de sinalizadores (*Flags*) que se estiverem setados ativam rotinas para processamento de envio e recebimento de dados. O *Flag* de recepção é ativado durante a rotina de processamento de interrupção de entrada de dados gerada pela USART do microcontrolador. Já o *Flag* de transmissão é ativado toda vez que é necessário o envio de dados pela porta serial, seja ele um comando de modem que emula a resposta de um modem até um pacote de resposta a uma requisição de dados no padrão HTTP.

De forma semelhante, o controle de utilização dos buffers de entrada e saída são feitos simplificadamente com o uso de duas variáveis para cada um. Uma variável de entrada é incrementada toda vez que o buffer de entrada ou saída é preenchido com um byte de dados. Uma variável de saída é incrementada toda vez que o buffer de entrada ou saída processa um byte de dados. A título de ilustração de parte de funcionamento do programa, fluxograma de controle do programa detalhando a parte de processamento dos dados do pacote IP é mostrado na figura 4.2.

A aplicação implementada funcionava como um servidor web apenas. Com isto muitas simplificações foram feitas. Uma delas é relacionada com uma limitação de memória do projeto. Como não utiliza memória para armazenamento de pacotes ela é capaz de atender apenas a uma requisição por vez. Outro fator de simplificação foi a aceitação de uma abertura passiva e desta forma a pilha TCP/IP não implementava todos os estados, somente os referentes ao estado passivo. Desta forma não foi necessário armazenar nenhum endereço e nem número de porta, uma vez que era necessário apenas copia-los da mensagem de entrada para a mensagem de saída.

Devido ao fraco gerenciamento de conexão, retransmissões de dados não são possíveis.

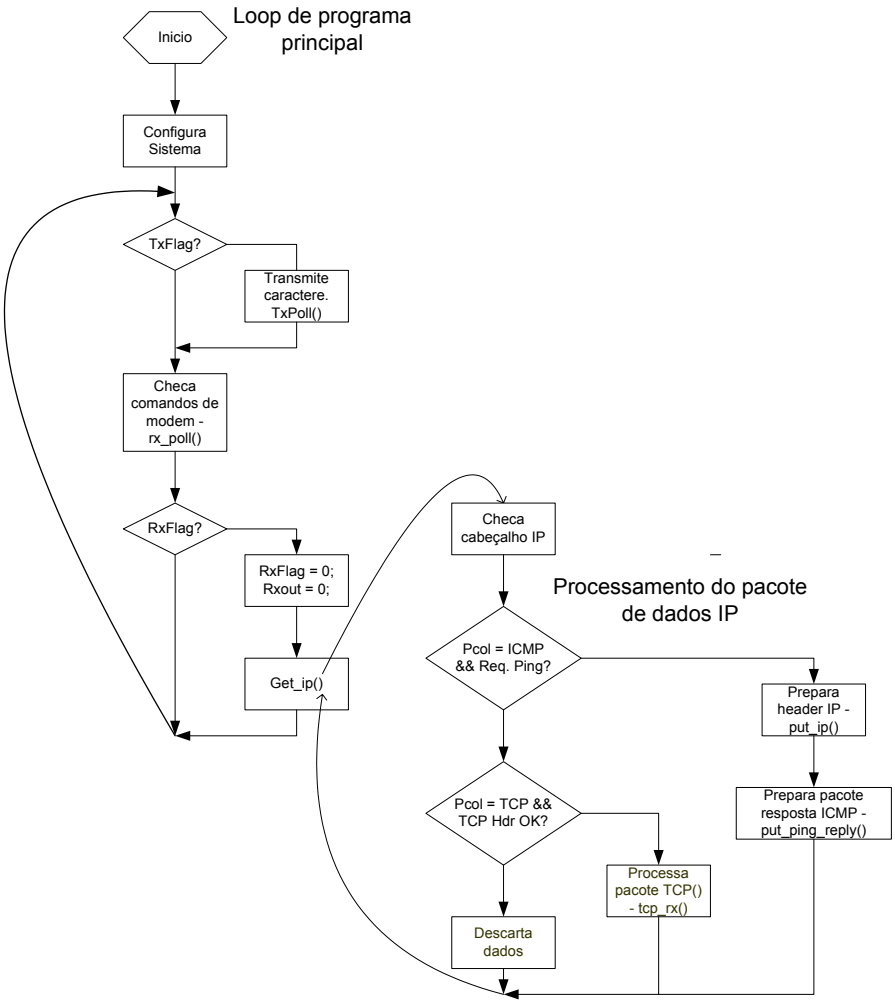


Figura 4.2: Fluxograma de parte do programa de controle da Pilha TCP/IP

Para contornar este problema são implementadas transferências de dados curtas que cabem apenas em uma transferência de dados (*One segment TCP data transfer*). Este tipo de transferência espera pela abertura de conexão e uma vez estabelecida, os dados são enviados de uma vez e o servidor fecha a conexão não deixando-a ativa. Caso desejar que os dados sejam variáveis é possível que apenas 56 bytes, de acordo com esta primeira implementação, sejam enviados o que torna a aplicação inviável, uma vez que são necessários que mais dados sejam enviados. A conexão sendo somente passiva impossibilita que o paciente envie dados remotamente quando desejar, não cumprindo com isto um dos requisitos iniciais do projeto.

Somando-se aos outros fatores anteriormente citados, por causa da memória limitada a aplicação HTTP teve que ficar fortemente acoplada à pilha TCP/IP de forma que a distinção entre elas ficou quase imperceptível o que tornou o programa mais difícil de ser entendido.

Uma das poucas vantagens desta implementação foi o tamanho de código para a pequena aplicação implementada. O código implementado ocupou 5,2Kb e pouco mais de 300 bytes de memória RAM, incluindo a memória estendida do microcontrolador ADUc834.

Até este momento cronológico do projeto, fora os pré requisitos atendidos com a escolha do hardware do monitor cardíaco, nenhum dos pré requisitos iniciais haviam sido cumpridos. As etapas realizadas até aqui serviram para solidificar o conhecimento e auxiliar no progresso com as etapas posteriores.

4.3 O Cartão Compact Flash 802.11b - Chipset Prism

A etapa anterior mostrou a possibilidade, embora com alguma limitação, de conectar o monitor cardíaco à internet por meio da pilha TCP/IP implementada. Como a implementação foi feita utilizando a porta serial como meio de comunicação era necessário implementar a interface de rede sem fio para satisfazer uma das necessidades de projeto.

Iniciou-se o desenvolvimento da interface de comunicação sem fio através da escolha do cartão Compact Flash que implementa uma interface de rede sem fio 802.11b. O modelo escolhido foi o *WCF12* do fabricante Linksys porque o DCC/UFMG já possuia tal cartão não havendo necessidade de compra. Uma vez escolhido o cartão, a próxima etapa foi descobrir que circuitos haviam dentro daquele cartão e o que era necessário implementar para possibilitar a comunicação com ele.

A pesquisa foi iniciada pela procura de drivers de dispositivo para a plataforma Linux. Existem grandes comunidades que desenvolvem vários drivers de dispositivo e muitas vezes é fácil achar os códigos prontos para a maioria dos dispositivos e quase sempre apoiados ou suportados pelo fabricante do Hardware. Foram encontradas comunidades [AS, pcma] de desenvolvedores com vários drivers de dispositivo desenvolvidos para o padrão PCMCIA.

Através de [Tou04] foi possível identificar que o fabricante Linksys utiliza em seus cartões Compact Flash e PCMCIA 802.11b, o Chipset PRISM originalmente desenvolvido pelo fabricante Harris/Intersil onde passou por várias gerações de desenvolvimento e no momento de publicação desta dissertação se encontrava na versão 3 fabricada pela Conexant.

O Chipset Prism surgiu em 1996 como uma solução altamente integrada de cinco chips que implementavam um modem RF com tecnologia de espalhamento de frequência por Sequência Direta (*Direct Sequence Spread Spectrum - DSSS*). Hoje em dia na terceira geração devido a alta integração dos chips, o chipset foi reduzido a apenas dois chips. Um destes chips que fazem parte do chipset PRISM, o *HFA384x*[Inc01], implementa o nível físico *PHY* e de controle de Acesso ao meio *MAC*. O *HFA384x* opera com o barramento Compact flash em 8 bits e em 16 bits.

Como o cartão Compact Flash de rede sem fio já estava implementado com todo o hardware necessário para o funcionamento do chipset Prism, os detalhes em relação ao funcionamento interno deste circuito não são relevantes para o projeto e portanto estão fora do escopo do trabalho. O único fator relevante relacionado com o chipset Prism é a

interface Compact Flash implementada pelo Chip *HFA384x* e o funcionamento deste chip para que a comunicação no padrão 802.11b seja implementada.

Através desta pesquisa inicial na procura pela forma de controle do cartão de rede sem fio pode-se ter uma idéia através dos drivers de dispositivos implementados no Linux do que seria necessário para a implementação da comunicação no projeto no monitor cardíaco. O primeiro obstáculo de se descobrir o que havia dentro do cartão Compact Flash e como se fazer a comunicação estava superado. Agora era necessário certificar se o cartão possuia tal chipset e posteriormente implementar um driver para este dispositivo.

4.4 Interface com o Cartão Compact Flash 802.11b

O chip *HFA384X* da família PRISM possui pinos compatíveis com a interface Compact Flash conforme mostrados a seguir. Nestes itens são indicados também as funções dos pinos e onde eles estavam originalmente conectados no projeto monitor cardíaco.

- `A9:A0`(I) - Linhas de endereço. `A9:A4` ligados originalmente a GND e `A3:A0` ligados ao ADUc834;
- `D7:D0`(I/O) - Linhas de dados. Todas ligadas originalmente ao ADUc834;
- `-CE1, -CE2`(I) - Habilitação do Cartão (16/8 bits). `-CE1` ligada ao ADUc834 e `-CE2` ligada a VCC
- `-OE(RD)`(I) - Habilitação de leitura de memória de atributos do cartão. Ligado ao ADUc834.
- `-WR`(I) - Habilitação de escrita de memória de atributos do cartão. Ligado ao ADUc834.
- `-IORD`(I) - Habilitação de leitura nos registradores de I/O do cartão. Ligado originalmente a VCC.
- `-IOWR`(I) - Habilitação de escrita nos registradores de I/O do cartão. Ligado a VCC.
- `-REG`(I) - Sinal de habilitação para leitura/escrita nos registradores do cartão no modo IO e de escrita na estrutura de informação do cartão (CIS). Ligado originalmente a VCC.
- `INPACK-`(O) - Sinal gerado pelo HFA384X para sinalizar que um ciclo de IO está em curso. Não conectado originalmente.
- `WAIT-`(O) - Gerado pelo HFA384X para sincronizar leitura e escrita com o microcontrolador. Não conectado.
- `IREQ-/READY`(I/O) - Durante o reset serve como saída para indicar que o cartão está pronto para operação após reset. Após reset serve como pino de interrupção. Não conectado.

- `RESET`(I) - Inicializa o cartão Compact Flash. Ligado a GND.

Através da comparação dos sinais de interface do *HFA384X* com os que existiam, percebeu-se que haviam vários sinais de controle necessários para interfacear com o chip PHY/MAC tais como IORD, IOWR, RESET que não estavam implementados originalmente no projeto monitor cardíaco. Além dos pinos de controle, mais tarde descobriu-se que também era necessário a alocação de mais pinos de endereço por causa do acesso a registradores de I/O do *HFA384X*. Não havia pinos disponíveis no microcontrolador ADUc834 que poderiam ser alocados para interfacear com os sinais necessários do chip *HFA384X*. O circuito de interface entre o ECG e o conversor AD já havia sido testado anteriormente pelo projetista e não era necessário testa-lo novamente.

Não foi encontrada uma outra solução, em tempo hábil, para o problema da falta de pinos para interfacear com o barramento Compact Flash do cartão. Em virtude disto, decidiu-se sacrificar alguns pinos do microcontrolador para a implementação da interface Compact Flash. Estes pinos primeiramente foram escolhidos de acordo com o critério de manutenção da funcionalidade original do circuito ECG. Mesmo assim, devido a necessidade inicial de uso de todos os pinos de endereços do microcontrolador, foi necessário sacrificar o restante do circuito compremetendo a funcionalidade do circuito ECG. Este problema é contornado através de solução proposta nos capítulos seguintes.

Procedeu-se então verificando-se que pinos teriam de ser realocados para serem ligados nos pinos do conector da Compact Flash(CF). Chegou-se na tabela 4.2. Nesta tabela também é indicada o nível de comprometimento da remoção do sinal para a funcionalidade do projeto. O pino CE1- foi colocado em nível GND, uma vez que o cartão Compact Flash 802.11b não possui memória comum.

Sinal Original	**Sinal CF alocado**	**Tipo sinal CF**	**Compromete funcionamento ECG?**
CE1-	A4	Endereço	Não
CS_ADC	A5	Endereço	Sim
TXD1	A6	Endereço	Não
DOUT	A7	Endereço	Sim
CLK	A8-A9	Endereço	Sim
CS_DSP	REG	Controle	Não
BUZ	IORD	Controle	Não
LED1	IOWR	Controle	Não
CS_ASIC	RESET	Controle	Sim

Tabela 4.2: Sinais originais alocados para os pinos da Compact Flash

Uma vez definidos os pinos a serem ligados, as trilhas do circuito impresso da placa montada do projeto monitor cardíaco foram cortadas e fios foram soldados ligando os pinos do microcontrolador no conector Compact Flash. O resultado da modificação do circuito pode ser observado na figura 4.3.

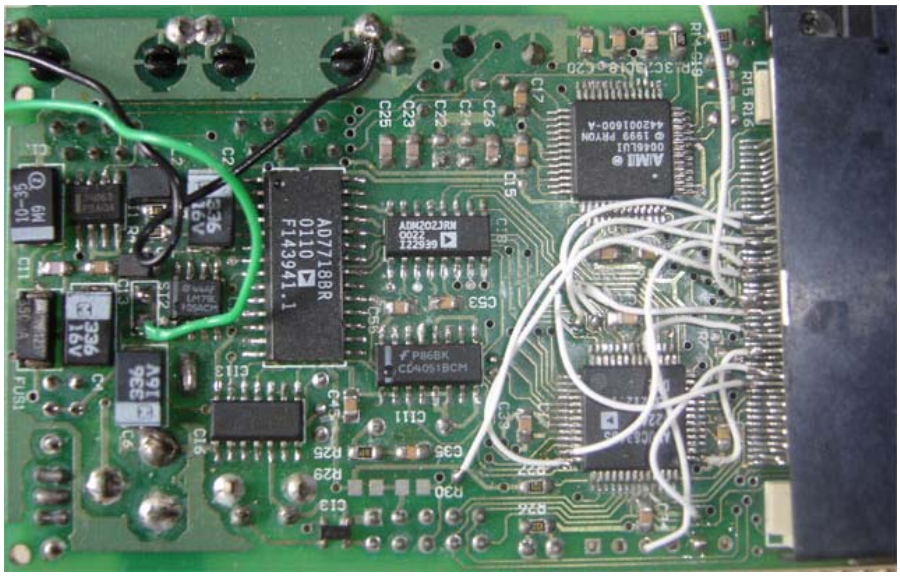


Figura 4.3: Circuito do Monitor Cardíaco modificado para operar junto com o cartão CF de rede sem fio 802.11b

Pode ser observado na figura 4.3 que a tensão de alimentação do cartão Compact Flash (fio branco saindo para fora da placa) teve de ser alterada para 3,3V, uma vez que o cartão suporta somente tal tensão.

4.5 Documentação para desenvolvimento do driver de Dispositivo de rede sem Fio 802.11b

Com a modificação do hardware do projeto era necessário valida-lo para verificar a geração dos sinais de controle e endereços para lerem ou escreverem os dados que se faziam necessários para controle da interface de rede sem fio.

O driver para o cartão de rede sem fio desenvolvido para o sistema operacional Linux [AS] serviu como ponto de partida para saber que funções eram necessárias para implementação do driver para o projeto monitor cardíaco. Ainda assim era necessário algum documento que descrevesse melhor o funcionamento do chipset PRISM de forma que se pudesse desenvolver o software de controle.

Em pesquisas realizadas nos sítios de Internet dos fabricantes foi impossível descobrir alguma referência a alguma informação mais detalhada sobre o funcionamento do chipset PRISM. Depois de muita pesquisa foi descoberta a documentação para o desenvolvimento do Driver para o dispositivo: um manual [Inc02] onde é explicado como usar os serviços fornecidos pelos chips de controle de acesso ao meio *HFA384X*. Tal documentação é fornecida pelo fabricante com acordo de não divulgação (*NDA - Non Disclosure Agreement*).

A seguir são fornecidas algumas características de funcionamento do chip de MAC PRISM para que fosse possível o entendimento das funções necessárias para controle do cartão CF de rede sem fio.

4.6 Características gerais do MAC PRISM

O Controlador de Acesso ao Meio *MAC* do chipset PRISM fornece um conjunto de serviços para o microcontrolador ou para o software do driver de dispositivo incluindo:

- Envio de pacotes de dados formatados em frames 802.3 (Ethernet) e 802.11 (Wi-Fi);
- Recebimento de pacotes de dados formatados em frames 802.3 e 802.11;
- Reporta mudança de estado de conexão tais como associação, autenticação e disassociação;
- Configuração de vários parâmetros de operação de um adaptador de rede sem fio 802.11.

O microcontrolador pode operar o MAC PRISM através de um conjunto de comandos que são configurados através de um registrador de comandos. Para cada comando lançado, o MAC PRISM indica o resultado do comando através de eventos que são setados no registrador de status de eventos. A interface do MAC PRISM trava a interface de controle não deixando que outros comandos sejam executados enquanto um conjunto de bits de ocupado (*Busy bits*) associados com o registrador de comandos e através de outros comandos e registradores de parâmetros estiverem ativos.

O MAC PRISM é projetado para mover e processar dados em blocos que são chamados de *Buffers*. Estes *buffers* ficam localizados em uma memória de uso específico existente no cartão de interface rede sem fio. O microcontrolador acessa os *Buffers* indiretamente através de chamados Caminhos de Acesso para Buffers(*Buffer Access Paths - BAPs*) que são registradores que permitem endereçamento, leitura e escrita em áreas de memória para os *Buffers* chamadas de Identificadores de Quadro (*Frame Identifiers- FIDs*).

O microcontrolador gerencia e configura os dados por meio do mesmo mecanismo dos *BAPs* usando valores especiais chamados de Identificadores de Recursos (*RIDs*) que podem ser lidos ou escritos em um buffer de uso especial. A escrita de um *RID* em um *BAP* copia

a informação de configuração da memória do MAC PRISM para um buffer especial que pode ser acessado pelo microcontrolador para leitura ou escrita.

O MAC PRISM é implementado usando um controlador proprietário projetado para processamento de dados e execução do protocolo 802.11 através de um hardware dedicado para processamento de eventos e gerenciamento de memória. Cada máquina de estado do protocolo de MAC 802.11, máquina de estado para controle da camada física e a resposta aos comandos do microcontrolador é operado pelo *firmware* interno do chip. Este *firmware* pode ser configurado para operar tanto como uma *Estação* ou como um *Ponto de Acesso*. No caso do cartão Compact Flash, o *firmware* foi configurado para operar com o modo de operação *estação*.

4.7 Implementação de funções para controle do cartão de rede sem fio

Para a operação do MAC PRISM como cartão de estação de rede sem fio, são necessárias as seguintes principais tarefas:

1. Inicialização do cartão e do firmware;
2. Alocação de buffers para transmissão;
3. Uso da interface RID, registros de configuração e frames de informação para gerenciamento de operação do cartão;
4. Habilitação da recepção de dados e alocação de eventos de recepção;
5. Manipulação de eventos de transmissão.

Em resumo foram criadas três funções principais para controle do dispositivo de rede sem fio. A primeira função denominada `wifidev_init()` executa o que podemos chamar de inicialização geral do cartão englobando as três primeiras tarefas citadas acima além da habilitação da recepção de dados. A função `wifidev_read()` faz o tratamento dos dados recebidos e a função `wifidev_send()` manipula os dados a serem transmitidos pelo cartão.

4.7.1 Inicialização Geral do Cartão

A inicialização do hardware e firmware do cartão consiste em uma série de operações a serem executadas de modo a deixar o cartão pronto para funcionamento podendo ser capaz de receber e enviar pacotes de dados. Todo o processo de inicialização é implementado pela função `wifidev_init()`. As tarefas executadas por esta função são mostradas no fluxograma da figura 4.4.

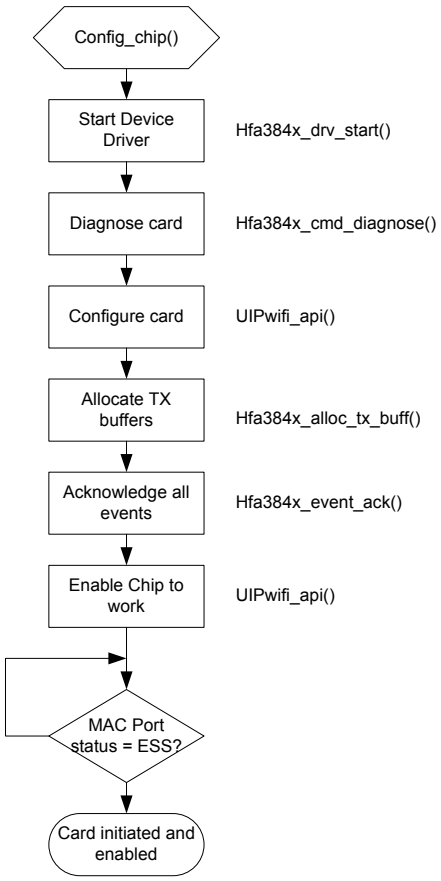


Figura 4.4: Sequência de tarefas executadas pela função wifidev_init() tendo ao lado qual função implementa cada bloco

A função `config_chip()` é uma configuração do microcontrolador apenas. Nela são configurados o clock do microcontrolador, configuração de pinos de I/O, configuração da UART para debug de programa através da porta serial, etc .

Toda a sequência de inicialização do cartão de rede sem fio descrita no bloco *Start Device Driver* mostrado na figura 4.4 foi incorporada na função `hfa384x_drvr_start()`. A descrição das tarefas executadas por esta função é descrita no fluxograma mostrado na figura 4.5.

