ESCOLA POLITÉCNICA

Departamento de Sistemas e Computação

SISTEMA DE COMUNICAÇÃO *BLUETOOTH* UTILIZANDO MICROCONTROLADOR

Trabalho de Conclusão de Curso

Engenharia da Computação

Davidson Fellipe da Silva

Orientador: Prof. Sérgio Campello Oliveira

UNIVERSIDADE DE PERNAMBUCO

ESCOLA
POLITÉCNICA DE
PERNAMBUCO

DAVIDSON FELLIPE DA SILVA

SISTEMA DE COMUNICAÇÃO *BLUETOOTH* UTILIZANDO MICROCONTROLADOR

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

Recife, dezembro 2009.

Agradecimentos

Agradeço a Deus, porque sem ele absolutamente nada é possível.

Agradeço a minha família e a minha noiva Juliana Malta (que acabou aprendendo *Bluetooth* "por tabela") pelo grande apoio nessa caminhada.

Agradeço a Sérgio Campello, Leandro Honorato, Sidney Lima, Bruno Montenegro e Daniel Barlavento pela enorme contribuição a esse trabalho. Sem vocês este trabalho não teria sido concluído.

Agradeço os comentários de Ismael Mascarenhas e Diego Liberalquino sobre o trabalho.

Agradeço os meus companheiros de turma, por termos feito com que a turma 2005.1 fosse uma das que mais contribuíram para o crescimento do nosso curso. Valeu turma!

Agradeço aos professores que continuam no departamento e que estão lutando para reconstrução do nosso curso.

Agradeço aos colegas da webinterativa por me tolerar durante esse tempo de escrita da monografia.

Enfim também sou grato a todos que colaboraram positivamente, ou negativamente, com o possível final da minha jornada universitária em 5 anos.

Resumo

Engenheiros desenvolvem soluções cada vez mais robustas, para nos trazer maior simplicidade, praticidade e eficiência para o paradigma de comunicação sem fio e torná-la cada vez mais presente em nossas vidas. O *Bluetooth* foi nasceu diante da necessidade de uma solução para comunicação sem fio para curtas distâncias segura, de baixo custo, com suporte a comunicação por voz e por dados e com facilidade de integração aos protocolos de comunicação. A proposta deste trabalho é o desenvolvimento de um sistema de comunicação *Bluetooth* utilizando microcontrolador PIC. O objetivo é desenvolver um módulo para plataforma de desenvolvimento PIC e outro que será o módulo responsável para comunicação *Bluetooth*. Por fim fornecer uma biblioteca de funções, de código aberto, escrita em C, para facilitar o desenvolvimento de aplicações que desejem utilizar o protocolo de comunicação *Bluetooth* para microcontroladores da família PIC.

Abstract

Engineers develop solutions that are more robust, to bring us simplicity, practicality and efficiency to the paradigm of wireless communication and make it increasingly present in our lives. Faced with a solution for wireless communication for short distances to safe, affordable, supporting voice communications and data and ease of integration with communication protocols, there was Bluetooth. The proposed by this work is the development of a Bluetooth communication system using PIC microcontrollers. In order to develop a module for PIC development platform and another that is the module responsible for Bluetooth communication. Finally provide a library of functions, open source, written in C, to facilitate the development of applications wishing to use the Bluetooth communication protocol for PIC microcontroller family.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

Em ordem de alfabética:

- **ACL:** *Asynchronous Connection-Less*
- **API:** *Application Programming Interface*
- **CCS:** CUSTOM *Computer Service*
- **CISC:** *Complex Instruction Set Computer*
- **CTS:** *Clear to Send*
- **DSSS:** *Direct Sequence Spread Spectrum*
- **EDR:** *Enhanced Data Rate*
- **EEPROM:** *Electrically-Erasable Programmable Read-Only Memory*
- **FHSS:** *Frequency Hopping Spread Spectrum*
- **GFSK:** *Gaussian Frequency-Shift Keying*
- **HCI:** *Host Controller Interface*
- **I²C:** *Inter-Intergrated Circuit*
- **ICD:** *In-Circuit Debugger*
- **ISM:** *Industrial, Scientific and Medicine*
- **L2CAP:** *Logical Link Control and Adaptation Protocol*
- **LCD:** *Liquid Crystal Display*
- **LMP:** *Link Manager Protocol*
- **MCLR:** *Master Clear*

- **OFDM**: *Orthogonal Frequency-Division Multiplexing*
- **PANS:** *Personal Area Networks*
- **PCB:** *Printed Circuit Board*
- **PWM:** *Pulse-Width Modulation*
- **QoS:** *Quality of Service*
- **RAM:** *Random Access Memory*
- **RISC:** *Reduced Instruction Set Computer*
- **RTS:** *Request to Send*
- **SCO:** *Synchronous Connection-Oriented*
- **SIG:** *Special Interest Group*
- **ULA:** Unidade lógica e aritmética
- **WPAN:** *Wireless Personal Area Network*

Capítulo 1

Introdução

A tecnologia *Bluetooth* [1] nasceu no final década de 90, sendo que, os primeiros produtos só apareceram comercialmente após o ano 2000, em aplicativos de dispositivos móveis [2]. A tecnologia já vendeu cerca de 2 bilhões de produtos na primeira década de existência [1] e possui uma ótima aceitação no mercado WPAN (*Wireless Personal Area Network – Padrão de Rede de Área Pessoal Sem Fio*). Esse padrão designa redes com alcance suficiente para conectar os diversos dispositivos utilizados por uma pessoa padrão separados por até 10 metros [3]. O *Bluetooth* tem a finalidade de realizar a comunicação entre diversos dispositivos como computadores, telefones, *smartcards*, impressoras, entre outros, em uma determinada rede. Um dos grandes diferenciais da tecnologia *Bluetooth* é que ela permite a conectividade tanto de dispositivos fixos quanto móveis. Eliminando assim, um grave problema encontrado em redes tradicionais que é o cabo de conexão proprietário. Esse problema é caracterizado pelo fato que cada fabricante possui uma especificação, o que dificulta a interligação dos equipamentos diferentes [4]. Outra característica do *Bluetooth* é o baixo consumo de energia. Esse fator permite que uma gama de produtos seja fabricada com a tecnologia acoplada, sem comprometer a autonomia das baterias.