A primeira tarefa relacionada com o cartão de rede sem fio é iniciada através de um reset no cartão Compact Flash feito colocando-se o pino de reset do cartão CF em nível lógico alto esperando-se por quatro ciclos de NOPs[1] e em seguida retornando o pino para nível lógico baixo. Em seguida a estrutura *CIS* é lida através da função `ReadCISTable()` para identificação do *MAC Address* e do endereço do Registrador de Opção de Configuração

[1] 1 NOP $\doteq$ 1us

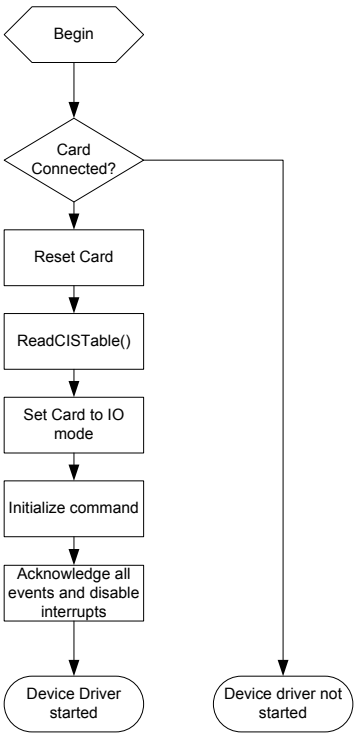


Figura 4.5: Sequência de tarefas executadas pela função hfa384x_drvr_start()

(*COR*) do cartão. Uma vez descoberto qual o endereço do COR, configura-se o cartão para operar no modo I/O setando o bit 0 deste registrador. Convém enfatizar que devido ao endereço do *COR* possuir 10 bits foi necessário no retrabalho do circuito, considerar 9 pinos do microcontrolador para acesso a este registrador somente. Foram considerados 9 pinos pois os dois bits mais significativos são somente usados para acesso a este registrador e por possuirem o mesmo valor lógico foram curto circuitados.

Para a leitura da tabela *CIS* e escrita no registrador *COR* são utilizadas funções de leitura e escrita `CFReadMemByte()` e `CFWriteMemByte()` que foram implementadas e validadas seguindo o padrão dos sinais mostrados na figura 4.1. A interação com o cartão Compact Flash até esta etapa do programa foi apenas com a memória de atributos do cartão. Esta interação só se faz necessária durante esta primeira etapa de inicialização.

A segunda tarefa inicializa o cartão Compact Flash através de um comando de inicialização. Este comando de inicialização é passado como parâmetro para a função `hfa384x_cmd()`. Através desta função é possível escrever alguns dos 12 comandos no PRISM MAC. A sequência necessária para a escrita de um comando no PRIM MAC é mostrada na figura 4.6.

Todos os comandos e registradores do PRISM MAC possuem formato de 16 bits. Como a interface Compact Flash com o microcontrolador é de oito bits foi necessário o desenvolvimento de funções para suportar este tamanho de barramento. Primeiramente foram

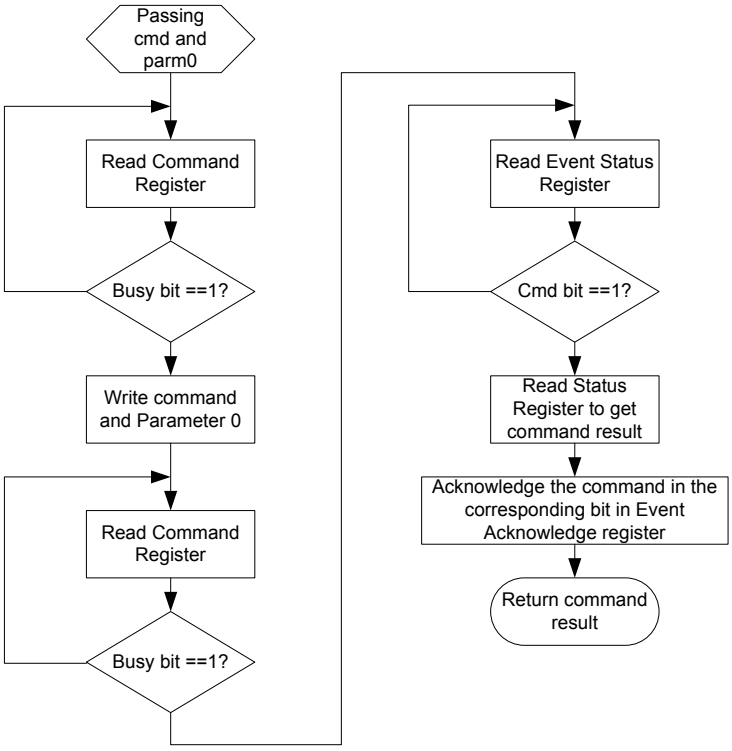


Figura 4.6: Sequência necessária para a escrita de um comando no PRIM MAC

desenvolvidas as funções `CFWriteIOByte()` e `CFReadIOByte()` para escrita e leitura em oito bits nos registradores de entrada e saída. Posteriormente foram criadas funções as funções `hfa384x_setreg()` e `hfa384x_getreg()` para escrita e leitura nos registradores do PRISM MAC. Foi utilizado para isto a chamada das funções anteriores de oito bits duas vezes passando como parâmetro no caso da escrita os bytes de registradores e valores a serem escritos e no caso da leitura os bytes do registrador a ser lido. A função de escrita nos registradores ficou então com o formato mostrado a seguir:

```
void hfa384x_setreg(int val, unsigned int reg)
{
    /* Write the low byte first */
    CFWriteIOByte((char)val&0xFF, reg);
    CFWriteIOByte((char)(val>>8), reg+1);
    return;
}
```

Seguindo a sequência de inicialização mostrada na figura 4.4, para certificar que o cartão estava funcionando internamente e também que os pinos da interface Compact Flash es-

tavam operando normalmente foi utilizado um comando para diagnóstico do cartão. Para execução deste comando foi necessário a implementação de uma função especial denominada `hfa384x_cmd_diagnose()`. A sequência de comandos desta função se assemelha à função `hfa384x_cmd`, porém, ao invés de um parâmetro são passados dois. No final da execução do comando de diagnóstico, se o cartão estiver funcionando internamente é retornado um resultado de sucesso e caso contrário um código de falha. No final também se os pinos da interface estão funcionando corretamente, os dois parâmetros passados anteriormente podem ser lidos em registradores específicos. O teste com esta função ajudou a tirar uma duvida inicial de ter-se danificado o cartão por ter descoberto após alimenta-lo com a tensão de alimentação de 5V que o circuito suportaria somente 3,3V. Após o funcionamento deste comando tanto com o resultado de sucesso retornado quanto os valores lidos foi possível verificar que o cartão poderia ser utilizado sem problemas devido a boa parte do circuito ter sido validada.

Durante o processo de inicialização do cartão vários parâmetros da rede sem fio podem ser configurados. Entre eles pode-se citar:

- Modo de operação da rede sem fio: infra estrutura (*BSS* ou Independente (*IBSS*);
- Nome da rede que se deseja conectar (*SSID*);
- Tamanho máximo do pacote de dados a ser recebido pelo cartão de rede sem fio.

Para configuração destes parâmetros e outros foi desenvolvido uma pequena API *(Application Program Interface)* com o protótipo `uIPwifi_api(unsigned int cmd, unsigned int val)`. Com esta API, depois do comando de diagnóstico, a rede foi configurada para operar no modo infraestrutura. O nome da rede a se conectar foi setado de acordo com o teste a ser realizado. O comprimento de um pacote máximo de dados foi ajustado inicialmente para o tamanho do buffer interno do microcontrolador de recepção dos dados com tamanho de 1000 bytes.

A API desenvolvida utiliza um comando de configuração para alterar os parâmetros do cartão de interface de rede sem fio. Este comando foi implementado por meio da função `hfa384x_cmd_access()`. Através dela é possível ler ou modificar algum registro de configuração. Este registro de configuração possui um código identificador de 16 bits que é denominado de *RID (Resource Identificator)*. Durante o reset e inicialização do cartão os registros de configuração são preenchidos com os seus valores padrões. Os registros de configuração são organizados basicamente em dois grupos:

- Parâmetros de rede, entidades de configuração estática - configurados com o cartão em modo desabilitado. Influenciam o comportamento do processo de habilitação do cartão. Exemplos: Tipo de rede a se conectar, configuração do *SSID*, configuração do tamanho máximo de dados.
- Parâmetros de rede, entidades de configuração dinâmica - contem valores que imediatamente influenciam na execução de processos com o cartão em modo habilitado.

Exemplos: Configuração do modo gerenciamento de energia, configuração do cartão em modo promíscuo.

A troca de informações entre o microcontrolador e o PRISM MAC é feita através de uma estrutura de registros que possui o formato mostrado na tabela 4.4. Esta estrutura de gravação é armazenada em um buffer temporário da API `uIPwifi_api` e passado como ponteiro para a variável `buf` da função `hfa384x_cmd_access`. De forma geral a API tem o seguinte formato:

```
void uIPwifi_api(unsigned int cmd, unsigned int val)
{
Declaração de variáveis

  Switch(cmd)
  {
    case valor_cmd:
      tmp_buff[0]   = tamanho da estrutura;
      tmp_buff[1]   = RID;
      tmp_buff[2]   = Dado 1 a ser configurado no RID;
      ...
      tmp_buff[2+n] = Dado n a ser configurado no RID;
      hfa384x_cmd_access(write, RID, tmp_buff, tamanho da estrutura);

      break;
    ...
    ...
  }
}
```

Offset de palavras (Words)	**Nome do campo**	**Tamanho(Palavras)**
0	Tamanho do registro	1
1	RID	1
n	Dados(opcionais)	variável

Tabela 4.3: Estrutura de registros para configuração dos parâmetros do PRISM MAC

O acesso ao registro de configuração é feito através dos chamados Caminho de Acesso aos Buffers ou *BAPs(Buffer Access Paths)*. O PRISM MAC possui dois conjuntos de

registradores *BAP0* e *BAP1* que são usados para acesso ao buffer. Para a leitura ou escrita dos dados nos *BAPs* foi desenvolvida a função `char hfa384x_rdwr_bap()`. Esta função é usada não só para modificação ou configuração dos parâmetros do cartão de rede sem fio mas também para a escrita e leitura de dados. A função para acesso aos *BAPs* é explicada com mais detalhes nas seções seguintes.

Posteriormente, ainda na etapa de inicialização do dispositivo são alocados buffers identificados por descritores de quadros. A alocação de buffers é necessária dentre outras coisas para transmissão de dados. Cada descritor de quadro possui um identificador denominado de *FIDs*. A alocação dos buffers de transmissão foi feita através da função `hfa384x_alloc_tx_buff()`. Foram alocados 3 buffers de transmissão que se mostrou ser um número suficiente uma vez que eles podem ser reaproveitados. Os buffers foram alocados utilizando-se outro comando do PRISM MAC, o comando *allocate*. Este comando inicia a alocação de buffers, que uma vez alocado é sinalizado através de um bit no registrador de eventos e o FID pode ser lido através de um outro registrador específico. O comando allocate é invocado através da função `hfa384x_cmd_allocate()`. O tamanho do buffer alocado foi de 1560 bytes que corresponde ao tamanho de uma pacote ethernet somado aos cabeçalhos Ethernet e estrutura de quadro do PRISM MAC.

As etapas até agora descritas permitiram inicializar e configurar diversos parâmetros do cartão. Após este processo, o cartão está pronto para se conectar a uma rede sem fio 802.11b. Para se conectar à rede é utilizado o comando *Enable*. Este comando é implementado pela API através da sintaxe `uIPwifi_api(ENABLE, ON)` que chama a função `hfa384x_cmd` ateriormente descrita. Após este comando todo o processo de associação, autenticação e outros implementados pelo protocolo 802.11b são executados em plano de fundo e o microcontrolador consegue saber se o processo de conexão foi bem sucedido lendo uma *RID* dinâmica que identifica o estado da conexão.

Após o processo de habilitação do dispositivo, ele está pronto para enviar e receber pacotes, tarefas que são descritas a seguir.

4.7.2 Quadros de Comunicação do PRISM MAC

Para entendimento da recepção e transmissão de dados, primeiro se faz necessário o conhecimento do formato da estrutura do quadro de comunicação utilizado pelo PRIM MAC. Este quadro é mostrado na tabela 4.4 com os principais campos mostrados. Nela pode-se notar quatro segmentos distintos sendo eles controle, cabeçalho 802.11, cabeçalho 802.3 (*Ethernet*) e dados.

O segmento de controle possui campos que permitem controlar parâmetros de transmissão e obter informações sobre parâmetros de recepção. Na transmissão por exemplo, pode-se controlar a geração de sinalizadores para dados enviados ou não enviados com sucesso, setar qual dos cabeçalhos (802.3 ou 802.11) usar na transmissão dos dados, etc. Já na recepção pode-se obter informações sobre o nível de sinal recebido pelo modem RF do cartão, taxa de recepção do quadro e outros.

Os segmentos de cabeçalho 802.3 e 802.11 utilizados pelo PRISM MAC são padrões

Offset(Words)	Estrutura do quadro TX	Segmento da estrutura	Estrutura do quadro RX
0	Status	Controle	Status
5			
6	Controle TX		
7	Controle Quadro	Cabeçalho 802.11	Controle Quadro
9	Endereço 1		Endereço 1
12	Endereço 2		Endereço 2
15	Endereço 3		Endereço 3
18	Controle de Sequência		Controle de Sequência
19	Endereço 4		Endereço 4
22	Tamanho dado		Tamanho dado
23	Endereço Destino	Cabeçalho 802.3	Endereço Destino
26	Endereço fonte		Endereço Fonte
29	Tamanho dado		Tamanho dado
30	Dados	Dados	Dados

Tabela 4.4: Formato dos Quadros de Comunicação do PRISM MAC

exceto que no pacote 802.3, o campo de tipo de pacote recebido (IP ou ARP) é substituido pelo campo de tamanho de dados. O PRISM MAC só recebe frames de dados sendo que todos os outros frames de controle e gerenciamento são manipulados internamente pelas funções do PRISM MAC. Na transmissão, no cabeçalho 802.11, os campos de controle do frame, endereço 2 e sequência de controle são preenchidos automaticamente pelo *firmware* do PRISM MAC.

O campo de dados possui os dados recebidos ou a serem enviados pelo PRISM MAC. Dentro do campo de dados existe encapsulada uma outra estrutura denominada *SNAP(Sub-*

Network Access Protocol). Ela possui oito bytes e nos seus dois ultimos bytes é implementado o campo de tipo de pacote de dados (ARP ou IP) que foi útil na montagem do cabeçalho original 802.3 na recepção dos dados. O campo de dados, incluindo a estrutura *SNAP* possui capacidade máxima de 2304 bytes caso o quadro não esteja encriptado ou a encriptação seja feita pelo *firmware* e 2312 bytes caso a encriptação seja feita pelo programa de controle.

Devido a facilidade de manipulação do cabeçalho Ethernet e sua fácil integração com a pilha TCP/IP utilizada, ele foi utilizado como cabeçalho para recebimento e transmissão dos dados. Na recepção, o endereço fonte do cabeçalho 802.3 é armazenado em uma tabela ARP para depois ser aproveitado no envio de volta para este mesmo endereço fonte. Na transmissão dos dados, o cabeçalho 802.3 é setado na estrutura de controle e o PRISM MAC gera automaticamente o cabeçalho 802.11 a partir do 802.3. Com isso não é preciso a manipulação dos endereços dos pontos de acesso intermediários para a conexão sem fio, somente o endereço fonte e destino. Estas simplificações facilitaram bastante o desenvolvimento das funções de transmissão e recepção dos dados.

4.7.3 Recepção de dados do cartão de rede sem fio

Para a recepção dos dados foi implementada a função `wifidev_read()`. Ela é responsável pela verificação de recebimento dos dados e armazenamento dos dados recebidos. A função retorna a quantidade de bytes recebidos durante o processo de recepção. A função de recepção dos dados utiliza duas variáveis que serão utilizadas pela pilha TCP/IP utilizada. Estas variáveis representam o buffer de dados recebidos, chamado de `uip_buff` e o tamanho deste buffer, `uip_len`. A sequência de tarefas executadas pela função de recepção é mostrada na figura 4.7.

O recebimento dos dados é sinalizado através de um evento de recepção, representando por um bit setado no registrador de eventos. Se este bit estiver ativo, a função procede com a recepção dos dados lendo um descritor de arquivos *FID* localizado em um registrador de descritores de arquivo de recepção. Depois a função `get_free_bap()` verifica qual dos dois Caminhos de Acesso ao Buffer (*BAPs*) está livre para recepção dos dados e em seguida o processo de acesso aos dados via *BAP* escolhida tem início. O descritor de arquivos é escrito no registrador de seleção do *BAP* escolhido e o deslocamento*(offset)* para recebimento dos dados é setado.

O *offset* foi configurado para começar no último campo do cabeçalho 802.11 que de acordo com a tabela 4.4 começa no byte 44. A partir dele o tamanho do quadro de dados é lido e armazenado na variável `data_len`. A leitura dos dados é feita através do registrador de dados referente ao *BAP* escolhido. A cada leitura feita neste registrador, um ponteiro interno ao PRIM MAC é incrementado. O buffer `uip_buf[]` é preenchido a partir do cabeçalho 802.3 com o endereço MAC fonte, endereço MAC destino, tamanho do pacote novamente, *SNAP* e dados.

Para que o cabeçalho 802.3 fosse utilizado na forma padrão, o campo de tamanho de dados lido e armazenado no buffer `uip_buf[]` foi substituido pelo campo de tipo de pacote

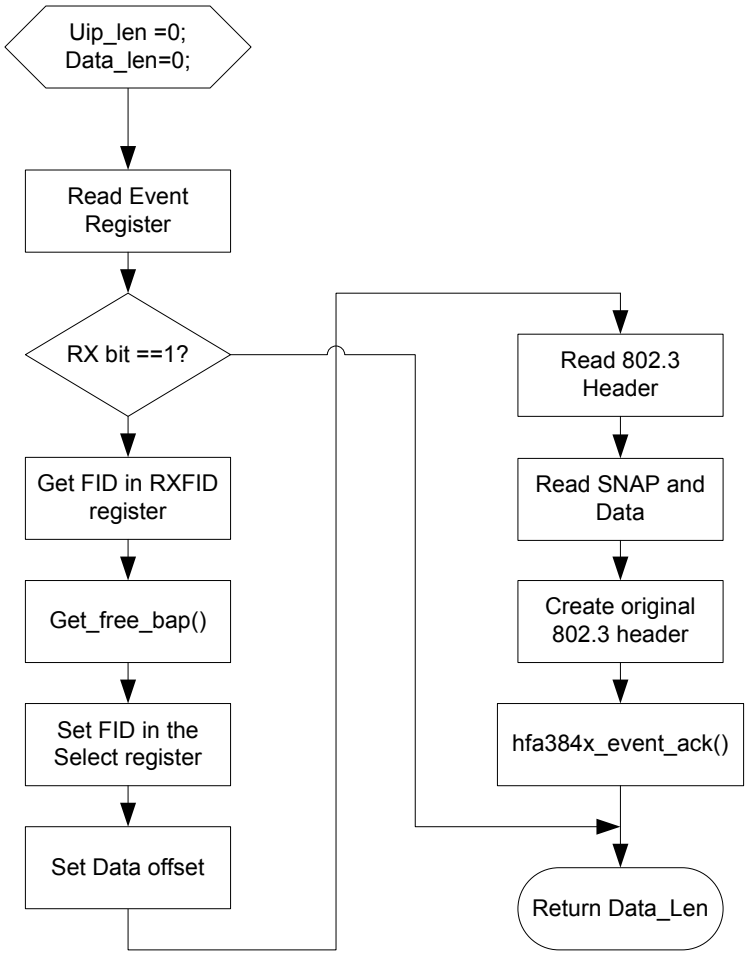


Figura 4.7: Tarefas executadas pela função de recepção wifidev_read

de dados, existente no protocolo *SNAP*. Esta adaptação permitiu que a pilha TCP/IP utilizada não precisasse de muitas modificações uma vez que tinha sido projetada originalmente para ter o protocolo Ethernet como nível de interface de rede.

Para que na próxima chamada da função `wifidev_read`, o bit de recepção de dados do registrador de eventos seja interpretado corretamente é necessário que o correspondente bit no registrador de reconhecimento seja reconhecido. Isto é válido para a ocorrência de qualquer evento ocorrido no PRISM MAC que esteja associado com o registrador de eventos. Além do evento de recepção, o registrador de eventos é capaz de registrar os seguintes eventos relacionados com o driver de dispositivo desenvolvido:

- Comando completo - sinalizado toda vez que a execução de um comando é completada pelo PRISM MAC;
- Alocação de buffer - sinalizado quando um buffer de estrutura de quadro é alocado

ou solicitado novamente para transmissão de dados;

- Erro de Transmissão - sinalizado toda vez que existe um erro de transmissão;
- Transmissão - sinalizado toda vez que que os dados são transmitidos com sucesso pelo PRISM MAC.

Para cada um dos sinalizadores acima existe um bit no registrador de reconhecimento que deve ser setado toda vez que um dos eventos ocorre. Logo em seguida o PRISM MAC se encarrega automaticamente de resetar internamente os bits de reconhecimento. Para o reconhecimento da recepção de dados bem como outros eventos que por ventura vierem a ser gerados foi criado a função `hfa384x_event_ack()` que verifica os bits setados no registrador de eventos e seta o bit correspondente no registrador de reconhecimento.

A função de recepção de dados `wifidev_read` retorna *zero* caso não seja gerado um evento de recepção de dados e *um* caso o tamanho do pacote recebido seja maior do que zero.

4.7.4 Transmissão de dados no cartão de rede sem fio

Para a transmissão dos dados foi elaborada a função `wifidev_send()`. Ela é responsável, além de transmitir os dados, por executar algumas tarefas para que a operação de transmissão seja bem sucedida. Esta sequência de tarefas é mostrada na figura 4.8.

Na recepção dos dados foi necessário transformar o cabeçalho 802.3 para o formato padrão. Na transmissão dos dados é necessário fazer o oposto, ou seja, tramsformar o cabeçalho 802.3 padrão criado pela pilha TCP/IP no cabeçalho 802.3 usado pelo PRISM MAC. Isto é possivel copiando o tamanho do pacote de dados armazenado na variável `uip_len` para as posições do buffer `uip_buf` referentes ao tamanho do pacote.

Posteriormente, um dos três descritores de arquivo alocados no processo de inicialização do dispositivo são usados para copiar os dados de transmissão do buffer `uip_buf` para o *BAP* através da função `hfa384x_rdwr_bap()`. Através de um *FID*, os dados do buffer de transmissão são passados para a função através de um ponteiro e ela os escreve em um dos *BAPs* do PRISM MAC. O funcionamento da função de leitura e escrita nos *BAPs* segue o mesmo processo descrito para a recepção porém, para a transmissão, os dados serão escritos nos *BAPs* ao invés de serem lidos.

Depois que os dados são copiados para o PRISM MAC, a função de transmissão envia um comando de transmissão para ele e em seguida aguarda pelo sucesso do comando que é indicado por um bit no registrador de eventos. O evento de transmissão é então reconhecido pela função `hfa384x_event_ack()` e o tamanho do buffer é resetado para que o buffer possa ser utilizado corretamente pela pilha TCP/IP a qual descrevemos a seguir.

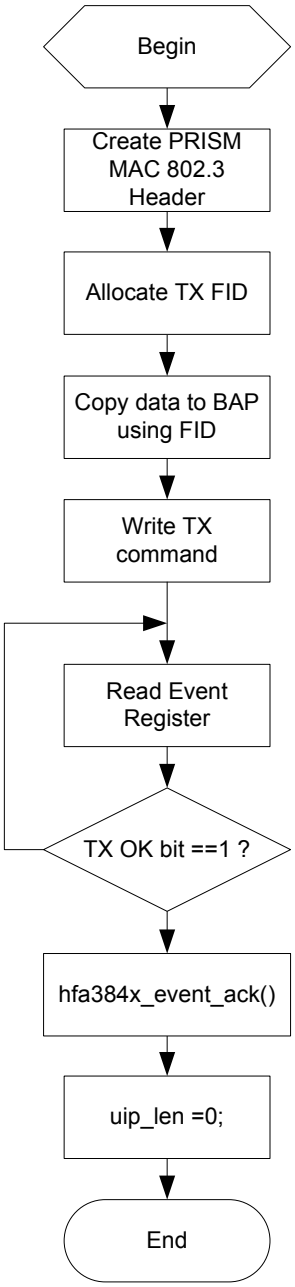


Figura 4.8: Tarefas executadas pela função de transmissão wifidev_send

4.8 A Pilha TCP/IP uIP

Dentre todas as pilhas TCP/IPs apresentadas no capítulo 3 a que se mostrou mais apropriada para o projeto foi a pilha TCP/IP uIP [Dum03]. Dentre as várias razões para se ter escolhido a pilha uIP pode-se destacar primeiramente o fato de ela já ter sido portada

para vários microcontroladores e compiladores, inclusive para a plataforma 8051 e Keil51, mostrando compatibilidade com o projeto. Outro fato relevante foi o tamanho do código que também se mostrou bastante adequado perante as necessidades de projeto.

A implementação uIP escolhida foi a utilizada em [mur]. Esta implementação além de portar a pilha uIP para o microcontrolador AT89C51AC2(família 8051) da Atmel, foi adaptada para o compilador C Keil51. Como interface de rede ela foi originalmente implementada com um *device driver* para o chip MAC Ethernet RTL8019AS da Realtek. Para o nível de aplicação foi desenvolvido um mini servidor web que apresenta algumas estatísticas da pilha TCP/IP tais como número de pacotes de cada protocolo aceito, número de pacotes perdidos, estados das conexões, etc. A parte reaproveitada deste código foi a referente à pilha TCP/IP e em um primeiro momento a aplicação do mini servidor web para teste da implementação dos protocolos. Todo o código da pilha reaproveitado é dividido nos arquivos conforme a seguir:

- uip.c: implementa os protocolos TCP/IP e ainda o protocolo ICMP e ainda o protocolo UDP;
- uip_arp.c: implementa o protocolo ARP com a criação e manutenção das tabelas de endereço MAC;
- uip_arch.c: implementa as funções para adição de 32 bits e cálculo de checksums;

A idéia geral de funcionamento da pilha TCP/IP em relação ao projeto proposto pode ser observado na figura 4.9. Ela pode ser entendida como o ponto central de conexão entre o *Device Driver* e a aplicação. Na recepção dos dados feita pela função `wifi_read()`, o buffer de dados `uip_buf` é preenchido com os dados e o tamanho dos dados `uip_len` sendo maior do que zero sinaliza para a pilha TCP/IP que existem dados para serem processador. Os dados recebidos são analisados pelas funções da pilha TCP/IP e se os cabeçalhos estiverem corretos, os dados são repassados para a aplicação. Durante o processo de conexão e desconexão, a pilha TCP/IP implementa toda a máquina de estados TCP e outros mecanismos internos a este protocolo para que a conexão seja confiável de modo a garantir a entrega correta dos dados. No envio dos dados, a aplicação passa para a pilha TCP/IP os dados a serem enviados e ela se encarrega de preencher os cabeçalhos para os protocolos. Todos os dados de transmissão são colocados no buffer `uip_buf` e a variável `uip_len` é configurada com o tamanho do pacote a ser enviado. A partir dai a função `wifi_send()` fica encarregada de enviar estes dados para o PRISM MAC.

Não é objetivo do trabalho detalhar o funcionamento da pilha uIP porque ela não foi desenvolvida neste trabalho. Ela apenas foi portada e reutilizada para o projeto proposto. Portanto serão apresentadas e explicadas somente as funções da implementação, relacionadas com as aplicações desenvolvidas. Estas aplicações são um assunto a ser discutido no item a seguir.

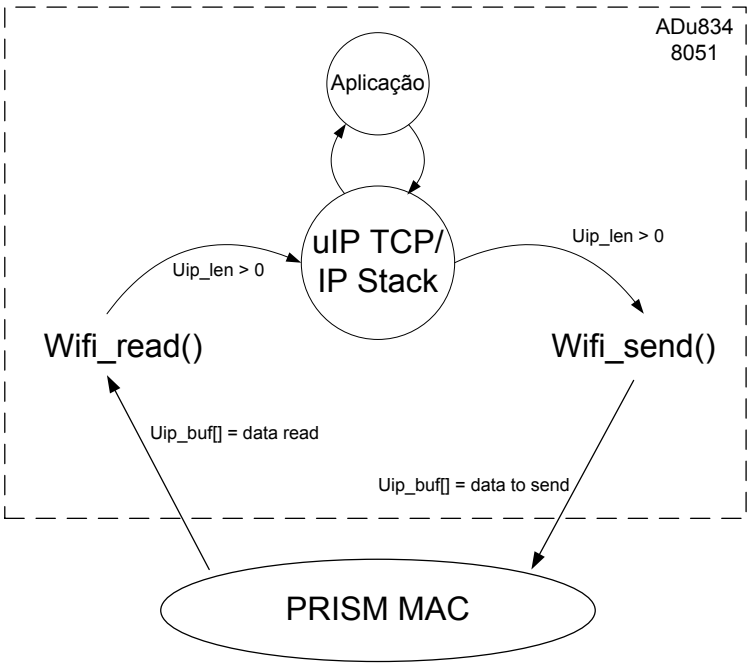


Figura 4.9: Idéia Geral do funcionamento da pilha TCP/IP

4.9 Aplicações desenvolvidas

Para conseguir que o monitor cardíaco funcionasse de acordo com os pré requisitos foi necessário dividir a aplicação principal. De um lado ficou a parte responsável pela coleta de informações da parte ECG do projeto e na outra ficou a aplicação relacionada com a camada de aplicação da pilha TCP/IP.

A idéia geral do lado da aplicação médica era que o monitor cardíaco sempre coletasse com certa frequência os dados de ECG do paciente e enviasse para a aplicação TCP/IP, em determinada situação os dados coletados. Mais precisamente, o que havia se pensado era criar um buffer circular que fosse capaz de armazenar os dados referentes a um ECG de um paciente durante 5 minutos. O paciente em um caso de emergência por exemplo, apertaria um botão da placa e desta forma seria capaz de enviar estes dados para o seu médico ou para uma central médica que tomaria as providências necessárias. Esta idéia original foi descartada porque para se obter uma boa resolução do sinal ECG é necessário uma frequência de amostragem de 1KHz. Durante 5 minutos seriam necessários 300.000 amostras o que implicaria em um buffer de tamanho aproximado de 293KB. Como a memória disponível para o armazenamento dos dados é somente a memória RAM do microcontrolador que se limita a pouco mais do que 1KB não é possível para o presente projeto suportar tal aplicação médica de ECG. O máximo que se consegue fazer com o projeto em questão é utilizar a memória de dados Flash do microcontrolador ADUc 834 que possui 4Kbytes para armazenamento. Com uma frequência de amostragem de 100Hz, seria possível armazenar

4096 amostras de oito bits em cerca de 41 segundos.

Outro problema encontrado durante o desenvolvimento da aplicação médica foi que sinais importantes para o funcionamento do *ASIC* e do conversor A/D tiveram que ser realocados para suportarem o funcionamento do modo I/O da interface Compact Flash. Com isto não foi possível a implementação no presente projeto de nenhuma aplicação que suportasse o ASIC ECG e o conversor A/D.

Para simular uma aplicação médica foi implementado uma interrupção associada ao temporizador *0 (Timer 0)* do microcontrolador que supostamente coleta as amostras do conversor A/D mas que na realidade são partes de amostras de um sinal ECG pré gravadas na memória. Esta interrupção é chamada com uma frequência de 24Hz. Ao ser chamada, a interrupção copia para o buffer de saída um byte armazenado na memória. A aplicação médica possui inteligência descentralizada, ou seja, não faz o processamento local dos dados. Portanto, é feito somente a coleta, armazenamento em um buffer central e envio de dados para o servidor onde eles podem ser analisados e processados posteriormente.

Para resolver o problema da escassez de memória RAM , uma possível abordagem não implementada seria coletar os dados até que o buffer fosse completado e depois disso fosse enviado. Com uma frequência de amostragem de 100Hz, o buffer com 1000 posições seria completado a cada 10 segundos. Para o envio de 5 minutos de dados, seriam necessários 30 envios de buffers completos para o servidor.

No que diz respeito a aplicação do lado da camada de aplicação da pilha de protocolos TCP/IP, por causa da necessidade de economia de energia, era necessário desenvolver uma aplicação que atuasse sob demanda, ou seja, ficasse em estado de economia de energia. A idéia seria que a aplicação apenas faria a aquisição de dados e quando fosse necessário "acordasse" e enviasse os dados adquiridos. Para atuar desta forma é necessário que a aplicação funcione como um *cliente*. Uma segunda alternativa é permitir que o médico veja o status do paciente remotamente e para isto se conecte ao endereço IP do monitor cardíaco e seja capaz de acompanhar os dados via um *Browser*, por exemplo. Atuando desta forma, a aplicação funciona como um *servidor*.

Foi elaborado uma aplicação *cliente* que a partir de um botão de emergência, muda o estado de uma variável interna ao programa e conduz a um processo de abertura de conexão ativa com um servidor web. Ao ser contactado, o servidor web, estabelece conexão com a placa *ECG*. A aplicação ECG prepara os dados para serem enviados formatando um pequeno cabeçalho que inclui a informação de hora, minuto e segundo provenientes do relógio de tempo real do microcontrolador e também da informação da frequência de amostragem dos dados provenientes do suposto conversor A/D. Depois da preparação dos dados, a aplicação cliente envia uma requisição *HTTP* do tipo *POST* para o servidor que recebe os dados enviados e os processa. O processamento dos dados pelo servidor não foi implementado deixando-o livre para a implementação de uma apresentação dos dados pelo próprio servidor ou mesmo o envio deles por email a partir de um script no servidor.

A aplicação médica e a aplicação *HTTP* foram escritas em aquivos separados. A pilha TCP/IP só faz acesso à função `webclient_appcall()` para manipulação dos dados. Esta função monitora alguns parâmetros da pilha tais como necessidade de retransmissão, timeout, abertura e fechamento de conexão através do teste de alguns sinalizadores da

pilha uIP. A aplicação ECG é acessada somente uma vez pela aplicação *cliente* através da função `ecg_prepare_data()` que prepara os dados para serem enviados para o servidor. Esta forma de organização permite manter separada e modular as aplicações de acesso à internet e acesso aos dados de ECG.

Para teste inicial da pilha TCP/IP e para demonstrar a capacidade de hardware e software para atender a um dos pré requisitos do projeto foi implementado como aplicação de Internet a partir do código [mur] um mini servidor web que é capaz de informar o estado da conexão em forma de estatísticas indicando também quantas conexões estão ativas. Através desta aplicação foi possível verificar que é possível com uma pequena modificação no código do programa permitir que um médico acompanhe os dados do paciente através do acesso remoto ao monitor cardíaco.

4.10 Gerenciamento de Energia nas aplicações

Para a implementação de um sistema de gerenciamento de energia nas aplicações desenvolvidas foi pensado inicialmente em duas abordagens para o projeto: uma utilizando as características do microcontrolador ADUc 834 e a outra explorando os recursos do cartão 802.11b na tentativa de economia de energia. Verificou-se através dos testes mostrados no capítulo a seguir que a utilização dos recursos do cartão 802.11b não foram satisfatórios e portanto foram descartados no fechamento da aplicação final.

Foi implementado uma máquina de estados que possui 3 estados denominados de *ativo, ocioso(Idle)* e *Power Down*. Uma idéia geral da implementação pode ser observada na figura 4.10. Os estados são monitorados por três interrupções diferentes de acordo com o seguinte:

- Interrupção externa INT0 - entra no modo *Ativo* a partir do acionamento da chave de emergência;
- Interrupção do temporizador 0 - controla a coleta de dados no modo *Ocioso* e aciona o modo *Power down* quando a coleta termina;
- Interrupção do Modo *Power Down* - "acorda" o microcontrolador do modo *Power Down* e ativa o modo *Ocioso* para coleta de dados.

Inicialmente, quando a placa *ECG* é ligada, a aplicação entra no estado ocioso onde são coletados por cinco minutos os dados do conversor A/D referentes as amostras de dados provenientes do ASIC. Neste estado é utilizado o recurso do microcontrolador onde é ativado o modo ocioso *idle*. Neste modo o oscilador continua funcionando mas o clock gerado internamente pelo PLL(*Phase Locked Loop*) é parado. Os periféricos internos ao chip continuam a receber o clock e o funcionamento deles é mantido. O estado da CPU é preservado com a pilha, contador de programa e todos os outros registradores mantendo os seus dados. O microcontrolador sai deste estado se qualquer interrupção habilitada for acionada ou se o microcontrolador receber um reset [Dev03].

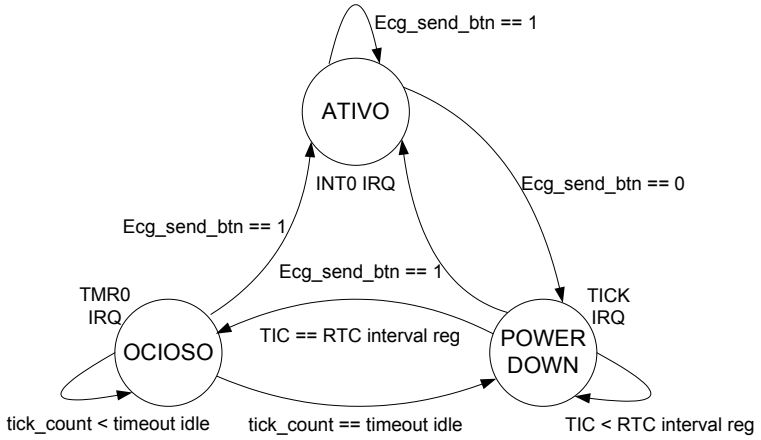


Figura 4.10: Máquina de Estados que implementa o Gerenciamento de Energia na placa ECG

Através da interrupção do temporizador 0 do microcontrolador, a variável `tick_count` é incrementada e uma função para coleta dos dados é chamada com a frequência de 24Hz. Quando a variável atinge o equivalente a 5 minutos de coleta de dados [2], o monitor cardíaco entra em modo *Power Down*.

No modo *Power Down*, o PLL e o clock do microcontrolador são parados. O oscilador do chip continua a oscilar e uma variável `TIC` associada ao clock do oscilador é habilitada. Todos os outros periféricos são desligados. Os pinos mantém todos os níveis lógicos neste modo.

Um conjunto de registradores são programados para que no intervalo de uma hora, uma outra interrupção seja acionada fazendo com que o microcontrolador saia do estado *Power Down* e vá para o estado *Ocioso*. No estado *Ocioso*, conforme descrito anteriormente, o microcontrolador fica 5 minutos coletando dados e depois retorna para o estado *Power Down*. Tanto no modo *Power Down* quanto no modo *Ocioso* o cartão de rede sem fio permanece resetado.

Os dados coletados são armazenados no buffer principal `uip_buf` e enviados para um servidor a partir de uma chave controlada pelo paciente que é associada ao pino de interrupção externa *INT0*. Quando a chave é acionada em nível lógico baixo, a interrupção associada a este pino é ativada, setando a variável `ecg_send_button` e mudando o estado do monitor cardíaco para *ativo*. Neste momento, em pouco tempo, a aplicação é chamada abrindo uma conexão com o servidor, enviando os dados para ele e aguardando por um OK no recebimento dos dados via protocolo HTTP. O servidor envia o OK, fecha a conexão e o monitor cardíaco reconhece voltando para o modo *Power Down* onde permanece até que a interrupção para a coleta de dados seja acionada ou que o modo *Ocioso* para coleta de dados seja ativado.

[2] Valor do registrador do temporizador em torno de 7200

Ao entrar no estado *Ativo*, o programa aciona o driver do cartão de rede sem fio, inicializando-o, conectando ao ponto de acesso configurado e posteriormente implementando todo o processo de conexão TCP/IP implementado pela pilha uIP. A aplicação desenvolvida se encarrega de repassar os dados coletados de *ECG* para a pilha uIP.

Através da implementação da máquina de estados foi possível reduzir o consumo de energia da placa *ECG* conforme será mostrado no capítulo seguinte.

Capítulo 5

Testes e Resultados

5.1 Ferramentas para depuração e testes dos programas

Os programas desenvolvidos foram testados primeiramente usando-se o simulador do compilador C Keil51. O compilador possui um poderoso sistema de simulação onde pode-se simular valores de variáveis, colocar *breakpoints* e verificar como o programa está se comportando de maneira geral. Para simular os valores dos registradores necessários para funcionamento do driver da interface de rede sem fio foi usado uma variável de simulação que dependendo do seu valor, atribuia valores aos registradores e variáveis. A utilização desta variável de simulação foi muito importante principalmente no desenvolvimento do driver do dispositivo de rede sem fio.

Outro recurso bastante utilizado durante o desenvolvimento foi a utilização de impressões de variáveis e informações relevantes sobre a localização da execução do programa na UART do microcontrolador ADUc834. A UART foi configurada para operar a 4800bps, 8 bits de dados, sem paridade e um bit de parada (4800 8 N 1). As impressões na porta serial eram lidas através do programa *HyperTerminal* do *Windows* conforme mostrado na figura 5.1. Com as impressões foi possível verificar como o programa rodava em diferentes situações.

Uma outra ferramenta utilizada na depuração foi a utilização de dois LEDs*(Light Emitting Diode)* existentes na placa. Através deles pode-se analisar situações onde a impressão de informações via porta serial não foi possível tais como dentro das rotinas de interrupção.

5.2 Testes com a interface de rede sem fio

Os testes de funcionamento da interface de rede sem fio foram feitos com a utilização de um Ponto de Acesso modelo WAP11 da Linksys. Este Ponto de Acesso foi configurado

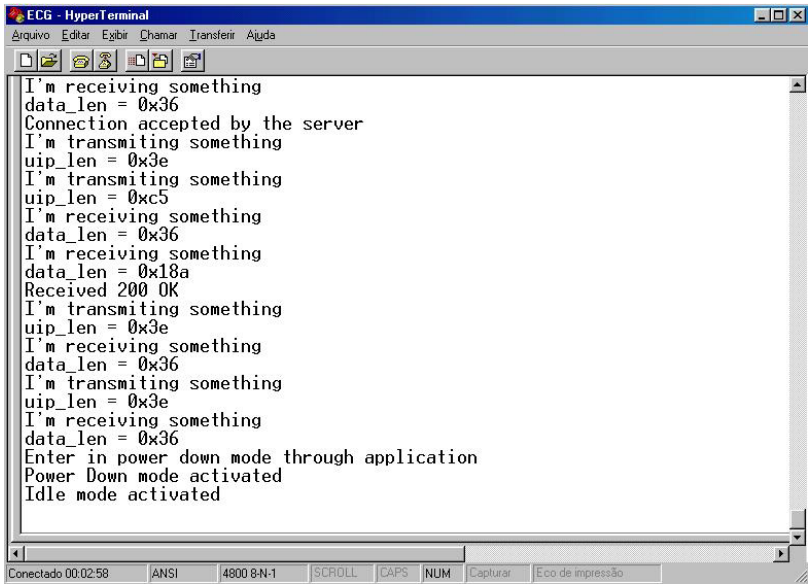


Figura 5.1: Exemplo de uma tela do Hyperterminal com uma sequência de mensagens impressas que auxiliaram na depuração dos programas desenvolvidos

para operar com o endereço IP 192.168.1.250. Ligado a este Ponto de Acesso estava um Computador PC com placa de rede Ethernet e endereço IP 192.168.1.50. A configuração de rede utilizada pode melhor ser resumida através da figura 5.2.

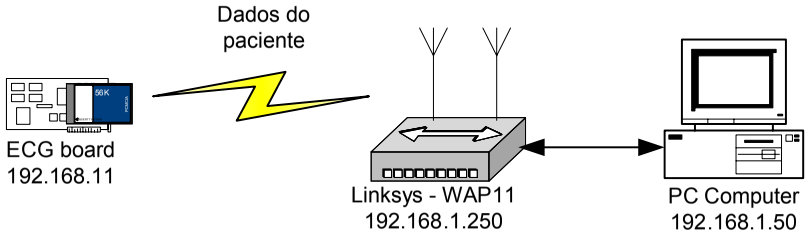


Figura 5.2: Configuração de rede utilizada para teste da Interface de rede sem fio.

O desenvolvimento do driver de dispositivo para a rede sem fio seguiu as especificações do manual [Inc02]. Os primeiros sinais de sucesso com a implementação vieram através da leitura da estrutura da informação do cartão (*CIS*). Os dados começaram a ser lidos e impressos na porta serial para confirmação. Todos os dados estavam de acordo com o manual. Depois do comando de inicialização que foi executado com sucesso, o cartão Compact flash ajudou na indicação do resultado através do LED de *Link* do cartão piscando. Ao tentar se conectar com o ponto de acesso, o LED de *link* do cartão começou a piscar com

uma intensidade maior até que ficou definitivamente aceso. Isto foi o sinal que a interface de rede havia conseguido se conectar com o ponto de acesso. Através de um identificador de recurso (*RID*) foi possível identificar o endereço de Controle de Acesso ao Meio *MAC* do ponto de acesso. Foram os primeiros sinais que indicavam que o driver da interface de rede sem fio estava funcionando.