Além da conectividade de dispositivos usuais, a comunicação sem fio vem sendo cada vez mais utilizada em ambientes onde redes tradicionais apresentam sérias limitações, como indústrias fabricantes de líquidos corrosivos e inflamáveis, hidrelétricas e estações de linhas energéticas. Elevadas temperaturas, altas tensões elétricas, riscos de acidentes e locais de difícil acesso são alguns dos principais fatores para escolha da conectividade sem fio nesses ambientes. Em regra geral também há o uso, nesses locais, de microcontroladores tanto para o controle de processos lógicos quanto para o armazenamento de informações acerca do comportamento de determinado sistema [5].

O trabalho proposto tem como principal objetivo projetar e implementar um sistema de comunicação sem fio, através da tecnologia *Bluetooth,* utilizando o microcontrolador PIC [6]. O objetivo é facilitar o desenvolvimento de sistemas que possuam comunicação *Bluetooth* utilizando essa família de microcontroladores. Os objetivos específicos do trabalho proposto são:

1. Projetar um *hardware* que servirá como módulo para o desenvolvimento de aplicações que desejam utilizar *Bluetooth*.

2. Desenvolver uma API (*Application Programming Interface* – Interface de Programação de Aplicativos) com funções básicas que permitam uma facilidade maior para se trabalhar com transmissão de dados via *Bluetooth* utilizando o microcontrolador *PIC*.

Capítulo 2

Microcontroladores

Um microcontrolador é um dispositivo eletrônico que contém um microprocessador, memórias com funções de leitura e escrita, interfaces de entrada e saída bem como diversos periféricos úteis no desenvolvimento de sistemas embarcados como: temporizadores, comparadores, geradores de *clock*, conversores analógico/digital e também conversores digital/analógico. Em regra geral microcontroladores contém uma grande quantidade de periféricos internos reduzindo, assim, a necessidade de utilização de muitos componentes externos. Essa é uma de suas principais diferenças em relação aos microprocessadores que não costumam ter periféricos em uma única pastilha, embora tenham capacidade de processamento maior [6].

Microcontroladores são computadores de propósito específico. Eles possuem tamanho reduzido, baixo custo e baixo consumo de energia. Devido a esses fatores há diversos segmentos, que os utilizam, tais como a indústria automobilística, de telecomunicações, de brinquedos, de eletrodomésticos, de eletroeletrônicos, bélica, etc.

A programação de um microcontrolador pode ser feita utilizando linguagens como Assembly, C, Basic, Pascal, entre outras. Após o programa ser compilado e montado em linguagem de máquina ele é transferido ao microcontrolador utilizando algum gravador compatível com o modelo. No trabalho proposto foi utilizado o gravador MPLAB ICD3 *In-Circuit Debugger* [7]. Essa ferramenta também pode ser utilizada para simulação e depuração do programa no dispositivo a ser gravado.

2.1 Características principais

A Figura 1 mostra uma representação, em alto nível, da organização de um microcontrolador PIC. Essa figura apresenta módulos de memória, CPU, temporizadores, interfaces de entrada e saída.

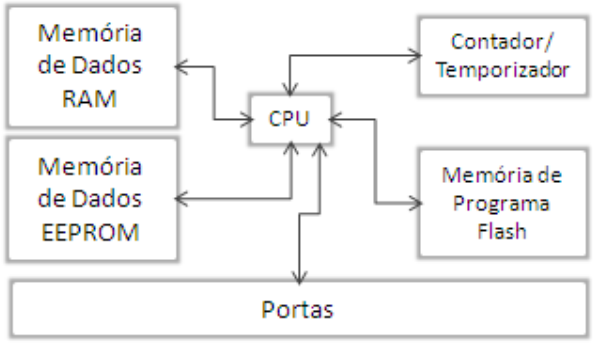


Figura 1. Representação da organização do microcontrolador PIC.

2.1.1 Memória

A memória é um dos componentes principais tanto na arquitetura Von Neumann quanto na Harvard, podemos dividi-la em duas categorias: memória de programa e memória de dados.

A memória de programa é onde são carregados os programas em execução e nela os programas podem gerenciar comunicação de entrada e saída, executar instruções, etc.

A memória de dados é utilizada para o armazenamento de resultados e dados que a CPU deve executar.

2.1.2 Interfaces de entrada e saída

As interfaces de entrada e saída são a forma de comunicação que o microcontrolador possui para comunicação com periféricos externos, e assim expandir suas funcionalidades, possibilitar iteração com usuário, e até controlar outros dispositivos eletrônicos. Essas interfaces podem utilizar transmissão serial ou paralela.

2.1.3 Temporizadores e Contadores

Os temporizadores são programados por software e podem operar independente dos demais sistemas do chip. O funcionamento deles é controlado por

registradores internos. São utilizados normalmente para retardos, geração de pulsos, entre outras rotinas onde há uma necessidade de um controle temporal ou de uma contagem.

2.1.4 ULA (Unidade Lógica e Aritmética)

A ULA está presente em todos os microprocessadores. Ela contém circuitos destinados a realizar funções de cálculo e manipulação de dados durante a execução de um programa. A ULA de um microcontrolador funciona de forma análoga a ULA de um microprocessador.

2.1.5 Interrupções

Interrupção é a forma na qual permite ao microcontrolador interceptar eventos externos e internos (exemplo: divisão por zero e overflow) ao programa em execução. Quando uma interrupção é ativada ela interrompe o fluxo atual do programa em execução e uma sub-rotina pode ser executada. Após a execução dessa sub-rotina específica, e caso seja possível, retorna-se para execução normal do programa.

2.2 Arquitetura de um microcontrolador

2.2.1 Arquitetura Harvard

O microcontrolador PIC utiliza arquitetura Harvard, enquanto boa parte dos demais microcontroladores apresenta a arquitetura Von-Neumann [6]. A arquitetura Harvard se difere da Von Neumann por possuir duas memórias diferentes e separadas: uma de dados e uma de programa, que são conectadas por barramentos distintos, sendo um de dados e outro de instruções. A arquitetura Harvard, logo, permite o uso de tamanhos diferentes de barramento. Em microcontroladores PIC, o barramento de dados é de 8 *bits*, enquanto o de instruções pode ser 12, 14 e 16 *bits*.

2.2.2 Arquitetura Von Neumann

A arquitetura Von Neumann é caracterizada pelo conceito de programa armazenado na memória, que consiste em armazenar seus programas no mesmo espaço da memória de dados. A máquina proposta pela arquitetura de Von Neumann deve possuir os seguintes componentes: memória, ULA, processador, unidade de controle e diversos registradores [8]. As diferenças entre as arquiteturas utilizadas em microcontroladores podem ser visualizadas na Figura 2.