Para teste do recebimento e envio dos dados foi utilizado a aplicação *ping* do protocolo *ICMP* que estava embutido na pilha TCP/IP *uIP*. A placa ECG foi configurada com o endereço IP fixo 192.168.1.1 e partir do computador o comando *ping* foi lançado. Inicialmente os dados foram recebidos sendo necessários alguns ajustes de deslocamento de forma a receber somente o cabeçalho Ethernet. Ao serem ecoados, os dados apresentaram problema e não puderam ser enviados. Depois de alguns dias descobriu-se que o problema estava na inversão da ordem dos buffers de saída do PRISM MAC que possui ordem inversa em relação ao recebimento o que levou a confundir o desenvolvimento da função de transmissão. Os comandos *ping* foram enviados e retornados com um tempo médio de 236ms.

5.3 Testes com as aplicações

Inicialmente foi adaptada uma aplicação desenvolvida juntamente com a pilha *uIP* para teste do funcionamento da pilha TCP/IP uma vez que as funções do driver de rede sem fio já se encontravam funcionando. A aplicação foi um mini servidor *web* que fornece informações sobre a pilha TCP/IP tais como número de conexões ativas, estatísticas sobre perda de pacotes, pacotes IP, TCP e ICMP recebidos dentre outras.

Para verificação do funcionamento da aplicação foi utilizado o software analisador de protocolo de rede *Ethereal* versão 0.10.9. Uma tela do software *Ethereal* mostrando o acesso a uma das páginas do servidor web implementado é mostrado na figura 5.3. A página com estatísticas do servidor pode ser observada na figura 5.4.

A aplicação médica em conjunto com a aplicação de internet final desenvolvida descrita no item 4.9 também foram testadas com o auxílio do software *Ethereal*. Para teste desta aplicação, a placa *ECG* atua como *cliente* e o computador PC onde estava conectado o ponto de acesso sem fio foi configurado para ser um *servidor* web. Para isto foi instalado neste computador o software HTTP server Apache[apa] versão 1.3.33. Para suporte e teste da aplicação HTTP POST implementada no monitor cardíaco foi necessário ainda a instalação e configuração do PHP versão 5.0.4.

Para teste da comunicação no protocolo HTTP usando o método POST, foi criado um arquivo na linguagem PHP no servidor WEB que apenas checa se o campo *User-Agent:* do cabeçalho HTTP POST é do tipo *ECG Webclient*. Caso positivo, uma mensagem de volta é enviada e a conexão é fechada pelo servidor. O cliente *ECG* recebe a mensagem HTTP com o cabeçalho OK, decodifica este cabeçalho e reconhece o fim da conexão enviando uma pacote de reconhecimento *ACK* e entrando no modo de energia *Power Down*.

Conforme mencionado, não foi implementado no servidor nenhuma aplicação para tratamento dos dados enviados. Apenas testou-se a capacidade de comunicação da placa *ECG*

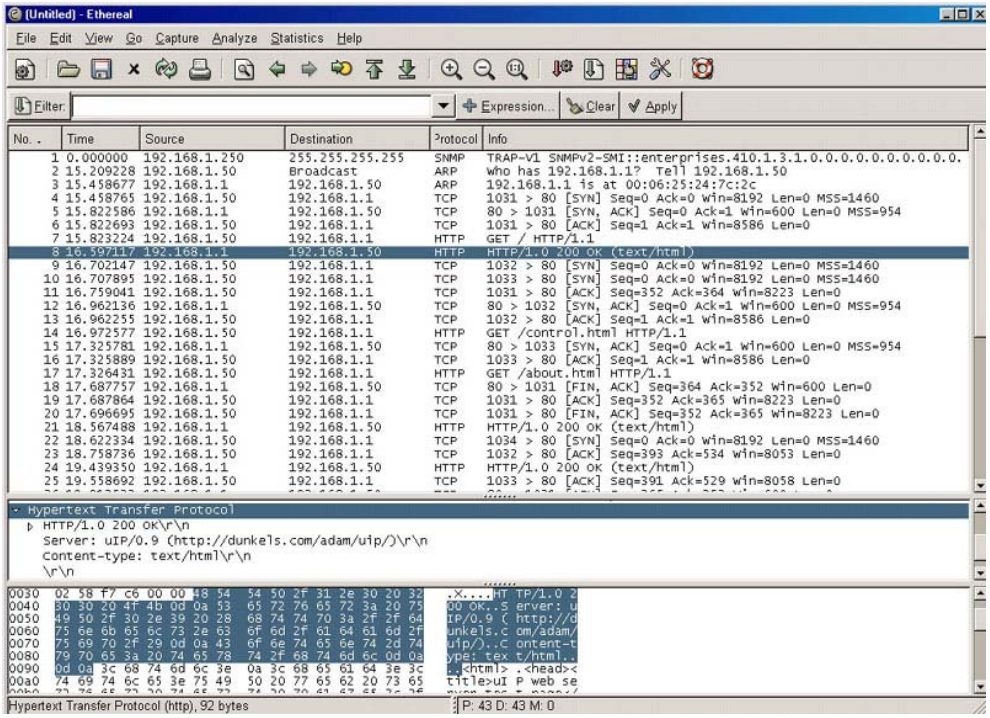


Figura 5.3: Sequência de dados a partir de um acesso ao servidor WEB implementado na placa ECG

com um servidor web. A tela do software *Ethereal* com a sequência de pacotes TCP/IP e HTTP enviados pode ser observada na figura 5.5.

5.4 Consumo de Energia do Monitor Cardíaco

Como descrito no item 4.10, foram implementados três estados diferentes de consumo de energia: *Ativo, Ocioso e Power Down*. Os modos de energia da placa *ocioso* e *Power Down* foram configurados para efeito de teste com mudança entre os estados a cada 30 segundos. O consumo de energia foi baseado no consumo de corrente da placa e medido com um multímetro digital Minipa modelo ET-1502.

Primeiramente, ao entrar no modo *Power Down* e *Ocioso*, antes da mudança para cada estado, o cartão de rede sem fio foi desconectado da rede na esperança de reduzir o consumo de energia. Com esta abordagem o consumo foi de 75mA no modo *Ocioso* e 64mA no modo *Power Down*. Decidiu-se colocar o cartão de rede sem fio resetado constantemente uma vez que ele não é utilizado em nenhum aspecto nestes dois modos. Depois desta alteração o consumo caiu para 61mA no modo *Ocioso* e 50mA no modo *Power Down*.

Descobriu-se posteriormente que o modo *Ocioso* do microcontrolador tinha sido con-

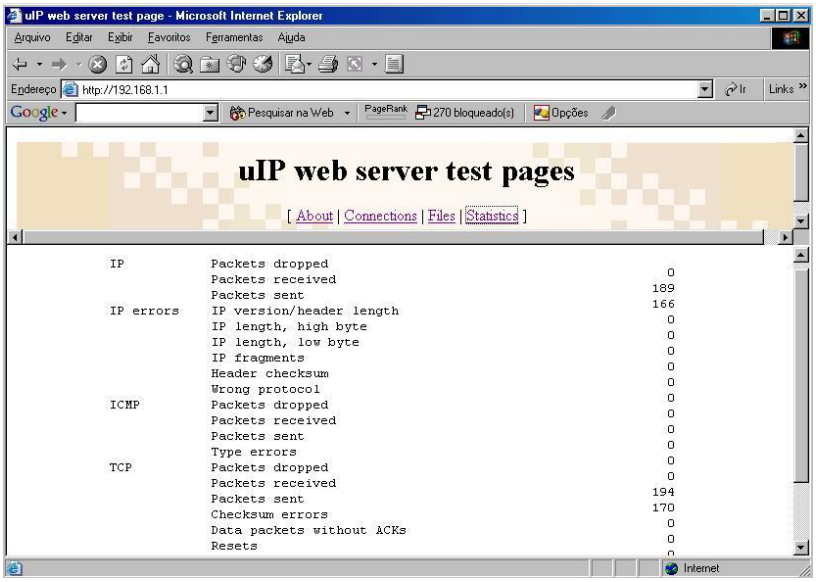


Figura 5.4: Uma das páginas acessadas no Mini-servidor Web implementado na placa

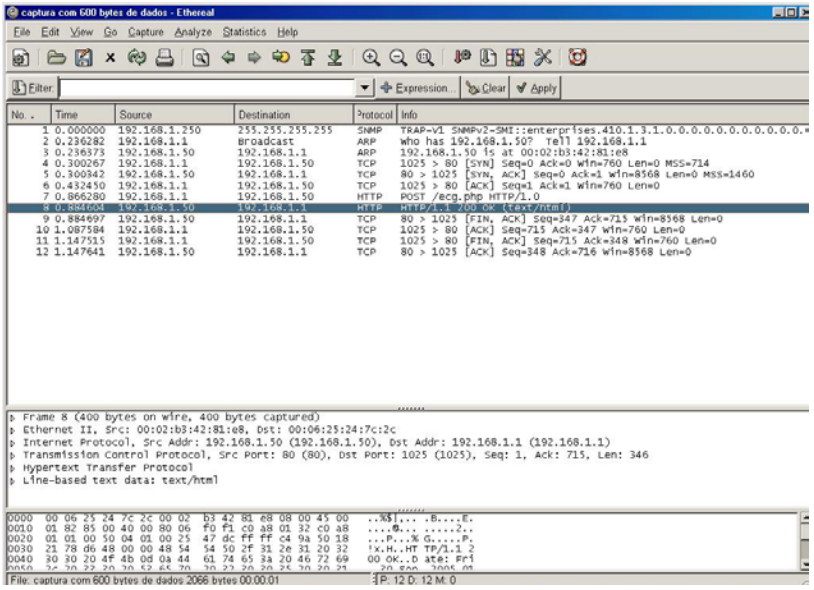


Figura 5.5: Sequência de pacotes trocados entre o cliente ECG e o servidor HTTP

figurado para ser ativado somente quando se entrava no estado e logo em seguida era desativado pela interrupção do temporizador 0. O programa foi alterado então para ativar o modo *Ocioso* constantemente durante a permanência do software neste estado. Com esta

modificação o consumo no modo *Ocioso* caiu para 55mA com a aplicação implementada.

No modo *Ativo* o consumo de energia foi variado com o mínimo de cerca de 70mA até picos de 283mA.

Implementado a máquina de estados e feito os testes partiu-se para a técnica de redução da frequência de clock do microcontrolador na expectativa de redução do consumo de energia da placa. Este teste foi possível porque o microcontrolador ADUc834 possui o recurso de alteração de frequência de clock através da configuração de um registrador onde é possível setar até oito frequências de clock diferentes chegando até a frequência máxima de 12,58MHz.

Em testes preliminares verificou-se que a escrita e leitura do cartão Compact Flash no modo I/O mostrado no capítulo 4 não funcionava com a frequência de clock menor que 12,58MHz, portanto no modo *Ativo* era necessário configurar o registrador para a frequência máxima de clock. Restava então alterar o clock no modo *ocioso* uma vez que no modo *Power Down* o registrador de clock era desligado. O clock foi alterado para a frequência padrão do microcontrolador que é de aproximadamente 1,57MHz. Com esta redução de frequência, o consumo no modo *Ocioso* caiu para cerca de 53mA. A frequência foi então configurada para o valor mínimo e com isto atingiu o consumo de cerca de 51mA se aproximando do consumo no modo *Power Down*.

Para verificar a influência do consumo do cartão de rede sem fio, foi feito outro teste desta vez removendo-se o cartão do conector Compact Flash. O consumo verificado foi de 19,5mA para o modo *ocioso* e 14,8mA para o modo *Power Down*. Verificou-se que o consumo neste caso do cartão de rede sem fio é cerca de 35mA o que significa 70% do consumo da placa. Em [Inc01] é citado que o consumo típico de corrente com Vcc=3,6V é de 33mA o que se aproxima do valor que foi medido.

No modo *ativo* foi verificado através do software *Ethereal* que o tempo para transmissão foi medido conforme a tabela 6.1. Percebe-se que quando são impressas informações via porta serial que foram usadas para debug do programa, o tempo para transmissão de dados praticamente dobra. Outro fator relevante é que cerca de 200ms a 300ms de tempo a mais é necessário para o envio de uma amostra de dados de tamanho razoável para aplicação do monitor cardíaco.

Tamanho da amostra de dados(Bytes)	**Impressão via serial?**	**Tempo(seg)**
0	Não	0,73
600	Não	1,07
0	Sim	1,82
600	Sim	2,02

Tabela 5.1: Resultados obtidos do tempo de transmissão dos dados pelo monitor cardíaco

Com base no tempo gasto para envio dos dados pode-se estimar o consumo da placa admitindo por exemplo que um paciente enviaria os dados para o seu médico 3 vezes por

dia e ficasse 24 horas por dia com o monitor cardíaco. Desta forma teriamos que em um dia 2 horas seriam dedicadas para a coleta de dados no modo *ocioso*, o que significa um consumo de 110mAh. Admitindo que o paciente enviaria os dados no momento que o monitor cardíaco estivesse no modo *Power Down*, este ficaria 21,99917 horas neste modo, consumindo 1099,96mAh. O consumo do envio dos dados seria de 0,23mAh. Somando-se tudo chegariamos ao valor de consumo de 1210,2mAh. Usando-se uma bateria de 2500mAh seria possível ao paciente utilizar o monitor cardíaco por 2 dias. Caso o paciente use o equipamento 8 horas por dia pode-se utilizar as baterias por até 6 dias.

A fonte projetada para a placa não suporta o consumo do cartão de rede sem fio por isso não foi possível fazer o teste com baterias no circuito para verificação da autonomia real do circuito. O único teste que foi feito foi o teste das baterias com o circuito sem o cartão de rede sem fio que não acrescenta em nada ao trabalho.

5.5 Tolerância à Falhas

Em alguns experimentos com a medição do consumo da placa foi percebido que em determinados momentos da transmissão dos dados, o cartão de rede sem fio provavelmente por causa do mal contato dos fios de alimentação, não conseguia transmitir. Além disto, embora não verificado, pode-se imaginar situações de conexão onde o acesso à rede sem fio pode ser prejudicado devido redução da qualidade do sinal ou alguma interferência, provocando uma falha causada por perturbação externa.

Uma outra situação de falha pode ocorrer em casos onde a energia de alimentação das baterias não consegue suprir o correto funcionamento do monitor cardíaco levando o sistema a cometer erros podendo até mesmo chegar a enviar dados incorretos sobre o paciente o que é proibido para um sistema desta categoria. Diante deste cenário, verificou-se que era necessário a implementação de um sistema de tolerância à falhas de modo a evitar que o cartão ficasse indefinidamente no modo *ativo*, justamente o de maior consumo de energia.

O microcontrolador ADUc834 fornece suporte para implementação de um sistema de tolerância à falhas através da implementação de um *Watch Dog Timer (WDT)*. Este recurso é implementado através de um registrador que é ligado a um temporizador de 16 bits. Este temporizador é setado para funcionar antes de determinada operação e se a operação não for completada até o reset dele, o microcontrolador é reinicializado e um bit do registrador de controle do *Watch Dog Timer* é setado. É possível configurar oito diferentes tempos para o reset do temporizador *WDT*.

Como os pontos críticos de funcionamento ou de provável falha no funcionamento do projeto foram identificados inicialmente como sendo relacionados com a interface de rede sem fio, foi habilitado antes das operações principais de inicialização, transmissão e recepção dos dados o temporizador *WDT* para funcionar com um tempo de 2 segundos em cada uma. O tempo de 2 segundos foi considerado como o tempo máximo para abertura de conexão TCP/IP, envio dos dados e fechamento de conexão conforme resultados obtidos no item anterior.

Caso o tempo de 2 segundos expire, o microcontrolador é resetado. Sendo resetado, inicialmente ele entra no modo *ocioso* de energia para a coleta dos dados por 5 minutos. Após a coleta dos dados, como houve o reset pelo *WDT* é possível reconhece-lo através de um bit do registrador de controle do temporizador. Desta forma pode ser identificado que houve algum problema no processo de envio dos dados já que o *WDT* é vinculado a operações relacionadas ao cartão de rede sem fio. O estado do monitor cardíaco passa novamente para o *ativo* onde novamente o processo de envio dos dados é acionado. Caso os dados sejam enviados com sucesso, o circuito entra no modo *Power Down* caso contrário o microcontrolador é resetado novamente e o processo de coleta e transmissão dos dados é invocado. Este processo é indicado na figura 5.6.

Testes foram realizados simulando condições de falha na transmissão tais como configurações do *WDT* para tempos inferiores ao de realização das tarefas principais de inicialização, transmissão e recepção dos dados e verificou-se que o reset do microcontrolador funcionou. A tentativa de envio dos dados, posterior a coleta após o reset via *WDT* foi verificada após as falhas provocadas comprovando o funcionamento do sistema.

Para resolver o problema da questão da pouca energia para a alimentação do circuito, foi idealizado mas não testado um sistema baseado em outro recurso do microcontrolador ADUc834. Trata-se do monitor de fonte de alimentação, ou *Power Supply Monitor (PSM)*. Este recurso quando habilitado indica se a alimentação cai abaixo de quatro níveis selecionados pelo usuário que variam de 2,63V a 4,63V.

A queda na alimentação abaixo dos níveis programados pelo usuário é indicada por um bit de um registrador dedicado para esta funcionalidade do microcontrolador. Este bit gera uma interrupção que quando acionada pode permitir que, por exemplo, o ultimo conjunto de dados adquiridos do paciente sejam copiados da memória RAM para a memória Flash do microcontrolador. Esta ação evitaria a perda de dados que pode ser útil em determinadas situações de necessidade de acompanhamento mais continuo, por exemplo.

5.6 Análise de custo

O projeto do monitor cardíaco utiliza componentes de baixo custo a não ser pelo preço do cartão de rede sem fio que custou em torno de U$50. Em conversas com o projetista do monitor cardíaco foi estimado o custo de protótipo do monitor cardíaco como sendo de U$300 e custo para produção em larga escala de no máximo U$250.

Em comparação com outros sistemas similares na literatura a maioria deles usavam um *PDA* como hardware principal. Considerando o preço de um *PDA* de baixo custo que inclui rede sem fio 802.11b chegariamos ao preço de $300 desconsiderando o preço do módulo de coleta dos dados fisiológicos. Também como solução para recebimento, processamento e envio dos dados pode-se usar um celular com características especiais que nos dias de hoje tem um custo aproximado de U$400 que ainda precisa ser acoplado a um módulo de coleta dos dados tendo como padrão de comunicação geralmente utilizado o *Bluetooth*.

As soluções para coleta de dados fisiológicos com as interfaces para o *PDA* e *Bluetooth* são escassas, a maioria delas são projetos não comerciais feito pelos próprios desenvolve-

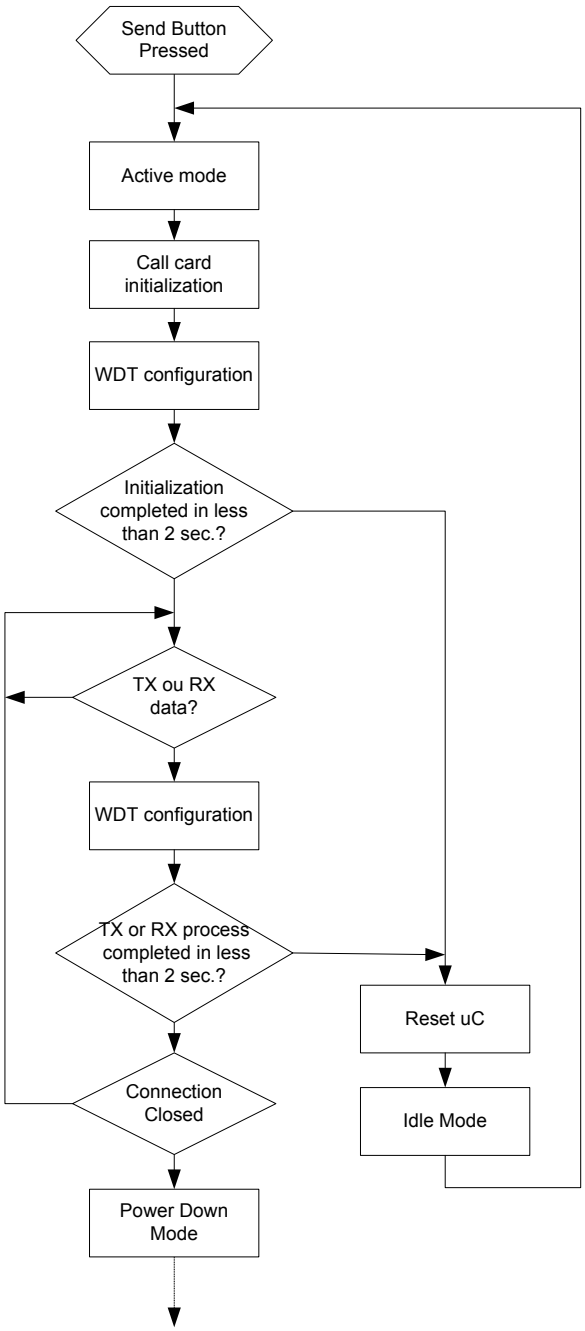


Figura 5.6: Sistema de tolerância à falhas implementado e testado para o monitor cardíaco

dores dos projetos. Isto torna os módulos de coleta caros e difíceis de serem encontrados no mercado.

Pode-se perceber então que em termos de custo o monitor cardíaco pode ser uma alter-

nativa mais viável para uso como plataforma de hardware de um sistema de monitoramento remoto de pacientes.

5.7 Tamanho do código das Aplicações

O tamanho do código foi uma importante consideração pois permitiu avaliar a possibilidade de expansão e implementação de novas funcionalidades. A tabela 5.2 sumariza o tamanho do código para as várias aplicações implementadas de acordo com o compilador C Keil51.

Aplicação	**Memória RAM de dados interna**	**Memória RAM de dados externa**	**Memória de Programa interna**
ECG Webclient	138	2023	18042
ECG Webclient (Debug Version)	138	2023	18753
ECG Webserver	183	2022	23120

Tabela 5.2: Tamanho de código em bytes para as várias aplicações desenvolvidas

Pode-se perceber que a memória RAM tanto interna quanto externa são um recurso escasso. A memória RAM externa depende muito do número de conexões que a aplicação pode receber e do tamanho do buffer `uip_buf()` utilizado. No caso da aplicação *ECG Webclient* são suportadas duas conexões e um tamanho de buffer equivalente a um pacote de dados Ethernet, ou seja, de 1500 bytes e no caso do *ECG Webserver* são suportados um tamanho de buffer de 1500 bytes e três conexões. A aplicação *cliente* possui menos suporte a conexões devido ao suporte para tamanho do cabeçalho HTTP enviado que na aplicação *servidor* possui 5 bytes e na aplicação *cliente* possui 56 bytes.