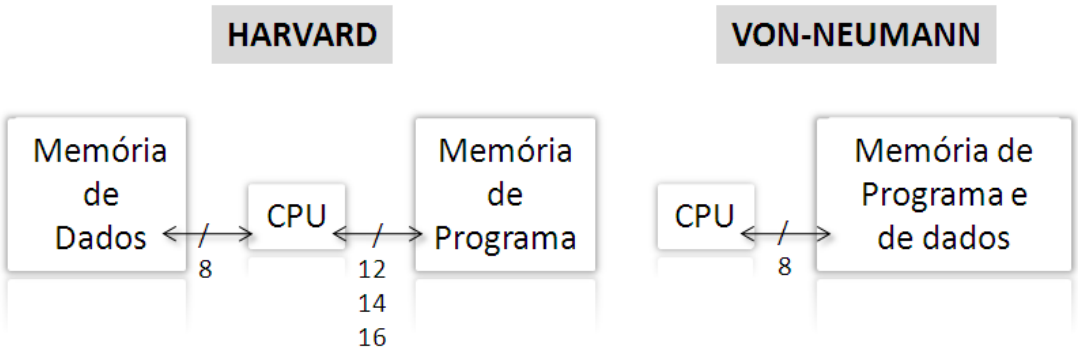


Figura 2. Arquitetura *Harvard* x Arquitetura Von-Neumann

2.2.3 Tecnologia RISC

Os PICs utilizam a tecnologia RISC (*Reduced Instruction Set Computer* – Computador com conjunto Reduzido de Instruções) e possui um conjunto reduzido de instruções, cerca de 35 no total. Isso faz com que o programador deva ter mais habilidade para elaborar suas próprias funções básicas [8].

2.2.4 Tecnologia CISC

Na maioria dos casos microcontroladores baseados em arquitetura Von Neumann utilizam tecnologia CISC (*Complex Instruction Set Computer* – Computador com Conjunto Complexo de Instruções) [6]. Ao contrário dos microcontroladores de tecnologia RISC, eles podem possuir um conjunto de instruções vasto, em média mais de cem instruções. Isso implica que muitas instruções já estão implementadas por padrão, facilitando, assim, a tarefa do programador [8].

Capítulo 3

Bluetooth

Bluetooth é uma especificação industrial para a comunicação sem fio de curto alcance. Ela visa substituir o cabeamento necessário para comunicação entre dispositivos eletrônicos, manter níveis elevados de segurança, ter baixo consumo de energia e baixo custo [1], Além disso, fornece tanto serviços síncronos, como transmissão de voz, quanto assíncronos, como transferências de arquivos.

O desenvolvimento da especificação *Bluetooth* começou em 1994 com a *Ericsson Mobile Communications* que tinha como objetivo pesquisar melhorias na comunicação entre aparelhos celulares e acessórios visando eliminar o uso de fios. Essa pesquisa resultou na especificação do sistema rádio de curto alcance *MCLink* que tinha as vantagens de baixo consumo e baixo custo. Depois do lançamento, empresas como *Intel*, *IBM*, *Toshiba* e *Nokia* se juntaram à *Ericsson* e formaram o consórcio *Bluetooth* SIG (*Special Interest Group* – Grupo de Interesse Especial) em 1998. O SIG é o consórcio responsável por publicar as especificações, administrar o programa de qualificação de empresas e também proteger os direitos da marca *Bluetooth*. Atualmente o SIG possui mais de 12 mil membros [1].

A tecnologia *Bluetooth* opera sobre uma banda de radiofreqüência denominada ISM (*Industrial, Scientific and Medicine* – Industrial, Cientifica e Médica), ela utiliza a técnica de transmissão FHSS (*Frequency Hopping Spread Spectrum* – Espalhamento Espectral por Saltos em Freqüências) [9]. O consórcio *Bluetooth* SIG até já lançou campanhas para alcançar a harmonização total entre os países com relação à banda de freqüência [10]. A harmonização da faixa de freqüência tem o objetivo de reduzir a ocorrência de interferências entre dispositivo *Bluetooth* e outros dispositivos [11].

Com essas faixas de alcance é possível a formação de pequenas redes privadas conhecidas como PANS (*Personal Area Networks* – Redes de Área Pessoal) ou *piconets*. Uma *piconet* é uma rede formada por até oito dispositivos, na

qual um dispositivo deve assumir a função de mestre e os demais de escravos. Um mesmo dispositivo pode atuar como escravo em mais de uma *piconet*, sendo até mesmo mestre em outra *piconet*, dizendo-se assim que ele faz parte de uma *scatternet*. A única restrição é que um dispositivo pode ser mestre de apenas uma rede *piconet* ao mesmo tempo.

A Tabela 1 apresenta uma comparação entre as principais características das especificações de padrões que permitem a comunicação sem fio entre dispositivos. Atualmente estão entre os mais utilizados o *Bluetooth*, o *Wi-Fi* e o *ZigBee*. *Wi-Fi* é um padrão que usa a técnica de transmissão DSSS (*Direct Sequence Spread Spectrum* – Espalhamento Espectral por Seqüência Direta) [12] e OFDM (*Orthogonal Frequency-Division Multiplexing* – Multiplexação Ortogonal por Divisão de Freqüência) [13] e usa o protocolo IEEE 802.11 [14]. *ZigBee* é focado em baixa potência de operação e baixa taxa de transmissão de dados, também faz uso da técnica de transmissão DSSS e utiliza o protocolo IEEE 802.15.4 [15].

Tabela 1. Principais características das especificações: *Bluetooth*, *Wi-Fi* e *ZigBee.*

Especificação	Máximo de nós	Alcance	Corrente típica	Técnica de transmissão
Bluetooth	Até 8	100m	65 a 170 mA	FHSS
Wi-Fi	Mais de 100	100m	350 mA	DSSS e OFDM
ZigBee	Até 65535	100m	30 mA	DSSS

3.1 FHSS

A técnica de transmissão FHSS (*Frequency Hopping Spread Spectrum* – Espalhamento Espectral por Saltos em Freqüências) é uma das variantes do *Spread Spectrum* (técnica que habilita a coexistência de múltiplas redes em uma mesma área). A FHSS converte a representação binária dos dados em sinais de rádio

adequados para transmissão e executa operações através de técnicas de chaveamento de freqüência e modulação de sinal [16]. Ele transmite alguns bits numa determinada freqüência e depois pula para outra freqüência, transmitindo mais alguns bits, e assim por diante. Cada pacote no modelo FHSS é transmitido em uma freqüência diferente, sendo que essa freqüência varia de forma conhecida, porém pseudo-aleatória, tendo como referência uma seqüência que somente é conhecida pelo transmissor e também pelo receptor, o que dificulta possíveis interceptadores do sinal. Pode-se definir pseudo-aleatoriedade como um conjunto finito de números que apresentam características de números aleatórios para qualquer um que não conheça a fórmula geradora dos números [17].

O FHSS pode transmitir dados no formato binário, tanto a 1 Mbps (Megabit por segundo) quanto a 2 Mbps. Para cada uma dessas velocidades se utiliza uma modulação diferente, GFSK (*Gaussian Frequency-Shift Keying* – Modulação Gaussiana por Chaveamento de Freqüência) [18] de dois níveis para a velocidade de 1 Mbps e GFSK de quatro níveis para a velocidade de 2 Mbps.

3.2 Arquitetura *Bluetooth*

A pilha de protocolos *Bluetooth* é apresentada na Figura 3. A figura mostra os protocolos essenciais para o funcionamento do protocolo e é dividida em três grupos lógicos:

1. grupo de protocolo de transporte;
2. grupo de protocolo de *middleware;*
3. grupo de protocolo de aplicação.

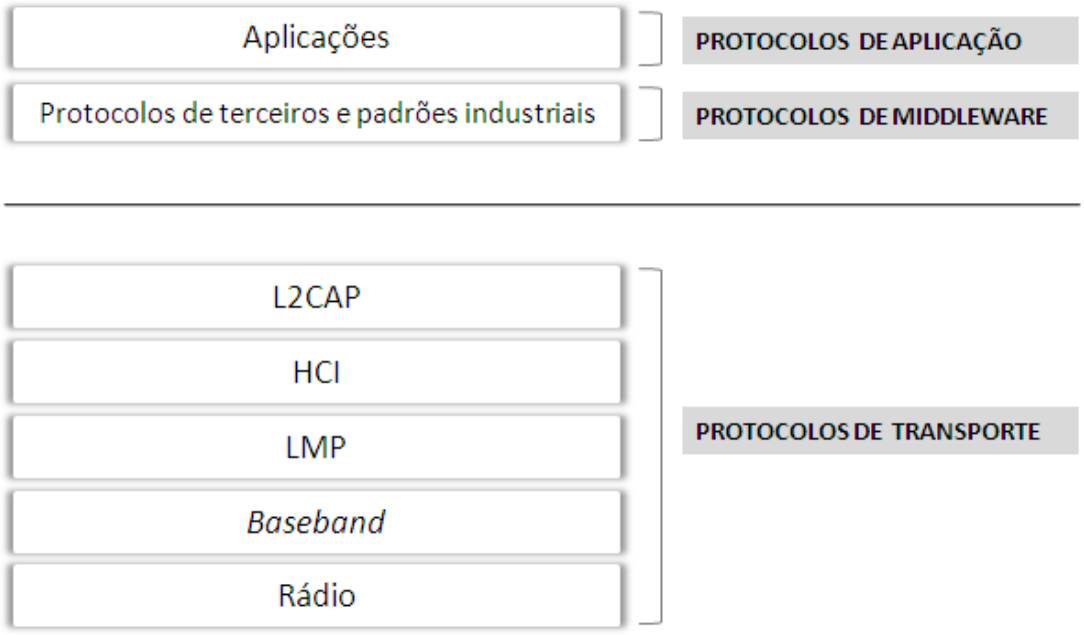


Figura 3. Pilha de protocolos *Bluetooth* [19].

O grupo de protocolos de transporte tem a finalidade de fornecer ao *Bluetooth* a capacidade de encontrar os dispositivos e coordenar *links* lógicos e físicos a camadas superiores. Já o grupo de protocolos de *middleware* permite que aplicações operem sobre *links Bluetooth.* E o grupo de aplicação são os aplicativos propriamente ditos, que oferecem algum tipo de funcionalidade relacionada ao uso do *Bluetooth.*

3.2.1 Rádio Bluetooth

O Rádio Bluetooth é a camada responsável por especificar detalhes, tais como freqüência de operação, potência e as técnicas de modulação e de transmissão utilizadas. Na maioria dos países essa banda de freqüência varia de 2400 MHz a 2483.5 MHz, sendo essa dividida em 79 canais ordenados e que medem 1 MHz, começando em 2402 MHz [1].

Para compensar regularizações em alguns países, existe uma banda de guarda superior de 2 MHz e uma inferior de 3.5 MHz. Algumas das características relacionadas a potências utilizadas no transmissor do *Bluetooth* podem ser

visualizada na Tabela 2 [1]. A taxa de transmissão pode chegar a 1 Mbps, caso faça o uso do EDR (*Enhanced Data Rate* – Transferência de Dados Avançada) pode alcançar 2 ou 3 Mbps [19][20][21].

Tabela 2. Características das potências utilizadas pelo *Bluetooth*

Classe	Alcance	Potência
Classe 1	Longo alcance (100 m)	Entre 1 mW e 100 Mw
Classe 2	Curto alcance (10 m)	Entre 0,25 mW e 2,5 mW
Classe 3	Curtíssimo (1 m)	Até 1 mW

3.2.2 *Baseband*

A *baseband* é a camada física do *Bluetooth*. Ela especifica as regras de acesso ao meio e procedimentos de camada física entre dispositivos distintos [1]. Entre essas regras que o *baseband* especifica estão: localização de dispositivos na rede, estabelecimento de conexão entre dispositivos, tipo e formato do pacote, modo de endereçamento. temporização, criptografia e correção de erros, transmissão e retransmissão de pacotes, papéis que um determinado dispositivo pode assumir na rede.

3.2.3 LMP (*Link Manager Protocol* – Protocolo de Gerenciamento de Enlace)

O LMP é utilizado para gerenciar o processo de autenticação, estabelecimento de conexão e configuração entre dois dispositivos. Dentre as principais responsabilidades do LMP podemos incluir a função de monitoramento de QoS (*Quality of Service* – Qualidade de Serviço), segurança e serviços ligados a conexão.