Para permitir que novas variáveis sejam adicionadas em novas aplicações que necessitarem de quantidade de memória maior é necessário ou diminuir o número de conexões ou diminuir o tamanho do buffer `uip_buff()`.

Em relação à memória de programa, percebe-se que o espaço ocupado pelos códigos da tabela 5.2 é cerca de um terço do total de memória disponível que é de 62Kbytes. Desta forma é possível ainda implementar cerca de dois terços de programa o que permite grande possibilidade de incorporação de novas funcionalidades e recursos ao sistema.

5.8 Principais dificuldades

As principais dificuldades no desenvolvimento dos trabalhos foram relacionadas com desenvolvimento do driver do dispositivo de rede sem fio. A escassez de documentação do

PRISM MAC demoraram o entendimento do funcionamento do chip uma vez que pela documentação do driver de dispositivo desenvolvido para o *Linux* era de difícil entendimento. Estes problemas só foram resolvidos quando foi obtida a documentação que descrevia melhor o funcionamento do PRISM que possibilitou identificar corretamente o endereço do registrador de reset e posterior sucesso na leitura da tabela *CIS* e do desenvolvimento das outras funções de controle discutidas anteriormente.

O PRISM MAC possui formatos diferentes para escrita e leitura dos dados nos caminhos de Acesso os Buffers *BAPs*. Isto dificultou principalmente o processo de escrita dos dados e consequentemente a transmissão destes. Alguns dias foram gastos na tentativa de se descobrir qual a ordem correta os dados deveriam ser escritos no buffer para que eles fossem transmitidos corretamente.

Em relação a modificação realizada no circuito, a checagem para verificação de quais pinos iriam ser retirados, o corte das trilhas do circuito impresso da placa e a colocação de fios para ligação dos pinos do microcontrolador aos pinos do conector Compact Flash foi um processo demorado e complicado que necessitou de um grande cuidado principalmente no que diz respeito a danificação dos componentes uma vez que, a princípio, não existiam extras para substituição.

Capítulo 6

Novo Projeto, Conclusões e Planos Futuros

6.1 Novo Projeto

Tendo em vista que o projeto da placa ECG teve que ser alterado para suportar o modo I/O da Compact Flash, alguns pinos que originalmente pertenciam à interface com o circuito de ECG tiveram de ser realocados. Por isto, não foi possível implementar uma aplicação que demonstrasse ao mesmo tempo uma aplicação médica com a interface de rede sem fio.

Para resolver este problema é proposto um novo projeto que modifica a interface original para que todos os pinos sejam realocados sem a perda de funcionalidade dos circuitos. Alguns circuitos foram acrescidos ao projeto original para incluir recursos que pretendem melhorar principalmente a questão da economia de energia. Alguns pinos são realocados de posição somente para facilitar a manipulação pelo programa de controle como é o caso da troca do pino mais significativo com o menos significativo e implementação do barramento de endereços na porta 2 do microcontrolador. Todas as modificações no projeto são mostradas no esquema elétrico no Anexo C. Estas modificações acrescentam um custo mínimo no projeto final.

6.1.1 Modificações no circuito do Microcontrolador

Para acesso aos registradores do PRISM MAC são necessários 6 linhas de endereço. Para a leitura de todos os endereços da tabela *CIS* são necessários 8 linhas de endereço e para o acesso ao endereço do registrador de reset são necessários 10 linhas de endereço. Se considerarmos que a leitura da tabela *CIS* é apenas necessária para descobrir o endereço de reset e que apenas uma leitura parcial é suficiente podemos ignorar os seus dois últimos bits e considerar para o acesso a ela apenas 6 bits. O endereço de reset possui os quatro bits mais significativos com valores iguais e portanto podem ser curto circuitados entre si

com o sétimo bit mais significativo. Esta simplificação reduz a quantidade de pinos de endereços a serem utilizados de 10 pinos para 7 pinos.

Para acesso do modo I/O foi necessário a inclusão dos sinais IORD, IOWR, REG e RESET. A necessidade de inclusão destes sinais precisou que fosse acrescido um outro circuito no projeto original de modo a expandir os pinos do microcontrolador. Foi utilizado para isto um circuito decodificador de 3 linhas para 8 linhas *(3:8)*. A escolha dos sinais teve que ser feita de forma que um não fosse dependente do outro e que não ocorresse ao mesmo tempo, caso contrário não seria possível inclui-lo na linha de decodificação. Como os sinais de controle de escrita e leitura tanto do modo memória quanto no modo I/O são indepedentes e não ocorrem ao mesmo tempo, eles puderam ser incluidos nas linhas de decodificação. A tabela de decodificação dos sinais de controle de escrita e leitura é mostrada na tabela 6.1. Quando a combinação das entradas é atingida, apenas uma saída é ativada em nível lógico baixo enquanto todas as outras permanecem em nível lógico alto.

DEC_C	**DEC_B**	**DEC_A**	**Saída(L)**
0	0	0	RD#
0	0	1	WR#
0	1	0	RESET
0	1	1	LED1
1	0	0	LED2
1	0	1	-IOWR
1	1	0	-IORD
1	1	1	Enable_CF

Tabela 6.1: Tabela de expansão dos sinais de controle do microcontrolador

Uma chave para permitir que o paciente acione em caso de emergência ou quando quiser enviar os dados para o médico também foi acoplada ao projeto. Esta chave é ligada no pino de interrupção *INT0* que é acionada para alterar o estado do monitor cardíaco para *ativo* e proceder com a abertura de conexão no servidor e envio dos dados.

6.1.2 Modificações no circuito de alimentação

Como a interface de rede sem fio Compact Flash é alimentada com 3,3V há necessidade de se mudar esta alimentação dos 5V originais para 3,3V. Para isto é usado um regulador de tensão que converte os 5V para 3,3V. Um circuito que corta a alimentação do cartão Compact Flash também é previsto de forma que seja possível economizar a energia que seria utilizada para alimentar o cartão nos momentos em que ele não é utilizado. O sinal `Enable_CF` mostrado na tabela 6.1 é encarregado de comandar o desligamento da alimentação do cartão CF.

6.2 Conclusões

Considerando que inicialmente o projeto base para os trabalhos não tinha sido projetado para o funcionamento para uma rede sem fio, a meta de se transformar ou desenvolver um circuito que atendesse aos pré requisitos do projeto foi desafiadora e motivou a busca por soluções que contornassem o problema mantendo-se o mesmo hardware elaborado. Foi necessário um grande esforço e exploração dos recursos de hardware e software, principalmente do microcontolador e da interface de rede sem fio para que a maioria do que foi proposto pudesse ser implementada.

Considera-se desta forma que os resultados obtidos no desenvolvimento dos trabalhos foram muito satisfatórios. Conseguiu-se mostrar através do capítulo 5 que é possível para um projeto até então limitado em termos de comunicação ser integrado à Internet ainda que não tenha sido projetado para tal. Algumas modificações ainda são necessárias para o completo funcionamento do sistema mas estas agregam pouco tamanho ao circuito e representam um custo irisório diante do resto do projeto.

As aplicações práticas do projeto foram testadas e comprovaram que ele pode ser utilizado como um sistema de monitoramento remoto de pacientes necessitando de alguma pequena melhoria no circuito de alimentação. O *Device Driver* desenvolvido em conjunto com a pilha TCP/IP utilizada permitiram integrar o Monitor Cardíaco de modo simples e eficiente à Internet podendo este software desenvolvido ser utilizado para outros sistemas embutidos que demandem necessidades semelhantes ao deste projeto.

6.3 Planos e trabalhos futuros

Alguns aspectos não foram cobertos no desenvolvimento deste trabalho e desta forma surgem como oportunidade para exploração em trabalhos futuros. A seguir são apontadas as principais oportunidades para continuidade dos trabalhos e algumas oportunidades de pesquisa:

- **Segurança:** Apesar de oferecer suporte à segurança através dos protocolos *WEP(Wired Equivalent Privacy)* e *WPA(Wi-Fi Protected Access)*, não foi objetivo de trabalho explorar os aspectos de segurança da aplicação o que desta forma contribui com um importante aspecto a ser pesquisado e implementado.

- **Implementação de aplicações para tratamento e disponibilização do dados enviados pelo monitor cardíaco:** o trabalho se preocupou mais com o lado cliente da comunicação e não foi elaborado um sistema para tratamento dos dados recebidos pelo servidor. Este sistema poderia ser implementado de forma a poder disponibilizar os dados através de gráficos de *ECG* e poder tratar dados enviados de vários pacientes.

- **Implementação de outros protocolos da família TCP/IP:** Para ser utilizado em uma rede sem fio sem a necessidade de um IP fixo é necessário a implementação

do protocolo *DHCP*. Outros protocolos de aplicação tais como o *SMTP* podem ser implementados permitindo, por exemplo, o envio de emails diretamente para o médico do paciente.

- **Melhoria do Circuito de alimentação:** é necessário rever o circuito de alimentação para que consiga suportar corretamente o consumo da placa *ECG*.
- **Uso de ASIC atualizado:** Verificar alternativa de uso de outro ASIC mais avançado, com mais recursos e que consuma menos energia uma vez que o ASIC utilizado no projeto do monitor cardíaco pode encontrar-se obsoleto.

Devido ao reduzido tempo para fechamento dos trabalhos, não foi possível montar e testar a nova versão de projeto do monitor cardíaco que incorpora as funções de aquisição das amostras de *ECG*. Isto impediu que resultados mais reais fossem mostrados e analisados. Como continuidade dos trabalhos, pretende-se implementar o circuito projetado, testar e analizar os dados para permitir que novas conclusões sejam tiradas podendo se ter conteúdo de pesquisa suficiente para publicação de artigos em congressos e conferências afins com a área do trabalho desenvolvido.

Referências Bibliográficas

[apa] The apache software foundation. Disponível em http://www.apache.org.

[AS] Inc AbsoluteValue Systems. linux-wlan. http://www.linux-wlan.org/.

[Ben02] Jeremy Bentham. *TCP/IP Lean Web Servers for Embedded Systems*. CMP Books, 2nd edition, 2002.

[Ben03] Etienne Beneteau. Web tcp/ip solutions with atmel c51 flash microcontrollers. Technical report, Atmel, 2003.

[Bra02] J. Brady. Build your own 8051 web server. *Circuit Cellar*, (146), September 2002.

[Can01] Tom Cantrell. I-way the hard way. *Circuit Cellar*, October 2001.

[Can02] Tom Cantrell. I-way the hard(ware) way. *Circuit Cellar*, May 2002.

[Com03] Compact Flash Association. *CF+ and Compact Flash Specification Revision*, 2.0 edition, Maio 2003. Disponível em http://www.compactflash.org.

[Con01] Júlio César Dillinger Conway. Monitor de sinais vitais multiparamétrico vestíveis, July 2001.

[Cor98] Intel Corporation. *Embedded Microcontrollers - Databook*, January 1998.

[Cor03] BroadCom Corporation. Ieee802.11g - the new mainstream wireless lan standard. *Disponível em http://www.54g.org*, July 2003.

[Cyl04] I. Cyliax. Wi-fi sunlogger. *Circuit Cellar*, (172), November 2004.

[Dan04] A. Dannenberg. Slaa137a - msp430 internet connectivity. *Texas Instruments*, February 2004.

[Dev03] Analog Devices. *ADuC834 Datasheet*, 2003.

[DM04] Matt Welsh et al David Malan, Thaddeus Fulford-Jones. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. *International Workshop on Wearable and Implantable Body Sensor Networks*, April 2004.

[Dum03] A. Dumkels. *uIP 0.9 Reference Manual*, July 2003.

[Dun03] Adam Dunkels. Full tcp/ip for 8-bit architectures. Proceedings of the first international conference on mobile applications, systems and services (MOBISYS), 2003.

[Ead04] F. Eady. Tcp/ip stack solution - a detailed look at the cmx-micronet. *Circuit Cellar*, (172), November 2004.

[Ead05] F. Eady. Embedded wi-fi with trendnet. *Circuit Cellar*, (174), January 2005.

[Fer05] Geraldo Antônio Ferreira. Aplicações midp em aparelhos móveis celulares e monitoramento remoto de bio-sinais: considerações e desenvolvimento de uma solução, August 2005.

[FLCJ03] C. J. N. Coelho F. L. C. Junior. Monitor cardíaco. Technical report, DCC/UFMG, 2003.

[For05] Débora Fortes. A explosão das redes sem fio. *Revista Info especial WI-FI Coleção 2005*, page 14, 2005.

[Ger03] Vadim Gerasimov. Every sign of life, June 2003.

[IEE99] IEEE. *Ansi/IEEE Std 802.11. Part 11 - Wireless Lan Medium Access Control (MAC) and Physical Layer (Phy) Specification*, 1999.

[IM02] IBM and Montavista. Dynamic power management for embedded systems, November 2002.

[Inc01] Intersil Americas Inc. *HFA3842 Data Sheet*, June 2001.

[Inc02] Intersil Americas Inc. *PRISM Driver Programmers Manual*, 2.30 edition, June 2002.

[kad] Kwiknet tcp/ip stack. http://www.kadak.com/tcp_ip/tcpip.htm.

[KL04] et al Konrad Lorincz, David Malan. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response*, Oct-Dec 2004.

[LLP] B. S. Davie L. L. Peterson. *Computer Networks - A Systems Approach.* Morgan Kaufmann Publishers, 2nd edition.

[Loe99a] M. Loewen. An724 - using picmicro mcus to connect to internet via ppp. *Microchip Technology Inc*, 1999.

[Loe99b] M. Loewen. Internet appliance interface. *Circuit Cellar*, (108), July 1999.

[Man98] S. Mann. Definition of wearable computer. *http://wearcomp.org/wearcompdef.html*, May 1998.

[Max04] Maxim. *DS80C400 Network Microcontroller*, 2004.

[mur] Keil c51/ 8051 port of adam dunkels' uip v0.9 tcp/ip stack. http://members.iinet.net.au/ṽanluynm/.

[pcma] Linux pcmcia information page. http://pcmcia-cs.sourceforge.net/.

[PCMb] PCMCIA. Pcmcia home page. http://www.pcmcia.org.

[Pea05] G. Peacock. Ip and ethernet interfaces. February 2005.

[Pen04] Alex Pentland. Healthwear:medical technology becomes wearable. *Computer Magazine*, (37), May 2004.

[Raj02] N. Rajbharti. The microchip tcp/ip stack. *Microchip Technology Inc*, 2002.

[RD03] Jonathan Gips Alex Pentland Rich DeVaul, Michael Sung. Mithril 2003: Applications and architecture. 7th IEEE International Symposium on Wearable Computers (ISWC), 2003.

[Req81a] Request for Comment. *RFC 791 - Internet Protocol*, Setembro 1981.

[Req81b] Request for Comment. *RFC 792 Internet Control Message Protocol*, Setembro 1981.

[Req81c] Request for Comment. *RFC 793 - Transimission Control Protocol*, Setembro 1981.

[Req89] Request for Comment. *RFC 1122 - Requirements for Internet Hosts - Communication Layers*, October 1989.

[Req94] Request for Comment. *RFC 1661 - The Point-to-Point Protocol (PPP)*, Julho 1994.

[RR00] Steve Humberd Rodger Richey. Embedding picmicro microcontrollers in the internet. *Microchip Technology Inc*, 2000.

[SBA+01] Tajana Simunic, Luca Benini, Andrea Acquaviva, Peter Glynn, and Giovanni De Micheli. Dynamic voltage scaling and power management for portable systems. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 524–529. ACM Press, 2001.

[Shr] H. Shrikumar. Ipic - a match head sized webserver. Technical report, CS/UMASS. Disponível em http://www-ccs.cs.umass.edu/ shri/iPic.html.

[SVGM00] Tajana Simunic, Haris Vikalo, Peter Glynn, and Giovanni De Micheli. Energy efficient design of portable wireless systems. In *ISLPED '00: Proceedings of the 2000 international symposium on Low power electronics and design*, pages 49–54. ACM Press, 2000.

[Tec01] Welch Allyn OEM Technologies. *ECG ASIC Datasheet*, May 2001.

[Tou04] Jean Tourrilhes. *Linux Wireless LAN Howto*, August 2004.

[TRFFJW04] Gu-Yeon Wei Thaddeus R. F. Fulford-Jones and Matt Welsh. A portable, low-power, wireless two-lead ekg system. pages 2141–2144. 26th IEEE EMBS Annual International Conference, 2004.

[uBM99] T. Šimunić, L. Benini, and G. D. Micheli. Event-driven power management of portable systems. In *International Symposium on System Synthesis*, pages 18–23, 1999.

[xbo] Motes, smart dust sensors, wireless sensor networks. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[YHL00] Giovanni De Micheli et al Yung-Hsiang Lu, Tajana Simunic. Quantitative comparison of power management algorithms. In *Design Automation and Test in Europe*, pages 20–26. Stanford University, March 2000.

[YHL01] Giovanni de Micheli Yung-Hsiang Lu. Comparing system-level power management policies. *IEEE Design & Test of Computers*, pages 10–19, March 2001.

Apêndice A

Protocolos TCP/IP

A.1 Protocolo IP

O protocolo IP [Req81a] é a base para o funcionamento da Internet. Um dos motivos para isto é porque ele implementa a característica da heterogeneidade [LLP] onde é possível que usuários de um tipo de rede são capazes de se comunicar com outros de outros tipos de rede.

O protocolo IP também suporta o príncipio da escalabilidade onde é previsto, principalmente na versão 6, o crescimento da Internet que dobra de tamanho a cada ano.

Basicamente, são implementados no protocolo IP as funcionalidades de endereçamento da rede e fragmentação dos dados. O endereçamento da rede permite verificar dentre outros, se os dados recebidos da rede são destinados às camadas superiores. Além disto permite identificar todos os dispositivos conectados à internet.

Já a fragmentação dos dados permite implementar a característica da heterogeneidade, fazendo com o que o dado, dependendo do tamanho seja fragmentado para que possa ser transmitido por interfaces de rede que comportem quantidades de dados que se encaixam no tamanho especificado. Quando o dado é recebido neste ultimo caso, o protocolo implementa a reconstrução *(reassembly)* dos pacotes de forma a serem entregues corretamente para o protocolo TCP.

Apesar de ser largamente utilizado, este protocolo é baseado em datagramas e portanto não garante confiabilidade na entrega dos pacotes.

A.1.1 Estrutura do Protocolo

O protocolo IP, conforme mostrado previamente na figura 2.4 possui um campo para o cabeçalho e um campo para os dados. O cabeçalho da versão 4 possui vários subcampos conforme mostrado na figura A.1.

Embora a especificação do protocolo seja estritamente intolerante em relação a implementação de alguns campos, como por exemplo o de *Opções*, Conforme foi visto no

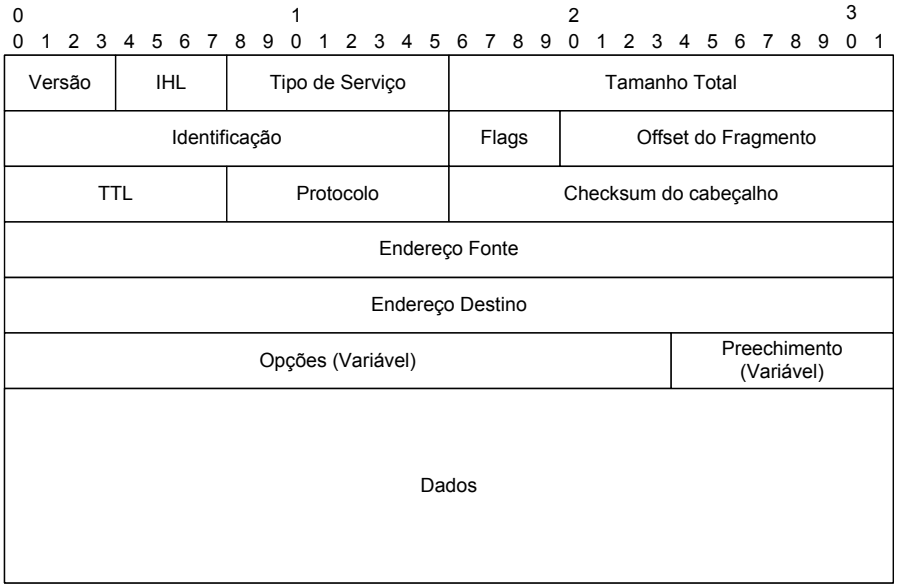


Figura A.1: Estrutura do Protocolo IPv4

trabalho realizado, as implementações de pilhas TCP/IPs para sistemas embutidos fazem diversas suposições no sentido de simplificar o desenvolvimento dos protocolos e reduzir o tamanho do código.

Diversas soluções implementadas pelo protocolo IP podem ser omitidas sem degradação de performance com o intuito de otimização do código para dispositivos embutidos.

A.2 Protocolo ICMP

O protocolo ICMP [Req81b] auxilia o protocolo IP na geração de mensagens de erros e diagnósticos através de pesquisas. Ele é muito útil devido a ausência de confiabilidade do protocolo IP e ajuda a geração de informações sobre a performance do protocolo IP em determinada rede.

Para o protocolo ICMP são implementadas diversas mensagens que são enviadas ao remetente quando um roteador ou um servidor ficou impossibilitado de processar um datagrama IP corretamente, por exemplo. Ele também pode indicar se determinado servidor pode ser alcançado pela rede ou não.

A implementação e uso mais comum do protocolo ICMP é a de pesquisa através de mensagens com eco (ECHO). Ela serve para verificar se um servidor com determinado número IP está conectado à rede e ainda provê alguns dados relevantes sobre o estado deste servidor na rede tais como tempo de retorno e estatísticas sobre o recebimento ou não dos dados pelo servidor de destino.

O funcionamento do retorno de mensagens por eco é simples. Um pacote do tipo

ICMP(06h) é enviado com a opção ECHO REQUEST (08h) com um quantidade de dados aleatória e variável dependendo de por exemplo, qual sistema operacional está sendo utilizado. O recebedor do pacote ICMP decodifica a mensagem, verifica que ela é uma requisição de eco, calcula o checksum dos dados recebidos e caso esteja correto então retorna uma outra mensagem do tipo ICMP com o mesmo conteúdo de dado e com a opção ECHO REPLY (00h). A utilização do protocolo ICMP na modalidade descrita anteriormente é largamente utilizada por meio do comando *ping* existente na maioria dos sistemas operacionais.

A.3 Protocolo TCP

O protocolo TCP [Req81c] foi criado inicialmente com o intuito de prover robustez para sistemas militares existentes na época que não ofereciam confiabilidade e eram muito sujeitos a problemas de congestionamento. Por ser um protocolo que suporta a comunicação de duas aplicações finais ele é comumente chamado de protocolo fim a fim (*end-to-end*).

Ao contrário do protocolo IP, para conseguir implementar a sua principal característica que é a confiabilidade, o protocolo TCP é orientado a conexão por meio de um circuito virtual. Isto significa que para um servidor se comunicar com outro é preciso o estabelecimento de uma conexão através da troca de mensagens. Somente depois de estabelecida a conexão, os dois lados conectados podem trocar dados entre si. Quando terminado a troca de dados, ambos os lados podem escolher por terminar a conexão que é feita novamente com troca de mensagens.

Além de resolver o problema da confiabilidade e de congestinamento, o protocolo TCP implementa também outras funcionalidades desejadas para um protocolo de transporte. Ele implementa um controle de fluxo de dados, suporta mais de uma conexão e também garante entrega dos pacotes na ordem que foram enviados.

A.3.1 Confiabilidade

A confiabilidade no protocolo TCP é atingida através da implementação de diversas soluções. Primeiramente, cada sequência de bytes (*Byte stream*) possui um número de sequência que começa em algum número aleatório. A sequência de dados é divida em segmentos de acordo com o máximo tamanho do segmento (*MSS*) de cada conexão.

Para cada número de sequência transmitido, pode-se dizer que é enviado um reconhecimento *ACK* para confirmar o recebimento da sequência de dados enviada. Se o *ACK* não for recebido dentro de um certo intervalo de tempo (*Timeout*), o dado referente a sequência enviada é retransmitido. Quando o dado é recebido, o *ACK* é enviado para o mais alto número de sequência recebido. Na verdade, o *ACK* enviado como reconhecimento de um certo número de sequência significa o próximo número de sequência esperado para recepção.

No destinatário, os números de sequência são usados para corrigir ordem de segmentos

que possam ter sido recebidos fora de ordem e também para eliminar sequências de dados duplicados. Segmentos de dados são verificados adicionando-se um checksum a cada segmento transmitido e checando-o no receptor. Segmentos com problema em relação ao checksum são descartados. No caso de segmentos descartados, estes serão retransmitidos por causa do *timeout*. Tal situação é mostrada na figura A.2.

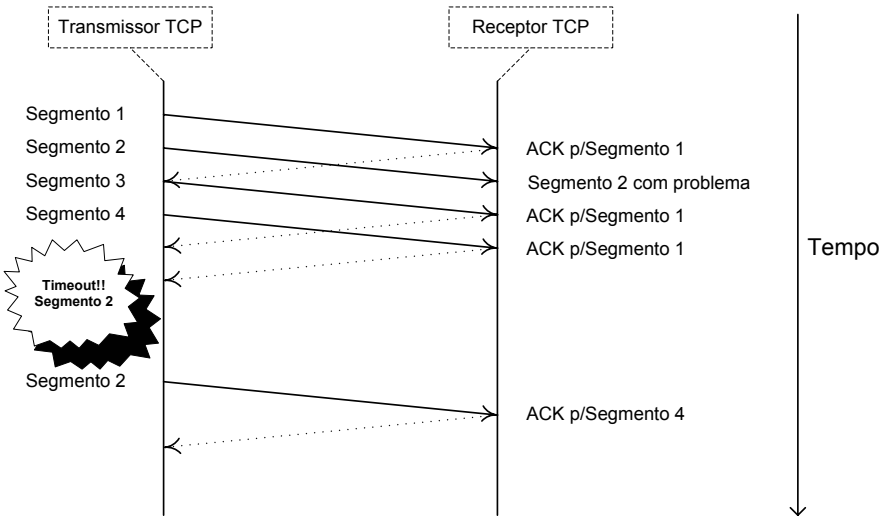


Figura A.2: Retransmissão de Segmento TCP por Recebimento errado de dado

A.3.2 Estimação do *Round-trip Time*

Um importante fator utilizado pelo protocolo TCP é a estimação do parâmetro RTT (*Round-trip Time*). O RTT é o tempo contado a partir de um pacote com determinado número de sequência ser enviado até o reconhecimento deste pacote ser recebido.

A estimação do RTT é importante porque ele é utilizado para cálculo do *timeout* de retransmissão dos dados. Se o RTT for muito menor do que o real, o transmissor irá fazer retransmissões desnecessárias congestionando a rede. Já se o RTT fo maior do que o real, o transmissor irá demorar mais tempo para retransmitir o dado o que irá comprometer o desempenho da conexão.

Devido a variabilidade das redes que compõem a Internet, os timeouts para retransmissão devem ser determinados dinamicamente. Para isto foi proposto o método original para estimação [Req81c] que se baseia em médias ponderadas de amostragens do RTT(medidos por meios de *ACKs* recebidos) e um parâmetro que suavisa o cálculo do RTT ajustando-o para detectar variações rápidas ou lentas.

A detecção de alguns problemas no algoritmo original levaram a proposição de outros algoritmos na literatura que introduziram novas variáveis no cálculo do RTT e refinaram algumas suposições que foram incorretamente previstas no método original.

A.3.3 Controle de Fluxo

O protocolo TCP fornece uma forma do receptor controlar a quantidade de dados que recebe do transmissor. Isto é possível retornando uma "janela" junto com cada reconhecimento *ACK* indicando uma faixa aceitável de números de sequência em torno do ultimo segmento recebido com sucesso. Esta janela indica uma quantidade de bytes que o transmissor pode transmitir. Desta forma o transmissor irá enviar apenas uma certa quantidade de dados que o receptor irá conseguir processar.

Com este príncipio é possível dois dispositivos com diferentes quantidades de memória poderem comunicar entre si sem comprometer a degradação dos dados. Este mecanismo de controle de fluxo é conhecido como janelas deslizantes (*Sliding Windows*).

A.3.4 Conexões

O TCP inicializa e mantém um estado da informação para cada sequência de dados. A combinação desta informação e outras tais como os números de sequência e tamanhos de janela são chamadas de conexão.

Quando dois processos desejam se comunicar, o protocolo TCP primeiro inicializa o estado da conexão para os dois lados. Como a comunicação deve ser confiável entre ambos os lados é implementado um mecanismo conhecido como *Three-Way Handshake*. Este mecanismo é baseado nos números de sequência e nos flags de estado de conexão e é usado para evitar erros na inicialização da conexão.

A.3.5 Estrutura do Protocolo

A figura A.3 ilustra a estrutura do protocolo TCP com o cabecalho e o espaço para os dados. Esta estrutura está acima do protocolo IP na pilha mostrada na figura 2.3.

O tamanho mínimo do cabeçalho TCP, assim como o IP, é de 20 bytes. Isto se não forem implementadas as opções do protocolo.

A.3.6 Máquina de estados TCP

De maneira a contribuir com a confiabilidade proposta pelo protocolo TCP, é implementado uma forma de criar, manter e fechar uma conexão entre dois nós da rede. Este processo é melhor representado através da máquina de estados TCP mostrado na figura A.4.

Seguindo um arquitetura do tipo cliente-servidor, o servidor, na abertura de uma conexão, requisita uma operação de *abertura passiva*, o que causa o diagrama se mover para o estado **LISTEN**. Em algum momento, o cliente abre uma conexão fazendo uma *abertura ativa* enviando um flag sinalizador de sincronismo de conexão(SYN) para o servidor. Neste momento o cliente passa para o estado **SYN-SENT**. Quando o servidor recebe o flag SYN, ele responde com um flag de reconhecimento ACK e solicita ao cliente um sincronismo de conexão enviando também um flag de sincronismo SYN. Ao enviar estes

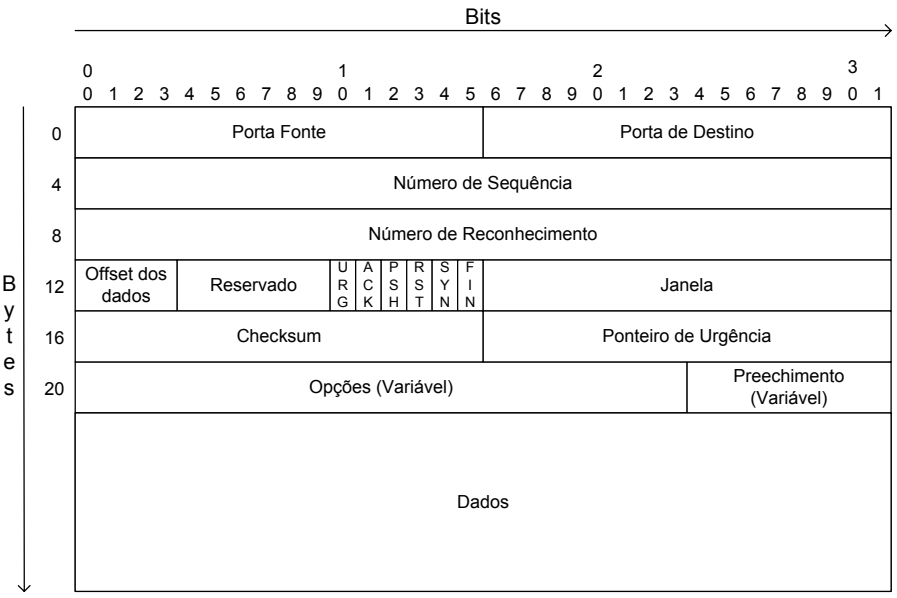


Figura A.3: Estrutura do Protocolo TCP

flags, o servidor passa para o estado **SYN-RCVD**. Ao receber os flags SYN+ACK, o cliente muda para o estado de estabelecimento de conexão, **ESTABLISHED** e envia um sinalizador de reconhecimento ACK para o servidor. Quando este ACK chega, o servidor passa também para o estado **ESTABLISHED** enviando outro ACK para o cliente. Este processo de abertura de abertura de conexão conforme mencionado anteriormente é chamado de *Three-way handshake*.

De fato, a abertura de conexão privilegia a arquitetura cliente-servidor onde ambos os lados da conexão partem do estado inicial **CLOSED** e a partir dai seguem estados diferentes até estabelecerem a conexão. É possível também que os dois lados da conexão estejam escutando a conexão no estado **LISTEN** e serem capazes de decidir se tornarão clientes abrindo a conexão ou permanecerão como um servidor esperando a requisição de uma conexão. Para isto, basta a partir do estado **LISTEN** que um dos lados faça o *Envio* de flag de sincronismo de conexão para o outro lado. Também é possível que os dois lados abram a conexão ao mesmo tempo sendo que para isto eles seguem os mesmos estados desde **CLOSED** passando por **SYN-SENT**, **SYN-RCVD** até estabelecerem a conexão no estado **ESTABLISHED**.

O protocolo TCP também trata problemas na abertura de conexão quando um dos lados não estiver sincronizado com o outro, quando um dos lados envia um flag para reset (RST) da conexão e o outro lado vai para o estado conhecido de **LISTEN**. Outros problemas tais como falta de sincronismo quando uma conexão já está está aberta e um dos lados por algum motivo perde o estado da conexão são também tratados. Neste caso, o lado que perdeu o sincronismo, pode conseguir detectar que o sincronismo foi perdido e envia para o outro o flag de reset. O outro lado por sua vez ao receber o reset, aborta

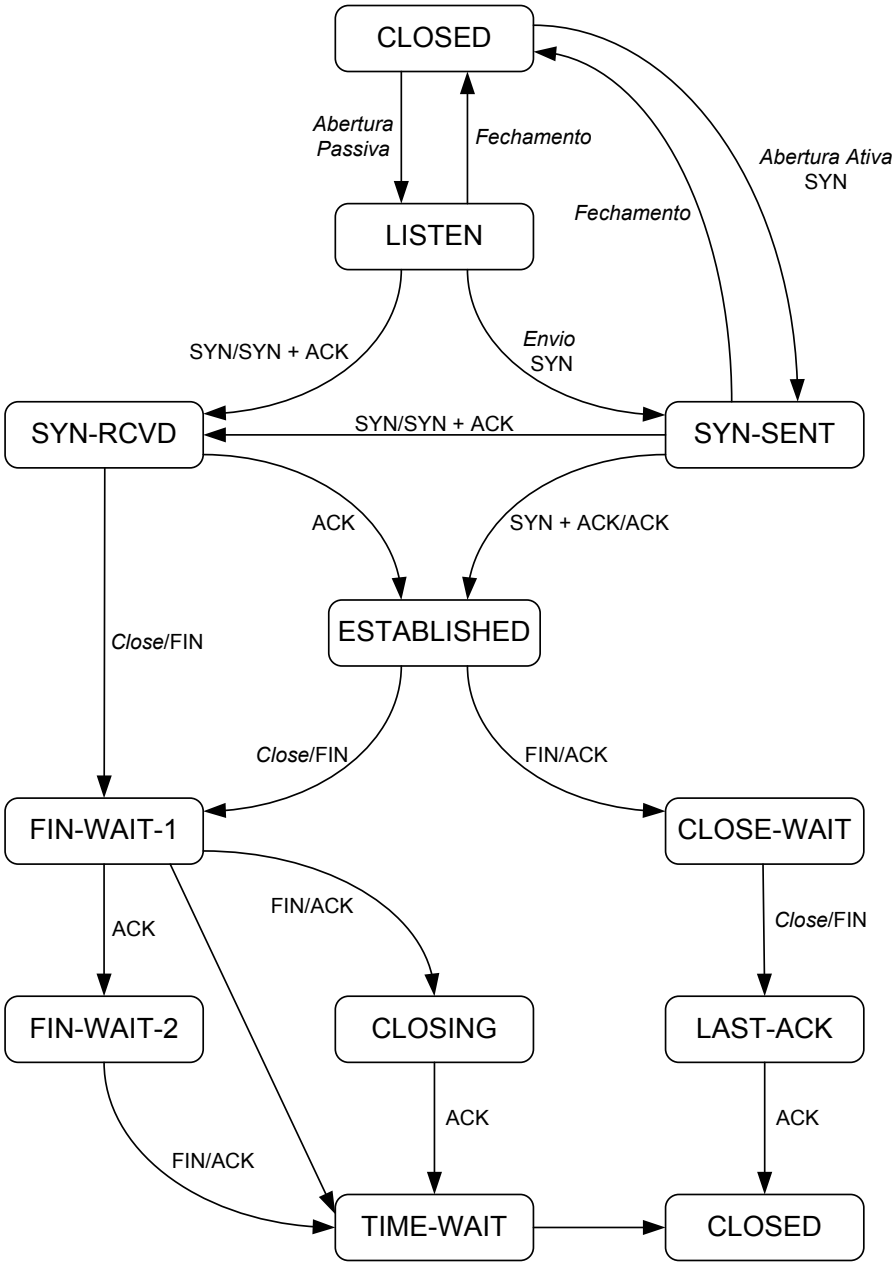


Figura A.4: Diagrama de transição de estados do Protocolo TCP

a conexão. Como pode-se perceber o flag de reset é utilizado em alguns casos onde um segmento aparentemente parece não pertencer a determinada conexão.

Ao ser estabelecida a conexão, Ambos os lados podem transmitir dados um para o outro sempre no formato de envio de uma lado do número de sequência e em resposta o envio pelo outro lado do flag de reconhecimento ativo e do número do reconhecimento para

a próxima sequência esperada.

O fechamento da conexão é um pouco mais complexo do que a abertura e estabelecimento da conexão porque é previsto que qualquer um dos lados pode fecha-la a qualquer momento ou ao mesmo tempo. Para facilitar o entendimento iremos das situações de fechamento iremos nomear o lado 1 como sendo um usuário local e o lado 2 como sendo a rede.

Quando o lado 1 solicita o fechamento da conexão, ele envia um flag sinalizador de finalização da conexão, FIN, indo para o estado **FIN-WAIT1**. O lado 2, recebe o flag FIN e reconhece-o enviando um reconhecimento ACK e indo para o estado **CLOSE-WAIT**. Recebendo o flag ACK, o lado 1 vai para o estado **FIN-WAIT2** e espera pelo lado 2 fechar a conexão. O lado 2 por sua vez envia para o lado 1 o seu flag para finalização de conexão FIN e vai para o estado **LAST-ACK**. Recebendo o flag FIN, o lado 1 vai para o estado **TIME-WAIT** e envia um flag de reconhecimento ACK por ter recebido o sinalizador de finalização da conexão. Quando receber o flag ACK, o lado 2 vai para o estado **CLOSED**. O lado 1 estando no estado **TIME-WAIT** espera por um tempo padrão que corresponde ao tempo máximo que um pacote IP sobrevive na rede (em torno de 2 minutos) para então possa passar para o estado **CLOSED**. Quando o lado 2 solicita o fechamento da conexão, repete-se o mesmo ciclo de estados citado anteriormente, porém, troca-se o lado 1 pelo lado 2 e vice-versa.

Quando ambos os lados fecham a conexão simultaneamente, os dois enviam o flag de finalização FIN um para o outro e vão para o estado **FIN-WAIT1**. Quando recebem o flag FIN em ambos os lados, enviam um reconhecimento ACK e vão para o estado **CLOSING**. Ao receberem o ACK em ambos os lados vão para o estado **TIME-WAIT**, esperando um tempo até fecharem totalmente a conexão no estado **CLOSED**.

Apêndice B

Redes sem fio e o Padrão 802.11

B.1 Tipos de rede sem fio

Para entendimento do funcionamento de uma rede sem fio do tipo 802.11 é necessário o conhecimento de seus principais componentes. Estes componentes são ilustrados na figura B.1.

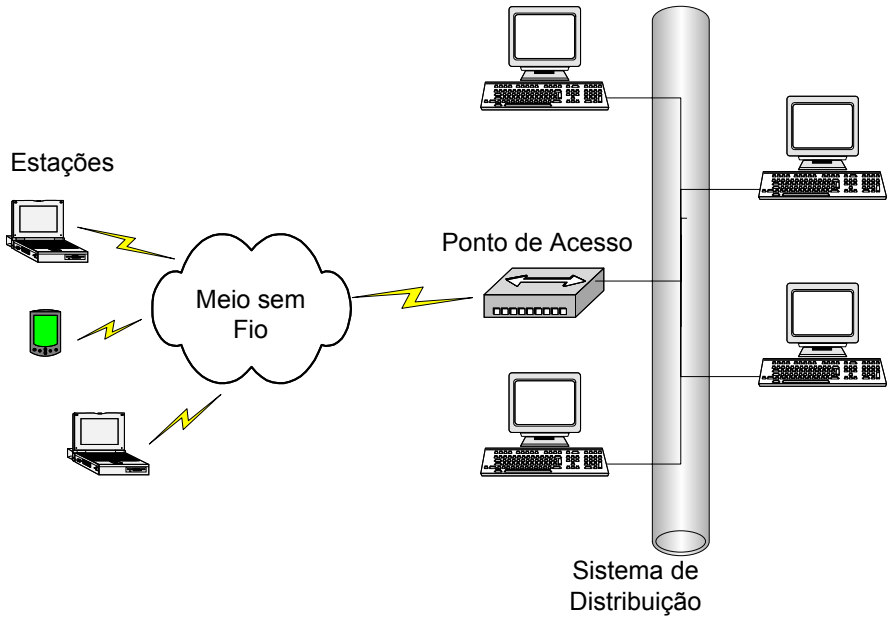


Figura B.1: Componentes de uma rede sem fio 802.11

1. **Sistema de distribuição** - Pode ser entendido como sendo uma rede usada para

comunicação dos dados fora da rede sem fio. Geralmente é uma rede do tipo Ethernet conectada à internet.

2. **Ponto de Acesso (AP)** - É uma estação que fornece acesso da rede sem fio ao sistema de distribuição além de atuar como uma estação da rede. Em outras palavras é a interface de uma rede sem fio com uma rede com fio.

3. **Meio sem Fio** - Meio que uma estação usa para trocar dados com a outra.

4. **Estação** - Dispositivo dotado de uma interface de rede sem fio que se comunica com outros através de uma rede sem fio.

Existem dois arranjos de redes sem fio que são utilizados dependendo do ambiente e da arquitetura de interligação entre elas. Elas são detalhadas a seguir.

B.1.1 Redes Independentes - *Ad Hoc*

Um conjunto composto de duas ou mais estações comunicando-se em um meio sem fio é chamado de conjunto de serviço básico ou *Basic Service set (BSS)*. Uma Rede sem fio independente nada mais é do que um BSS com duas ou mais estações que usam o meio sem fio para somente comunicarem-se diretamente umas com as outras. A figura B.2 mostra a arquitetura de uma rede sem fio independente.

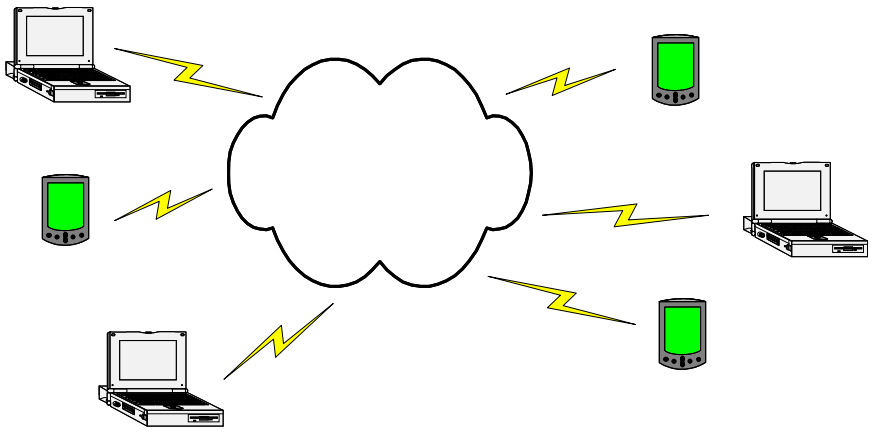


Figura B.2: Exemplo de uma Rede sem fio independente

Este modo de comunicação é chamado de *Independent Basic Service Set (IBSS)* e se caracteriza por um pequeno número de estações que ficam interligadas por um curto período de tempo geralmente para uma aplicação ou finalidade específica. Por estas características as redes sem fio independentes também são chamadas de *Ad Hoc*.