O LMP trabalha em função de troca de mensagens para realização de transações. Essas mensagens são filtradas e depois interpretadas pelo gerenciador de conexão. Entre as principais trocas de mensagens realizadas LMP, pode-se destacar:

- **controle de conexão:** estabelecimento de conexão, controle de energia, QoS, parâmetros de sistema de paginação, EDR (*Enhanced Data Rate* – Transferência de Dados Avançada);
- **segurança:** autenticação, emparelhamento e criptografia;
- **informações de modo de operação:** Modo de página, solicitação, conectado, bloqueado, escuta, transmissão, estacionado e espera. (descritos na seção 3.4);
- **informações das requisições:** precisão da temporização, versão do LMP, recursos suportados e nome da requisição;
- **modo de teste:** ativação e desativação de teste e controle do modo de teste.

3.2.4 HCI (*Host Controller Interface* – Interface de Controle de Host)

O HCI fornece uma interface com regras para a *baseband* e o LMP, acesso ao *status* do *hardware* e controle de registradores. De um modo geral, essa interface provê um método uniforme para acessar funções da *baseband*. A HCI é divida em três partes:

1. módulo de *driver;*
2. módulo de *firmware;*
3. módulo de transporte entre o *driver* do HCI e o *firmware* do HCI.

Módulo *driver* do HCI, está alocado na parte do *software* do *Bluetooth*. Ele tem a responsabilidade de receber notificações assíncronas de que algo ocorreu. Quando a notificação é recebida, ele irá fazer uma avaliação do pacote recebido e determinar qual evento ocorreu.

Módulo de *firmware* está alocado na parte do *hardware* do *Bluetooth*. Ele implementa os comandos para o *hardware* acessar os comandos de *baseband*, LMP, registradores de *status*, registradores de controle e registradores de eventos.

Módulo de transporte está entre o *driver* do HCI e o *firmware* do HCI. Ele provê uma transferência de dados, sem conhecê-los. O *Host* deve receber notificações de eventos HCI independente de qual módulo de transporte é utilizado.

3.2.5 L2CAP (*Logical Link Control and Adaptation Protocol* - Protocolo de Adaptação e Controle do Enlace Lógico)

O L2CAP está acima do protocolo *baseband* e pode ser considerado como o protocolo da camada de enlace, fornecendo serviços a camadas superiores e ocultando detalhes das camadas inferiores. Essa camada possui a responsabilidade de segmentação, remontagem, QoS, abstração de grupos, multiplexação e de-multiplexação dos dados trafegados.

Há dois tipos de links suportados pela camada *baseband*: SCO (*Synchronous Connection-Oriented* – Síncronos Orientados a Conexão) e ACL (*Asynchronous Connection-Less* - Assíncronos Não-orientados a Conexão). O SCO suporta tráfego de voz em tempo real e o ACL fornece o tráfego de melhor esforço.

3.3 Topologia do *Bluetooth*

Uma comunicação *Bluetooth* pode ser entendida a partir do conceito de *piconet*, que consiste em dois ou mais dispositivos ocupando um mesmo canal físico, ou seja, eles estão sincronizados em um único *clock* e FHSS. O *clock* padrão da *piconet* corresponde ao *clock* de um dos dispositivos pertencentes a ela, denominado mestre. A FHSS é derivada do endereço do dispositivo mestre e seu *clock*. Os demais dispositivos estão sincronizados sob o *clock* do dispositivo mestre e são conhecidos como escravos. Podem existir *piconets* independentes umas das outras e em qualquer número. Cada uma, no entanto, deve possuir um canal físico diferente. Isso quer dizer, cada *piconet* deve ter um dispositivo mestre diferente com seu próprio *clock* e FHSS. Logo, é impossível que duas ou mais piconets possuam o mesmo mestre. Um escravo, no entanto, pode ter vários mestres, ou seja, pertencer a várias piconets. Um dispositivo *Bluetooth* pode participar simultaneamente de duas ou mais *piconets*. Isso é possível graças a multiplexação por divisão de tempo [1][2][3].

Um dispositivo *Bluetooth* que é membro de duas ou mais *piconets*, pode-se dizer que ele está envolvido em uma *scatternet*. Os protocolos do núcleo *Bluetooth* não se destinam a oferecer roteamento de rede, essa funcionalidade é de responsabilidade dos protocolos a nível de aplicação e está fora do escopo da especificação do núcleo *Bluetooth*. Exemplos de operações em *piconets* e *scatternets* podem ser visualizados na Figura 4.

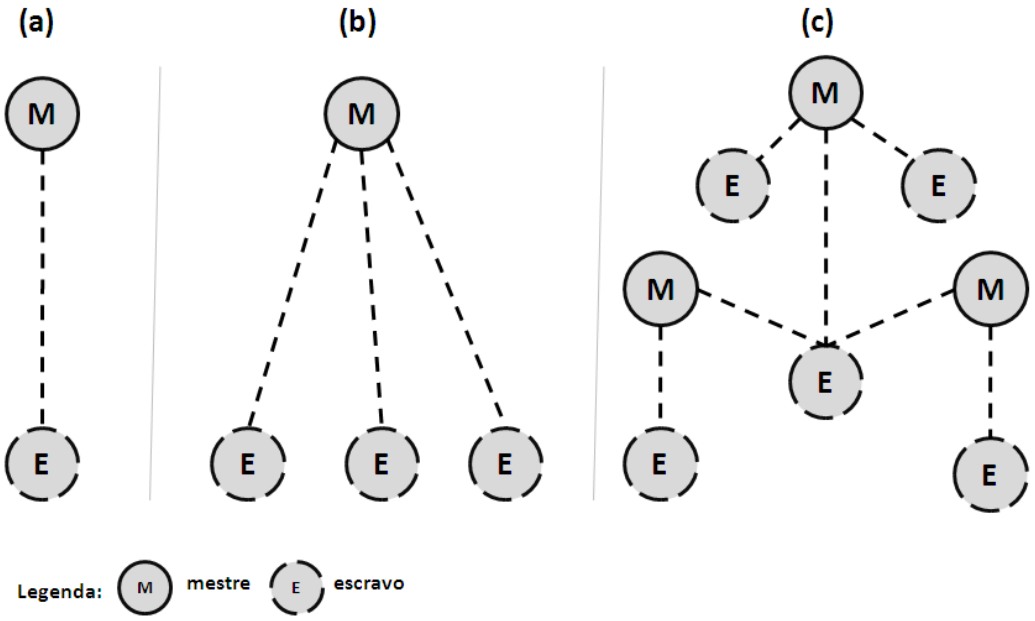


Figura 4. (a) simples operação mestre-escravo, (b) operação mestre com múltiplos escravos, (c) operação em uma *scatternet*

Observando a Figura 4 (a) é possível notar a representação da simples comunicação de um mestre com um escravo, a Figura 4 (b) demonstra um mestre se comunicando com 3 escravos e na Figura 4 (c) há a comunicação em *piconets* distintas.

3.4 Processo de estabelecimento de conexões

Um dispositivo que utiliza a tecnologia *Bluetooth* pode se encontrar em algum dos estados: página, solicitação, conectado, bloqueado, escuta, transmissão, estacionado e espera. A Figura 5 mostra como esses estados interagem entre si.

- **Estado de Espera:** Pode-se dizer que um dispositivo se encontra no estado de espera**,** caso ele esteja ligado e ainda não se conectou a uma *piconet.*

- **Estado de Solicitação:** Quando um dispositivo envia requisições para encontrar outros que ele pode se conectar.

- **Estado de Página:** Esse é um estado que exclusivamente **mestres** podem se encontrar, nele há um envio de mensagens em busca de encontrar outros dispositivos que possam entrar na sua *piconet*.

- **Estado Conectado:** O dispositivo entra nesse estado após o momento que há um estabelecimento de conexão e um dos pontos assume o papel de mestre e o outro de escravo.

- **Estado de Transmissão:** Estado em que o dispositivo envia dados a outro. Após o termino do envio ele volta para o estado conectado.

- **Estado de Escuta:** Nesse estado o escravo fica inativo por uma quantidade definida de *slots.*

- **Estado bloqueado:** Nesse estado o escravo também fica inativo por uma quantidade definida de *slots.* Mas, não ocorre transferência de dados nesse estado

- **Estado estacionado:** Nesse estado o dispositivo perde o seu endereço atual na *piconet.* Esse endereço retirado é dado a outro escravo que o mestre está tirando do estado de estacionado.

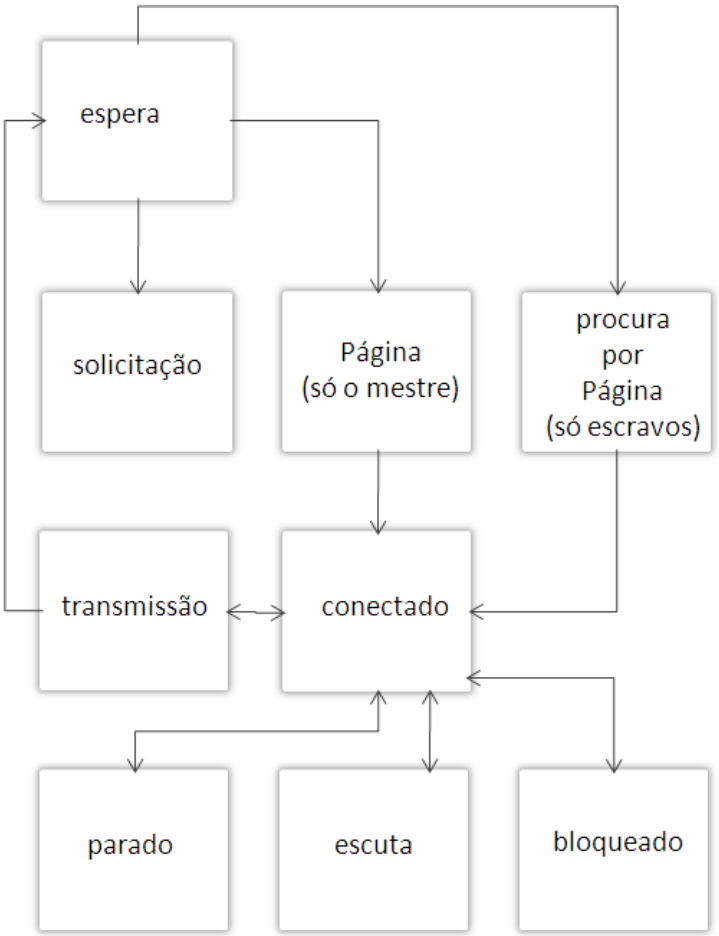


Figura 5. Máquina de estados de um dispositivo *Bluetooth.*

Capítulo 4

Projeto e desenvolvimento do Sistema de comunicação *Bluetooth* usado microcontrolador PIC

Esse capítulo descreve a proposta desse trabalho: um Sistema de Comunicação *Bluetooth* usando microcontrolador PIC. Ele apresenta os principais componentes utilizados, a descrição da API proposta e os detalhes sobre a implementação e o funcionamento do sistema.

4.1 Ferramentas Utilizadas

Essa seção descreve algumas das ferramentas utilizadas na confecção do *hardware* e do *software* do trabalho proposto.

4.1.1 Linguagem C para Microcontrolador

A linguagem C é bastante difundida na programação de sistemas embarcados. Os programas em C tendem a ser bastante compactos, possuírem bom tempo de execução e são mais fáceis de entender que os implementados em Assembly [22]. Outro aspecto que se pode destacar na linguagem C é que ele possui apenas 32 palavras reservadas [23]. Outras linguagens de alto nível têm uma quantidade bem maior, como BASIC o qual, em regra geral, possui mais de 100 palavras reservadas [24][25].

A linguagem C provê suporte a vários tipos de microcontroladores, tais como PIC, AVR, ARM e 80x51. Essa característica faz com que um programa escrito em C seja portável. Caso haja a necessidade da migração de plataforma se faz necessário pequenas adaptações no programa original. Essas mudanças não dizem respeito à

estrutura semântica da aplicação e sim as especificações do microcontrolador como dispositivos e *drivers*.

O programa proposto contém uma função principal que é o ponto de partida para a execução de todas as tarefas designadas. Ele também está modularizado em múltiplas funções as quais são invocadas do programa principal e de outras funções. A Figura 6 ilustra a hierarquia de funções na linguagem C. A função principal invoca as funções auxiliares, essas por sua vez invocam as sub-funções que podem ou não invocar sub-rotinas.