B.1.2 Redes Infraestrutura

As redes no modo infraestrutura além de utilizarem estações comuns interligadas pelo meio sem fio, também usam um ponto de acesso para interligar estas estações ao sistema de distribuição ou a outras redes sem fio. Neste tipo de rede, uma estação para se comunicar com a outra, não pode enviar dados diretamente para a outra. É necessário que os dados a serem enviados de uma estação para outra, independente se estiverem ligados em uma mesma rede sem fio ou não, passem sempre pelo ponto de acesso. Um exemplo de como seria uma rede do tipo infraestrutura é mostrado na figura B.3

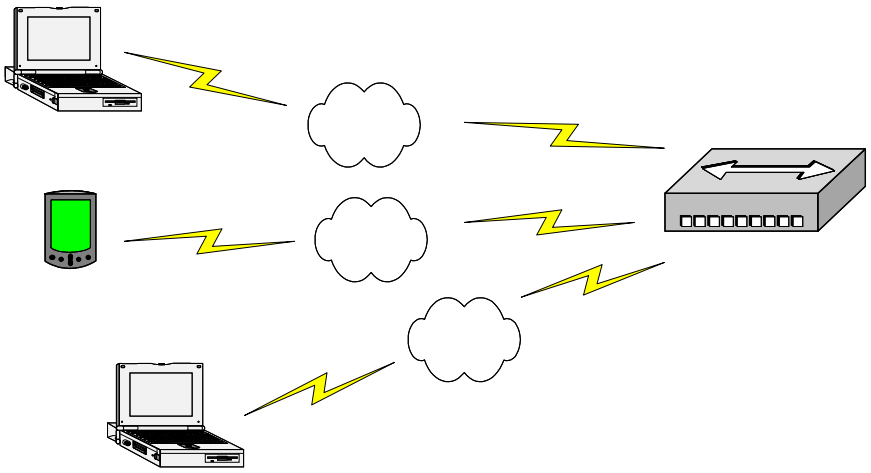


Figura B.3: Exemplo de uma Rede sem fio tipo Infraestrutura

Um conjunto de serviço básico do tipo Infra estrutura é caracterizado pela distância de cobertura de um ponto de acesso. Desta forma todas as estações dentro desta rede devem estar na faixa de alcançe do ponto de acesso. Dentro desta faixa de cobertura, as estações podem assumir qualquer distância para o ponto de acesso ou entre as estações.

Os pontos de acesso podem melhorar vários aspectos de comunicação em uma rede do tipo infra estrutura. Eles podem por exemplo detectar que uma certa estação está em modo Sleep*(power save)* e armazenar dados destinados à ela. As estações desta forma podem durar mais tempo na rede visto que otimizam o uso enviando dados e recebendo dados através de comunicação com o ponto de acesso somente quando necessário.

Em uma rede do tipo infraestrutura, uma estação para fazer parte da rede sem fio deve primeiro se *associar*. *Associação* é o processo pelo qual uma estação se agrupa a uma rede sem fio. Este processo é similar ao fato de se conectar um cabo de rede de um dispositivo a uma rede Ethernet. A estação deve se conectar a apenas um ponto de acesso que permite ou nega o acesso a rede sem fio baseado no conteúdo da requisição. Tipicamente, um ponto de acesso consegue servir até 20 estações.

B.2 Visão Geral dos serviços de rede 802.11

Em uma rede 802.11 existem serviços de forma a controlar diversos aspectos de funcionamento dos dispositivos na rede. A especificação do protocolo cita nove destes serviços. Seis deles são usados para operações de gerenciamento para que os pontos de acesso entreguem os dados corretamente e para identificar os nós da rede sem fio. Os outros 3 serviços servem para controlar o acesso à rede e para fins de confidencialidade.

Os serviços são resumidos na tabela B.1 e são classificados de acordo com o tipo como sendo de serviço de sistema de distribuição (*DSS*) ou serviço de estação (*SS*).

Um serviço de sistema de distribuição (*DSS*) conecta um AP em um sistema de distribuição *DS*. Um dos maiores papeis de uma AP é estender as funcionalidades de uma rede com fio para uma rede sem fio. Isto é possível através dos serviços de distribuição e integração. Outro aspecto importante do serviço de sistema de distribuição é o gerenciamento de estações em uma rede sem fio que é conseguido através dos serviços de associação, reassociação e disassociação.

Serviços de Estação (*SS*) são fornecidos pelas estações móveis ou pelas interfaces de rede sem fio dos pontos de acesso. As estações possibilitam a entrada e saída dos dados e para suportar este serviço são necessários a autenticação na rede para ser estabelecida uma associação à rede a qual ela pretende se integrar. Para proteger seus dados de invasores de redes sem fio, as estações podem utilizar o serviço de privacidade fornecido pelo protocolo.

B.3 802.11 MAC

Para o controle de acesso ao meio, o protocolo 802.11 utiliza um esquema semelhante ao utilizado nas redes Ethernet (*CSMA/CD*), só que ao invés de detecção de colisão, muito cara no meio sem fio, é utilizado a prevenção de colisão. Portanto o esquema utilizado é o *CSMA/CA* ou simplesmente *MACA*.

Este esquema tem a idéia básica de que para iniciar uma comunicação, duas estações devem trocar frames de controle. Estes frames, são iniciados a partir do transmissor que envia um comando de controle do tipo RTS (*Request To Send*) para a estação receptora. Este frame inclui um vetor de alocação de rede *NAV* que corresponde a um tempo que a estação pretende usar o meio. Neste momento todas as estações da rede sem fio próximas da estação transmissora recebem o frame RTS e registram o NAV contando-o até zero. Estas estações só poderão usar o meio novamente depois que o NAV for zero. A estação receptora ao identificar que o frame RTS é destinado a ela, retorna um outro frame de resposta do tipo CTS(*Clear To Send*)também com uma valor para o NAV. Todas as estações na vizinhança que escutam o frame CTS ficam em silêncio e não transmitem até que o NAV seja zero. Apenas estação transmissora após identificar que o frame CTS é para ela, inicia a transmissão dos dados. Após a transmissão, Todos os dados recebidos do transmissor são reconhecidas por um ACK pelo receptor.

O acesso ao meio sem fio é controlado por funções de coordenação. Semelhante ao protocolo Ethernet, o CSMA/CA implementa uma funcão distribuída de coordenação (*DCF*).

Serviço	**Tipo**	**Descrição**
Distribuição	DSS	usado em toda entrega dos dados para determinado endereço na rede
Integração	DSS	serviço usado na entrega dos dados para uma rede padrão 802.x forada rede sem fio
Associação	DSS	usado por uma estação para contactar uma AP e tentar fazer parte de uma rede sem fio
Reassociação	DSS	usado por uma estação para para mudar de AP e tentar fazer parte de uma outra rede sem fio. Acontece quando uma estação móvel detecta um baixo nível de sinal na AP conectado originalmente.
Disassociação	DS	Remove uma estação da rede.
Autenticação	SS	Usado por uma estação antes da associação para permitir a estação acessar determinada rede sem fio via AP. A AP autoriza ou não o acesso da estação à rede.
Desautenticação	SS	serviço para terminar com a autenticação de uma estação na rede. Como é um serviço anterior a associação, a desautenticação também disassocia uma estação da rede sem fio.
Privacidade	SS	Fornece proteção
Entrega MSDU[1]	SS	Serviço de obtenção dos dados de uma rede sem fio

Tabela B.1: Serviços utilizados pelo Protocolo 802.11

A *DCF* permite que multiplas estações independentes se comuniquem sem a necessidade de um controle central. Se é necessário uma janela de contenção livre ela pode ser implementada pelos pontos de acesso (APs) por meio da função de distribuição pontual (*PCF*).

O controle de acesso ao meio do protocolo 802.11 também implementa espaçamento entre os frames. São implementados quatro espaçamento entre frames sendo que 3 deles são usados para controle de acesso ao meio. Estes espaçamentos são de tamanho fixo independente da velocidade de transmissão. A idéia por trás destes espaçamentos é que

tempos diferentes criam diferentes níveis de prioridade dependendo do tipo de tráfego. Assim, um tráfego de alta prioridade não precisa esperar muito tempo após o meio ter se tornado desocupado. Desta forma se existe um tráfego de alta prioridade esperando para ocupar o meio, ele consegue acesso antes de um com prioridade mais baixa tentar acessa-lo. Os espaçamentos entre os frames possuem a seguinte denominação:

- Short Interframe Space (SIFS) - usada para transmissões de mais alta prioridade tais como CTS/RTS e ACKs;
- PCF Interframe Space (PIFS) - usada pela PCF durante a operação de contenção livre;
- DCF Interframe Space (DIFS) - menor tempo de espera para serviço baseado em contenção;
- Extended Interframe Space (EIFS) - usada somente em caso de erro na transmissão do frame. Não possui tamanho fixo.

Uma idéia geral dos tempos associados e comandos de controles utilizados para controle de acesso ao meio são ilustrados na figura B.4.

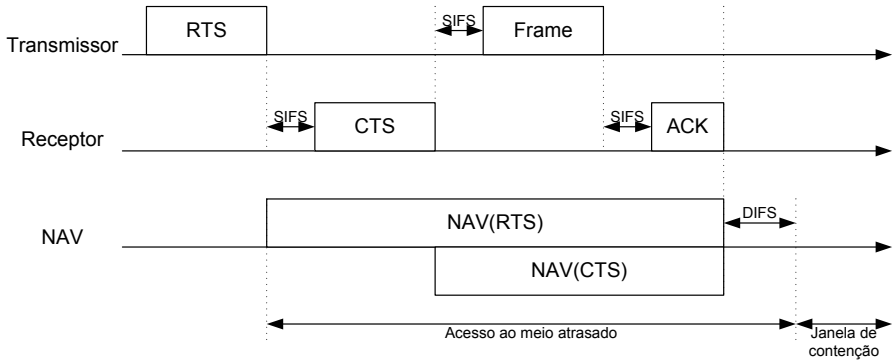


Figura B.4: Acesso ao meio usando NAV e tempos de espaçamentos entre os frames

Depois de esgotado o tempo DIFS, as estações que desejam transmitir entram em uma janela de contenção e é implementado o algoritmo *back-Off esponencial* para evitar colisão na transmissão dos dados.

B.4 Relações entre os serviços e Tipos de frames utilizados no protocolo 802.11

Em uma rede do tipo infra estrutura, podem existir diversos pontos de acesso (AP). Para distinguir uma AP da outra, cada uma usa um canal de comunicação e uma identificação.

Esta identificação é chamada de *SSID (Service Set Identifier)* que pode ser identificada por até 32 caracteres.

Para se conectar a uma rede, uma estação pode agir passivamente, técnica chamada de *passive Scanning*, escutando uma rede sem fio, esperando por uma sinalização (*beacom*) do ponto de acesso. Esta sinalização é enviada periodicamente pelas APs e inclui todas as informações necessárias para uma estação se conectar a elas incluindo o *SSID*, intensidade do sinal e velocidade de comunicação. A estação também pode se conectar a uma rede, procurando ativamente *Active Scanning* por pontos de acesso.

Após a etapa de procura, a estação pode se conectar à rede sem fio escolhendo a AP a qual deseja se conectar, especificando o *SSID* do ponto de acesso correspondente. A estacão também pode ser configurada em alguns casos para se conectar ao ponto de acesso com a intensidade de sinal mais forte que foi encontrado.

A fase de conexão de uma estação com um ponto de acesso compreende duas etapas. A primeira etapa é o serviço de autenticação. Esta etapa é iniciada através de um pedido de autenticação feito pela estação destinado a AP. A AP recebe o pedido e baseado em sua configuração pode requerer ou não alguma chave de segurança baseada no protocolo de segurança *WEP (Wireless Equivalent Privacy)* para permitir acesso à rede. O ponto de acesso também pode ser configurado para autenticar apenas certos endereços MAC cujo processo é denominado filtragem de endereço.

Após a autenticação é necessário que a estação se associe ao ponto de acesso que o autenticou. Isto é possível através do serviço de associação onde a estação envia um pedido de requisição de associação (*Association request*) para a AP e esta responde com uma resposta de associação (*Association Response*). A partir da associação, uma estação pode utilizar o ponto de acesso para se comunicar com o sistema de distribuição.

Em uma rede do tipo *Ad-Hoc* deve-se iniciar uma comunicação ponto a ponto configurando-se o mesmo canal e *SSID*. Como não é necessário a presença de um ponto de acesso para a comunicação também não é preciso a utilização dos serviços de autenticação e associação.

B.5 Formato do quadro 802.11

O formato do quadro genérico 802.11 é composto de três partes a saber:

- Cabeçalho - contém informações sobre frames tais como controle, duração, sequência.
- Dados - campo variável que comporta os dados.
- FCS - Sequência de Checagem do Frame que nada mais é do que o CRC-32 do quadro.

A figura B.5 ilustra o formato genérico do quadro 802.11 destacando também os campos do frame de controle:

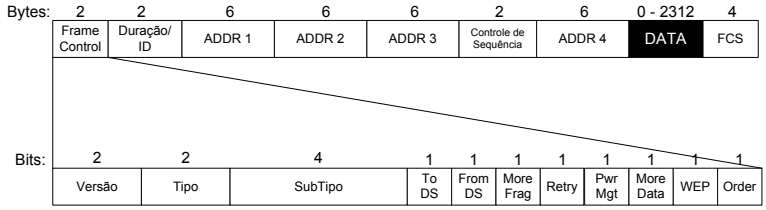


Figura B.5: Formato do frame 802.11 com destaque para o campo de controle do frame

B.5.1 Frame de controle

O frame de controle faz parte do cabeçalho do protocolo 802.11. Uma de suas principais funções é estabelecer qual tipo de frame do protocolo. Isto é feito pela configuração dos campos tipo e subtipo. No campo tipo é definido se o frame é do tipo controle(01b), gerenciamento(00b) ou dados(10h). Já no campo subtipo é definido, por exemplo, se o frame de gerenciamento é do tipo requisição de associação(0000b) ou se o frame de controle é do tipo RTS(1011b). Ao todo são 25 combinações de tipos e subtipos que definem os diversos quadros do protocolo 802.11.

Outra importante atribuição do frame de controle são os bits *From DS* e *To DS* que controlam a relação dos endereços em relação ao sistema de distribuição. De acordo com a combinação destes dois bits pode-se entender a aplicação dos endereços de MAC e a origem ou destino dos dados em certo tipo de configuração de rede sem fio. A tabela B.2 ilustra as possibilidades de combinação dos bits From DS e To DS.

	To DS = 0	**To DS = 1**
From DS = 0	Todos os frames de gerenciamento, controle e dados em uma rede Ad-Hoc.	Frames de dados transmitidos de uma estação sem fio em uma rede do tipo infra estrutura
From DS = 1	Frames de dados recebidos de uma estação sem fio em uma rede do tipo infra estrutura.	frames de dados de um ponto de acesso para outro

Tabela B.2: Significado da combinação dos bits From DS e To DS do quadro de controle

Existem ainda no frame de controle, bits que controlam o funcionamento de algumas características do protocolo tais como:

- bit Retry - quando setado indica retransmissão do frame anterior. Usado em frames

de dados ou gerenciamento.

- bit Pwr Mgnt - quando setado indica que estação está no modo de gerenciamento ou economia de energia.
- bit More Data - quando setado indica para a estação que está em modo de economia de energia que existem dados destinados a ela armazenados no ponto de acesso.
- bit WEP - quando setado indica que os dados foram codificados usando o algoritmo WEP.

B.5.2 Duração/ID

Este campo tem três funções distintas dependendo da combinação de seus dois bits mais significativos. Com o bit mais significativo em 0 ela representa o *NAV* que corresponde ao número em microsegundos esperado que o meio ficará ocupado pela estação durante uma transmissão de dados. Para frames transmitidos durante o período livre de contenção (*Contention free*) o valor do campo de duração é fixo em 32768. Para frames de controle do tipo *PS-Poll*, o campo de duração representa a identidade da associação *AID*. O valor do *AID* está na faixa de 1-2007.

B.5.3 O campo de Endereços

Conforme mostrado na figura B.5, existem quatro campos de endereços de MAC no protocolo 802.11. Cada endereço é um número de 48 bits e os campos são numerados porque são utilizados de maneira diferente dependendo do tipo do frame. Além disso, dependendo do frame nem todos os campos de endereço são utilizados.

Como regra geral, pode-se dizer que o endereço 1 é usado pelo receptor, o endereço 2 é usado pelo transmissor e o endereço 3 é usado para filtragem pelo receptor. Os endereços utilizados no MAC podem assumir as seguintes representações:

- Endereço de Destino (DA) - endereço que corresponde ao recebedor final que irá processar os dados pelos protocolos de nível superior ao de interface de rede.
- Endereço fonte (SA) - endereço que identifica a fonte da transmissão.
- Endereço Receptor (RA) - Endereço que identifica qual AP deve processar o frame. Caso seja uma estação sem fio, o endereço receptor é o mesmo do endereço destino.
- Endereço Transmissor (TA) - usado para identificar qual AP que transmitiu o dado para o meio sem fio.
- Identificação do Conjunto de serviço Básico (BSSID) - identifica redes sem fio locais. Em redes do tipo infra estrutura, o BSSID é o endereço MAC da interface de rede do AP. Em redes Ad-Hoc são gerados BSSID aleatórios.

Nos frames de dados a ordem dos endereços irá depender dos bits From DS e To DS provenientes do frame de controle. Os endereços de acordo com o valor destes dois bits são mostrados na tabela B.3.

Função	**To DS**	**From DS**	**Addr1**	**Addr2**	**Addr3**	**Addr4**
IBSS(rede independente)	0	0	DA	SA	BSSID	-
De um AP (Infraestrutura)	0	1	DA	BSSID	SA	-
Para um AP (Infraestrutura)	1	0	BSSID	SA	DA	-
Ponte	1	1	RA	TA	DA	SA

Tabela B.3: significado dos Endereços de acordo com os bits From DS e To DS

B.5.4 Número de sequência

O protocolo 802.11 utiliza soluções de ambos os protocolos TCP e IP no campo número de sequência. Este campo possui 16 bits e é dividido em número de sequência(12 bits) e número de fragmento(4 bits). Ele é utilizado para os quadros de dados e de gerenciamento.

Semelhante ao protocolo TCP é utilizado o número de sequência que é o mesmo em caso de retransmissões ou em caso de fragmentos de um mesmo pacote de dados. A diferença aqui é que o número de sequência que se inicia em zero é incrementado de um em um e vai até 4096 enquanto que no protocolo TCP esta sequência também é incrementada de um em um mas é iniciada a partir de um número aleatório de 32 bits.

Conforme utilizado no protocolo IP, também é usado fragmentação de dados que é incrementado de um em cada fragmento e pernanece constante em cada retransmissão.

B.5.5 Campo de dados

O campo de dados possui tamanho variável e carrega informação que será utilizada pelos protocolos superiores ao protocolo de interface de rede. Caso o protocolo de segurança *WEP* seja usado este campo é aumentado em mais 8 bytes através a inclusão dos subcampos *Initialization Vector(IV)* e *Integrity Check Value(ICV)* conforme mostrado na figura B.6.

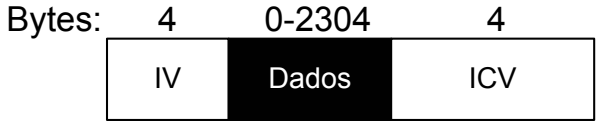


Figura B.6: Tamanho total do campo de dados do protocolo 802.11

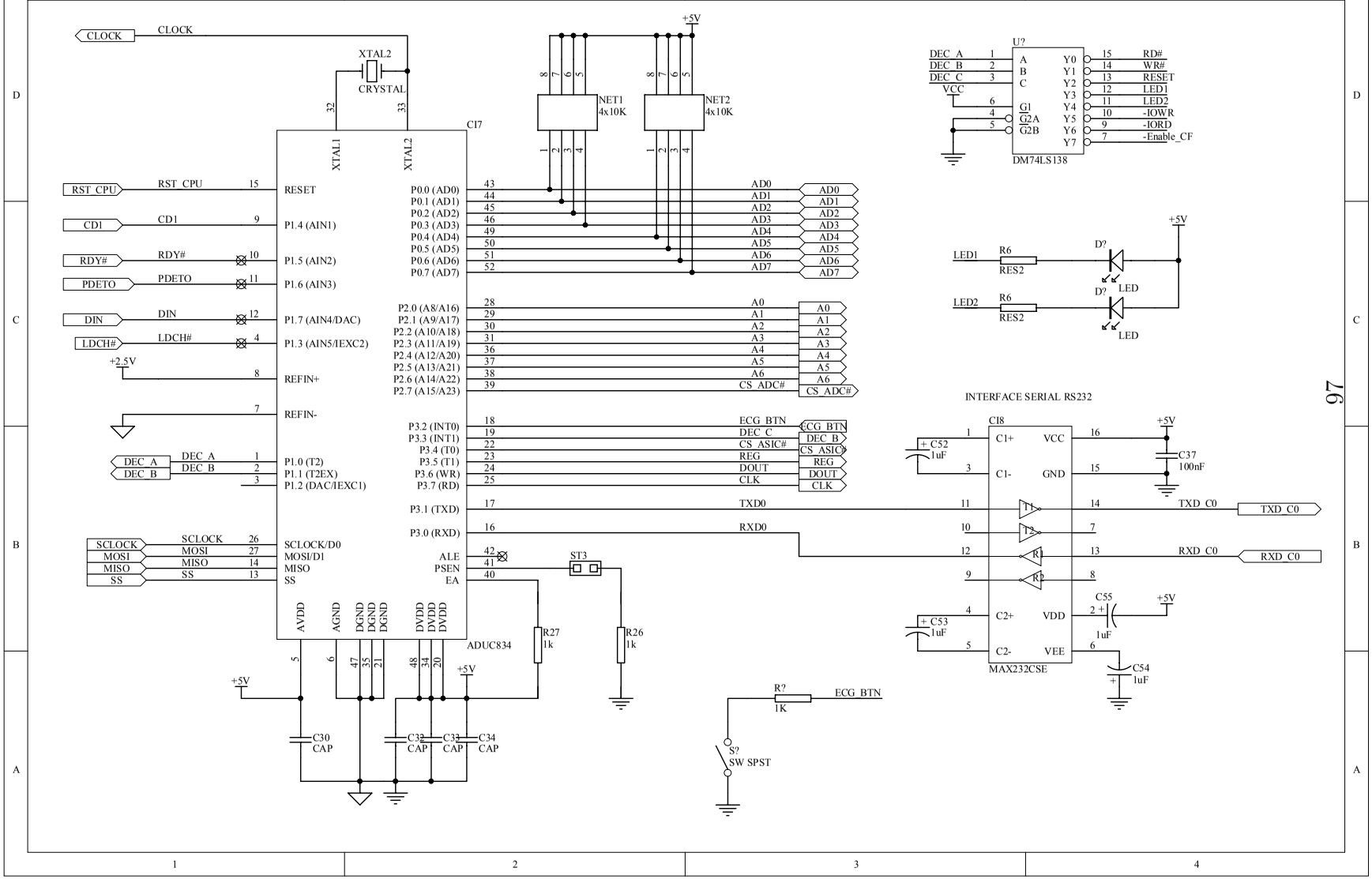
B.5.6 Campo Frame Control Check (*FCS*)

O campo FCS possui 32 bits e é corresponde ao cálculo do tipo CRC de 32 bits. O FCS é calculado em todo o cabeçalho MAC e também pelo campo de dados.