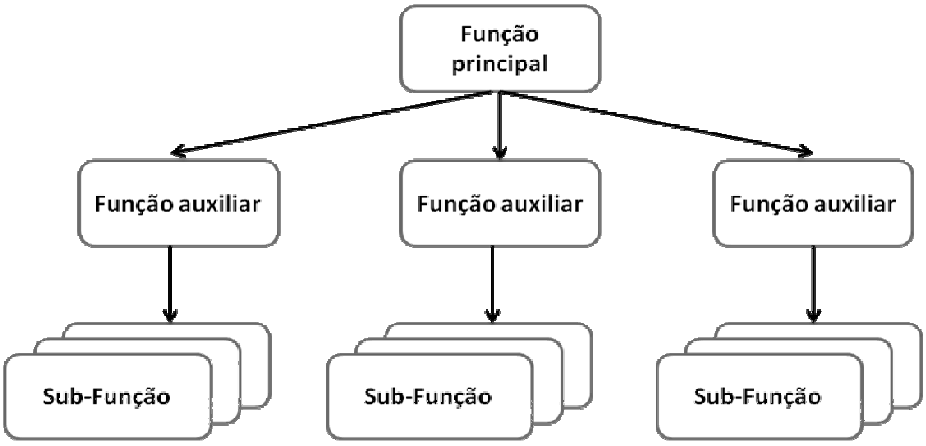


Figura 6. Hierarquia de funções na linguagem C.

As funções contêm declaração de dados, definições, comandos e expressões. O programa possui três arquivos diferentes juntamente com seus respectivos cabeçalhos. Cada arquivo contém um conjunto de funções que estão agrupados que acordo com as suas tarefas e co-relações. As diretivas de pré-processamento estão expostas a seguir:

- *#include*: incluem as funções do dispositivo alvo;
- *#fuses*: especificam os fusíveis para *o chip;*
- *#use delay*: determina a velocidade do *clock*;

O compilador, programa responsável pela tradução de uma linguagem para outra que geralmente converte o código de uma linguagem-fonte para uma linguagem-objeto, provê uma grande quantidade de bibliotecas, com funções pré-

definidas [26]. Essas funções podem acessar vários periféricos especificados pelos microcontrolador PIC.

O CCS (*CUSTOM Computer Service* – Sigla utilizada pela empresa fabricante do compilador) C *Compiler* [27] foi o compilador utilizado no trabalho proposto para o desenvolvimento da biblioteca de funções, essa ferramenta possui uma interface bastante amigável, apresentada na Figura 7, vale destacar ainda que, ele dá suporte a vários *drivers* os quais podem ser incluídos e usados no programa. Esse compilador ainda apresenta comandos para:

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

Além disso, o compilador fornece funções para conversação de valores analógicos, para comunicação via RS232, para comunicação com display de LCD (*Liquid Crystal Display* – Display de Cristal Líquido), além é claro de funções da linguagem C no padrão ANSI.

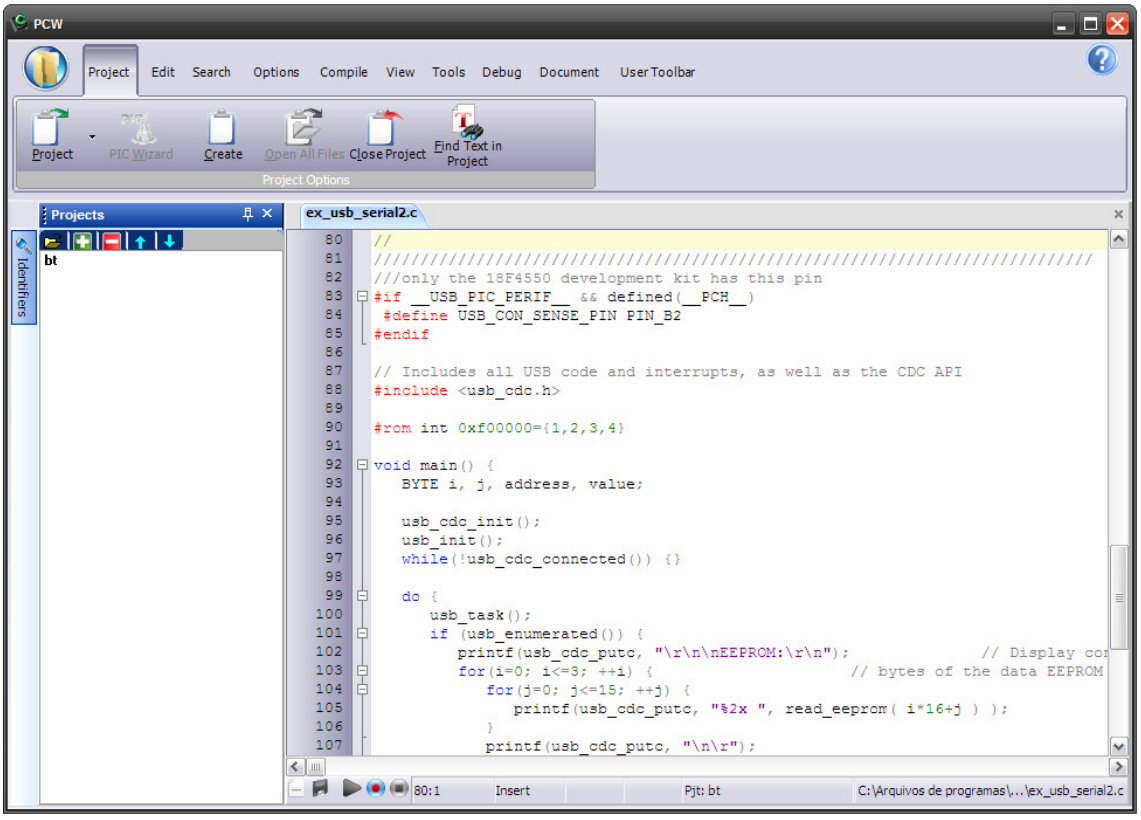


Figura 7. Interface do compilador CCS

O CCS oferece um bom suporte ao desenvolvimento de aplicativos para o PIC. Ele fornece bibliotecas para grande parte das famílias de microcontroladores da fabricante Microchip [28], incluindo o PIC16F877A, que foi utilizado nesse projeto. As famílias do microcontrolador PIC e os produtos da CCS estão relacionados na Figura 8.

Família	PIC10	PIC12/PIC16	PIC18	PIC24
Tamanho do OP-Code	12-bit	14-bit	16-bit	24-bit
Linha de comando integrados ao MPLAB (Windows e Linux)	PCB	PCM	PCH	PCD
IDE para Windows	PCW			
	PCWH			
	PCWHD			

Figura 8. Famílias PIC e respectivos produtos da CCS relacionados.

Quando comparado a outros compiladores C tradicionais, o CCS tem algumas limitações, tais como: funções recursivas não são permitidas, além de não permitir ponteiros para *arrays* constantes, devido ao fato de haver dúvidas na separação do segmento CODE/DATA no interior do PIC [29].