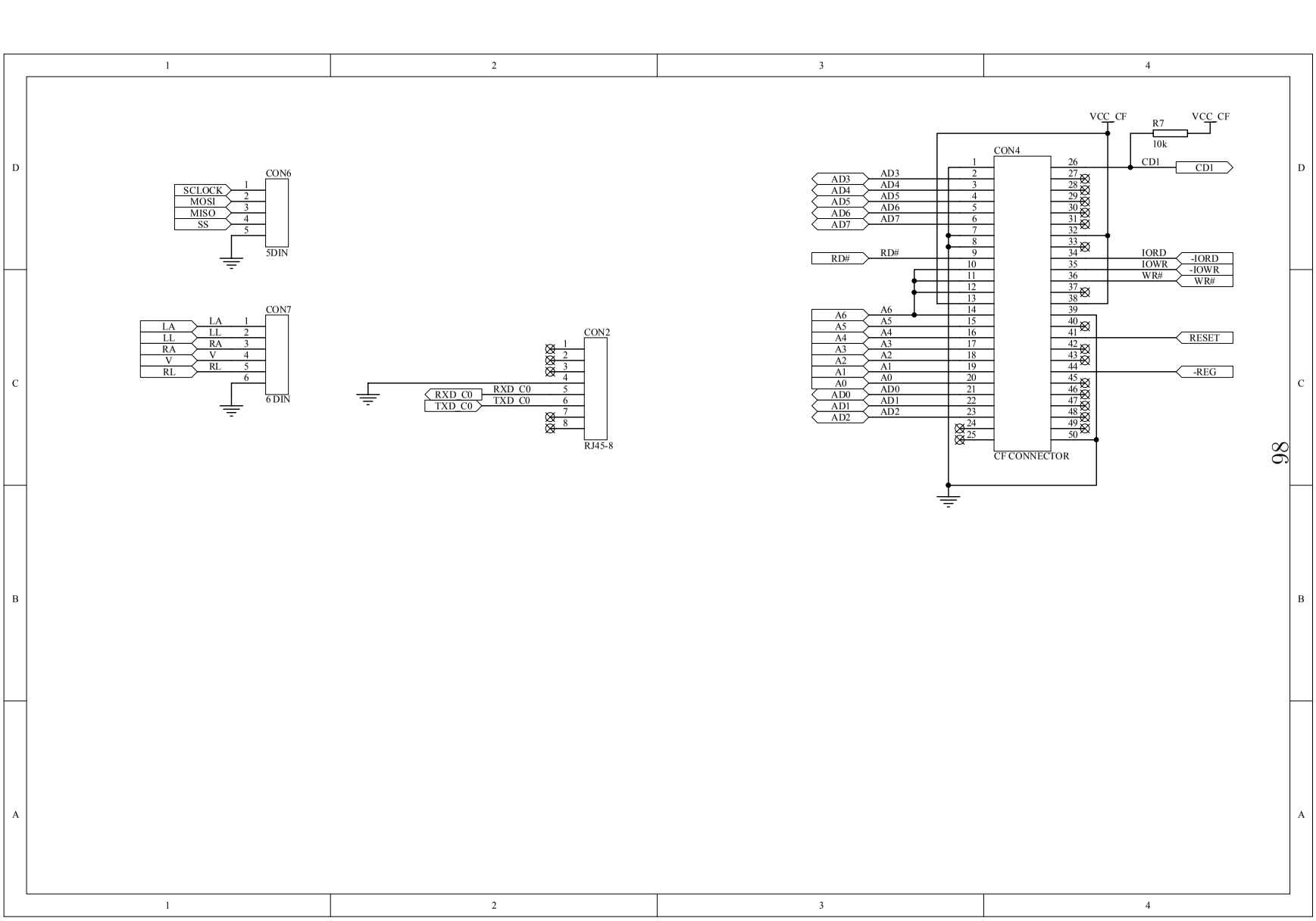
Apêndice C

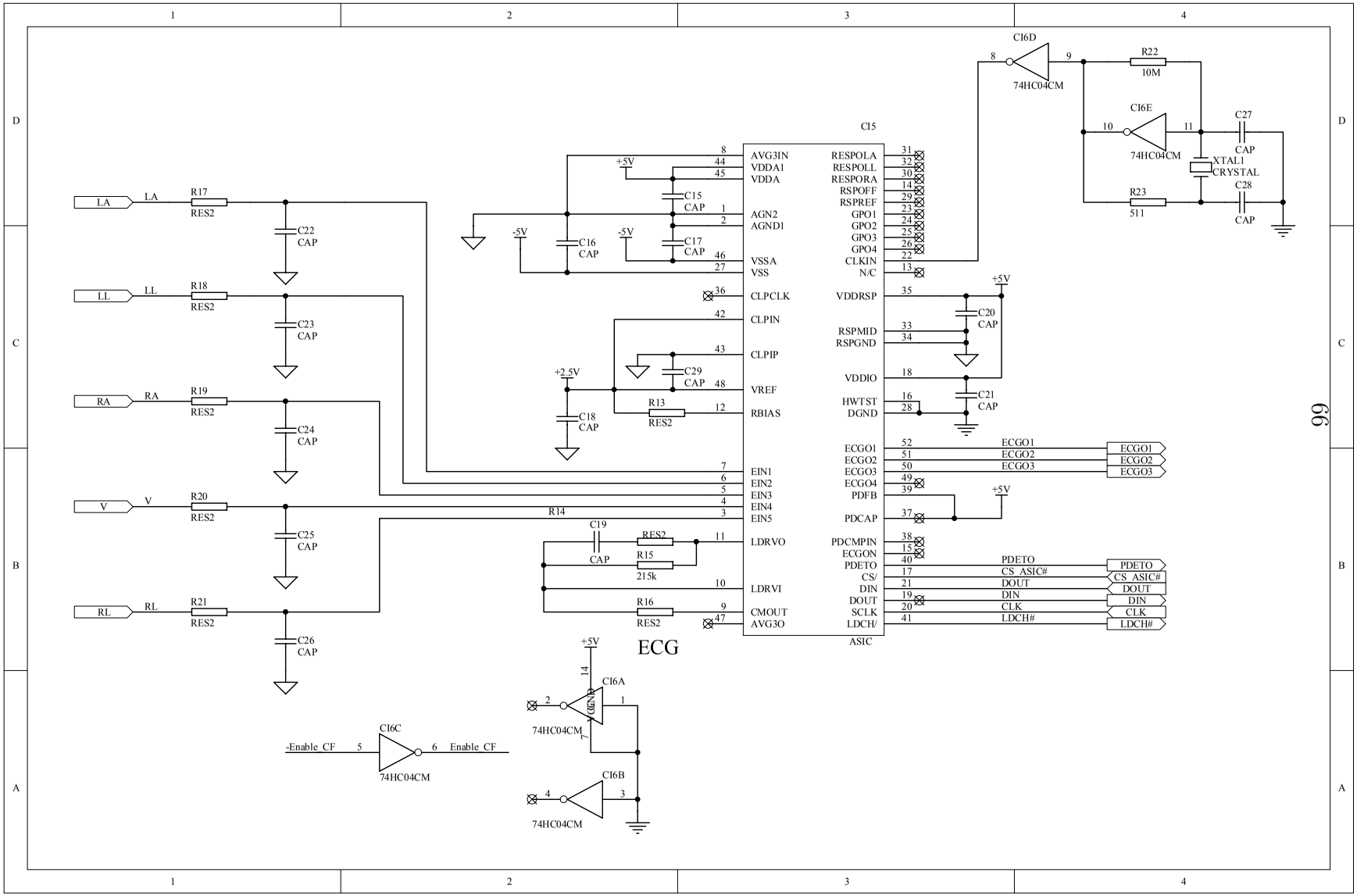
Circuito do Monitor Cardíaco modificado para suportar Comunicação sem fio e circuito ECG

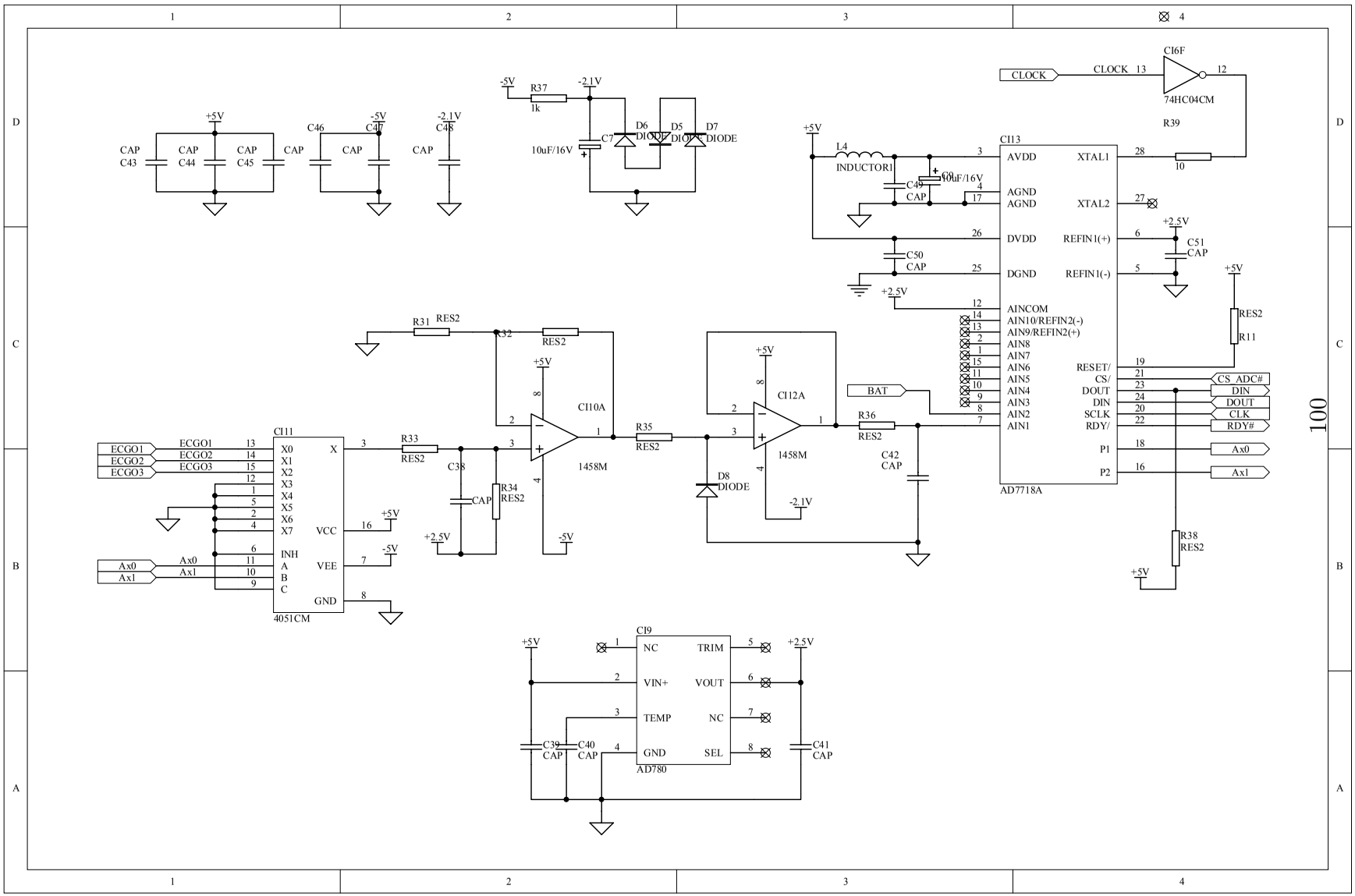
- Figura c-1 - Circuito da CPU
- Figura c-2 - Circuito do Conector
- Figura c-3 - Circuito do ASIC
- Figura c-4 - Circuito do ADC
- Figura c-5 - Circuito da Fonte

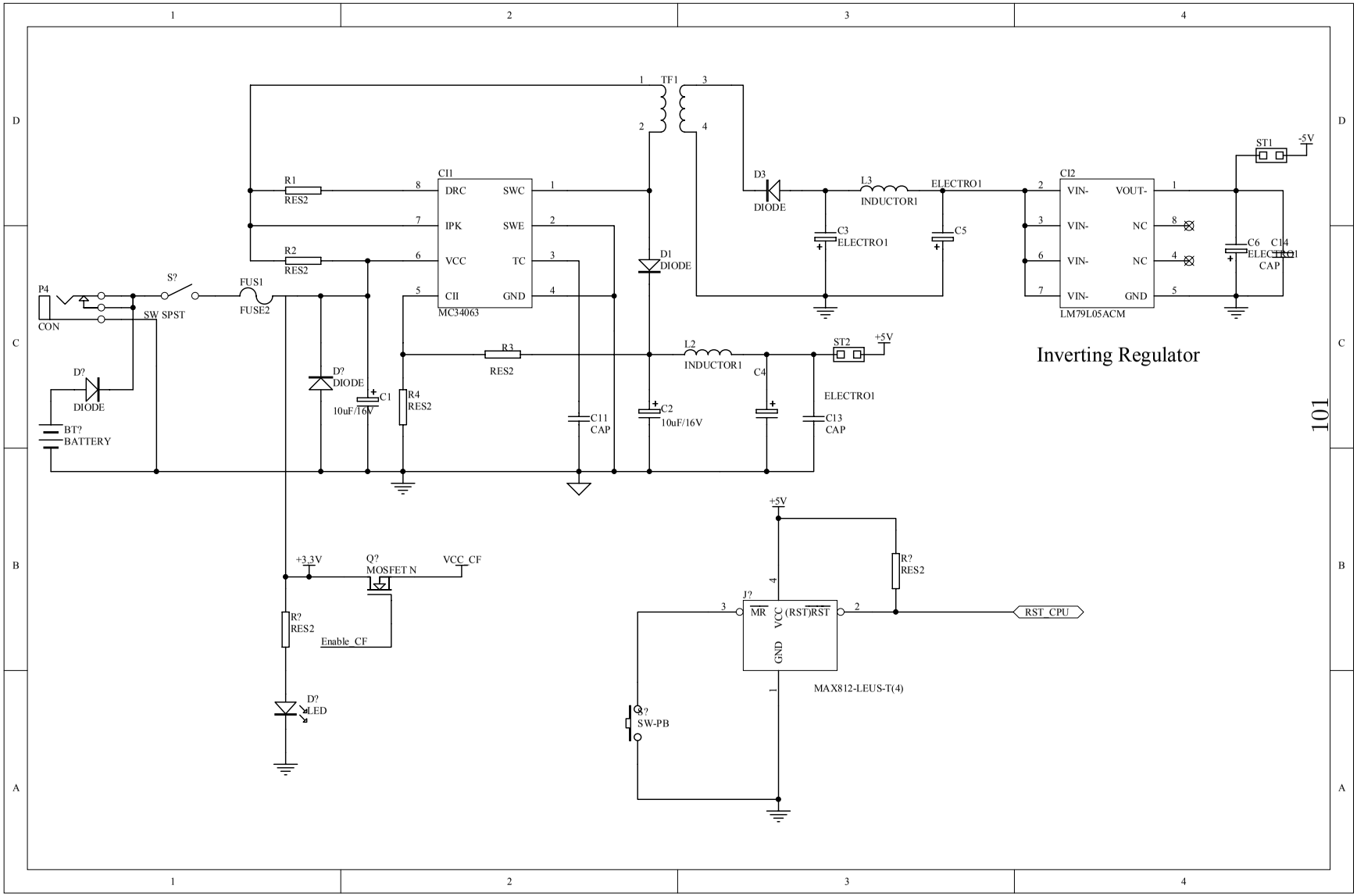

CLOCK
XTAL2
CRYSTAL
CI7
XTAL1
XTAL2
RST_CPU
CD1
RDY#
PDETO
DIN
LDCH#
+2.5V
DEC_A
DEC_B
SCLOCK
MOSI
MISO
SS
RESET
P1.4 (AIN1)
P1.5 (AIN2)
P1.6 (AIN3)
P1.7 (AIN4/DAC)
P1.3 (AIN5/IEXC2)
REFIN+
REFIN-
P1.0 (T2)
P1.1 (T2EX)
P1.2 (DAC/IEXC1)
SCLOCK/D0
MOSI/D1
MISO
SS
P0.0 (AD0)
P0.1 (AD1)
P0.2 (AD2)
P0.3 (AD3)
P0.4 (AD4)
P0.5 (AD5)
P0.6 (AD6)
P0.7 (AD7)
P2.0 (A8/A16)
P2.1 (A9/A17)
P2.2 (A10/A18)
P2.3 (A11/A19)
P2.4 (A12/A20)
P2.5 (A13/A21)
P2.6 (A14/A22)
P2.7 (A15/A23)
P3.2 (INT0)
P3.3 (INT1)
P3.4 (T0)
P3.5 (T1)
P3.6 (WR)
P3.7 (RD)
P3.1 (TXD)
P3.0 (RXD)
ALE
PSEN
EA
AVDD
AGND
DGND
DGND
DGND
DVDD
DVDD
DVDD
ADUC834
ST3
R27
1k
R26
1k
+5V
C30
CAP
C32
CAP
C33
CAP
C34
CAP
NET1
4x10K
NET2
4x10K
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A0
A1
A2
A3
A4
A5
A6
CS_ADC#
ECG_BTN
DEC_C
CS_ASIC#
REG
DOUT
CLK
TXD0
RXD0
R?
1K
S?
SW SPST
U?
DEC_A
DEC_B
DEC_C
VCC
G1
G2A
G2B
A
B
C
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
RD#
WR#
RESET
LED1
LED2
-IOWR
-IORD
-Enable_CF
DM74LS138
LED1
R6
RES2
D?
LED
LED2
INTERFACE SERIAL RS232
CI8
C1+
C1-
C2+
C2-
VCC
GND
VDD
VEE
T1
T2
R1
R2
C52
1uF
C53
1uF
C37
100nF
C55
1uF
C54
1uF
TXD_C0
RXD_C0
MAX232CSE

97


CON6
SCLOCK
MOSI
MISO
SS
5DIN
CON7
LA
LL
RA
V
RL
6 DIN
CON2
RXD_C0
TXD_C0
RJ45-8
CON4
VCC_CF
R7
10k
CD1
IORD
IOWR
WR#
RESET
-REG
RD#
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A0
A1
A2
A3
A4
A5
A6
CF CONNECTOR


ECG
CI5
ASIC
CI6A
CI6B
CI6C
CI6D
CI6E
74HC04CM
AVG3IN
VDDA1
VDDA
AGN2
AGND1
VSSA
VSS
CLPCLK
CLPIN
CLPIP
VREF
RBIAS
EIN1
EIN2
EIN3
EIN4
EIN5
LDRVO
LDRVI
CMOUT
AVG3O
RESPOLA
RESPOLL
RESPORA
RSPOFF
RSPREF
GPO1
GPO2
GPO3
GPO4
CLKIN
N/C
VDDRSP
RSPMID
RSPGND
VDDIO
HWTST
DGND
ECGO1
ECGO2
ECGO3
ECGO4
PDFB
PDCAP
PDCMPIN
ECGON
PDETO
CS/
DIN
DOUT
SCLK
LDCH/
LA
LL
RA
V
RL
R17
R18
R19
R20
R21
R22
R23
R13
R14
R15
R16
10M
511
215k
RES2
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
CAP
XTAL1
CRYSTAL
+5V
-5V
+2.5V
-Enable CF
Enable CF
CS_ASIC#
DOUT
DIN
CLK
LDCH#
99


CAP C43
CAP C44
CAP C45
CAP
C46
CAP
C47
CAP
C48
+5V
-5V
-2.1V
R37
1k
10uF/16V
C7
D6 DIODE
D5 DIODE
D7 DIODE
CLOCK
CLOCK 13
CI6F
74HC04CM
12
R39
10
CI13
AD7718A
AVDD
XTAL1
XTAL2
AGND
DVDD
DGND
REFIN1(+)
REFIN1(-)
AINCOM
AIN10/REFIN2(-)
AIN9/REFIN2(+)
AIN8
AIN7
AIN6
AIN5
AIN4
AIN3
AIN2
AIN1
RESET/
CS/
DOUT
DIN
SCLK
RDY/
P1
P2
L4
INDUCTOR1
C49
CAP
C50
CAP
C51
CAP
+2.5V
RES2
R11
CS_ADC#
DIN
DOUT
CLK
RDY#
Ax0
Ax1
R38
RES2
R31 RES2
R32
RES2
CI10A
CI12A
1458M
R35
RES2
R36
RES2
BAT
C42
CAP
D8
DIODE
CI11
4051CM
ECGO1
ECGO2
ECGO3
X0
X1
X2
X3
X4
X5
X6
X7
X
VCC
VEE
GND
INH
A
B
C
R33
RES2
C38
CAP
R34
RES2
CI9
AD780
NC
VIN+
TEMP
GND
TRIM
VOUT
NC
SEL
C39
CAP
C40
CAP
C41
CAP
100


TF1
CI1
DRC
SWC
IPK
SWE
VCC
TC
CII
GND
MC34063
R1
RES2
R2
RES2
D1
DIODE
D3
DIODE
L3
INDUCTOR1
ELECTRO1
C3
ELECTRO1
C5
CI2
VIN-
VOUT-
NC
LM79L05ACM
ST1
-5V
C6
C14
CAP
P4
CON
S?
SW SPST
FUS1
FUSE2
D?
DIODE
BT?
BATTERY
D?
DIODE
C1
10uF/16V
R4
RES2
R3
RES2
L2
INDUCTOR1
C4
ST2
+5V
ELECTRO1
C13
CAP
C11
CAP
C2
10uF/16V
Inverting Regulator
101
+3.3V
Q?
MOSFET N
VCC_CF
R?
RES2
Enable_CF
D?
LED
+5V
R?
RES2
J?
MR
VCC
(RST)RST
GND
RST_CPU
MAX812-LEUS-T(4)
Q?
SW-PB