4.2 PROTEUS

O software de desenho e simulação PROTEUS [30] é uma ferramenta útil para estudantes e profissionais que desenvolvem aplicações analógicas e digitais. Ele permite o desenho de circuitos empregando um entorno gráfico no qual é possível colocar os símbolos representativos dos componentes e realizar a simulação de seu funcionamento sem o risco de ocasionar danos físicos aos circuitos. A simulação pode incluir instrumentos de medição e a inclusão de gráficos que representam os sinais obtidos na simulação.

O PROTEUS oferece a capacidade de simular adequadamente o funcionamento dos microcontroladores mais populares (PICS, ATMEL-AVR, Motorola, 8051, etc.). Além disso, ele simula circuitos digitais e analógicos simultaneamente. O PROTEUS fornece ainda equipamentos, de forma virtual,

comuns de bancadas de desenvolvimento de eletrônicos, como osciloscópios, multímetros, geradores de sinais, entre outros [31].

No trabalho proposto foi utilizado o PROTEUS. Ele foi empregado na etapa de desenvolvimento da Plataforma de Desenvolvimento PIC, descrita na subseção 4.5.1. Nessa etapa foi realizada a verificação do projetado para placa, e realização de testes do idealizado.

4.2.1 MPLAB IDE

A ferramenta MPLAB IDE fornece um ambiente integrado para o desenvolvimento de código para aplicações embarcadas e contém componentes necessários para o projeto e implementação de sistemas. Ele permite que o projetista implemente o código-fonte, realize o processo de compilação, e também teste o microcontrolador escolhido pelo projetista.

4.2.2 MPLAB ICD3

O MPLAB ICD3 é uma ferramenta comercializada pela Microchip, e ele possui as funções de gravador e depurador. O MPLAB ICD 3 (apresentado na Figura 9) é um circuito depurador que é controlado via o software MPLAB IDE, que é utilizado no desenvolvimento de *software* e *hardware* em microcontroladores da Microchip. Com ele é possível depurar uma aplicação, inserir *breakpoints* tanto em *hardware* quanto em *software*, entre outras funcionalidades. Para o projeto proposto as funcionalidades relacionadas à depuração do MPLAB ICD3 não foram utilizadas.

Figura 9. A face do MPLAB ICD3

4.3 Principais dispositivos utilizados

4.3.1 Microcontrolador PIC 16F877A

O PIC 16F877A [28] da fabricante *Microchip Technology* é o microcontrolador utilizado para implementação do projeto. Ele pertence à família de microcontroladores de 8 bits e possui uma arquitetura RISC. As suas principais características são:

- possui somente 35 instruções;
- opera nas velocidades: 20 MHz de *clock* e 200 ns de ciclo de instrução;
- possui modos de endereçamento direto, indireto e relativo;
- 3 temporizadores;
- diversas formas de comunicação serial: SPI, I²C e USART;
- memória de programa de 8k e RAM de 368 bytes;
- capacidade para interrupções.

Um dos motivos para escolha da família PIC é devido ao fato que todos os modelos possuem um conjunto de instruções bem similares e preservam muitas semelhanças entre suas características básicas. Sendo assim, utilizar o PIC 16F877A, torna sua portabilidade para outros modelos PIC uma tarefa bem simplificada [32]. A Figura 10 mostra a pinagem do microcontrolador PIC 16F877A. Ela apresenta em destaque as portas do dispositivo. Quando o dispositivo é habilitado, esses 33 pinos têm a funcionalidade de interface de entrada e saída para propósito geral.

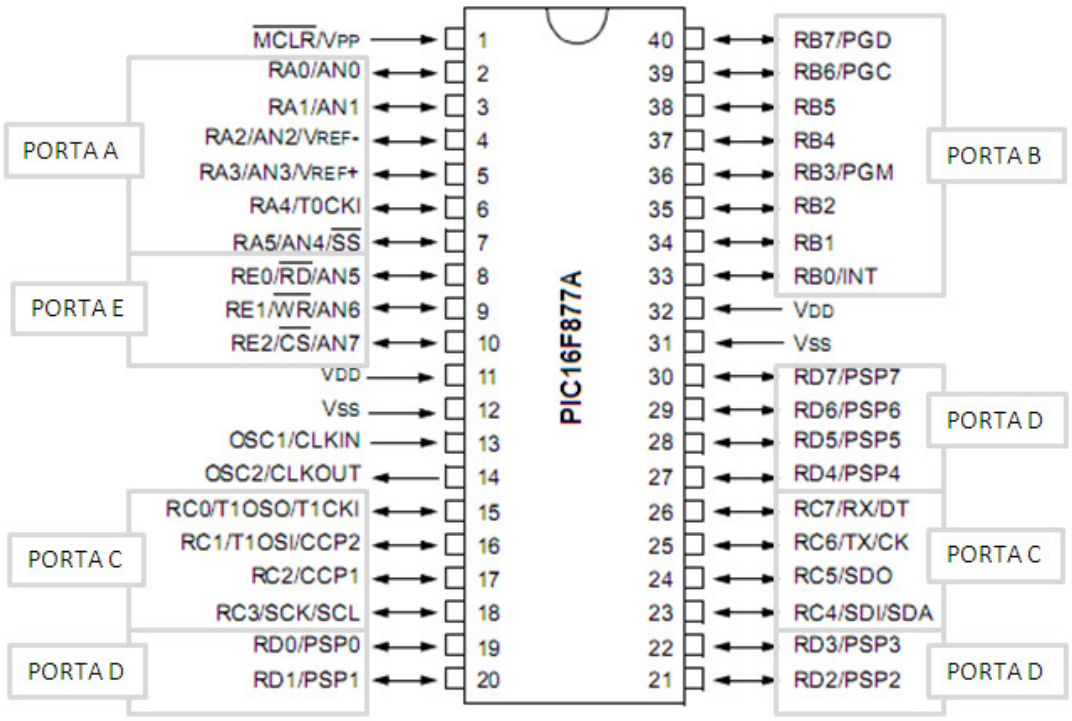


Figura 10. Pinagem do PIC16F87A (adaptada de [24])

4.3.2 Módulo Bluetooth KCWirefree KC-21

O módulo *Bluetooth KCWirefree* KC-21 é um *chip* para comunicação sem fio que utiliza a especificação *Bluetooth*. Ele possui uma antena interna, que conforme a Tabela 2 pode-se enquadrá-la como de classe 2. Sendo assim o KC-21 possui limite de alcance de 10 metros e oferece uma comunicação serial de velocidade máxima de 921 Kbaud (*baud* - representa o número de mudanças na linha de transmissão, seja em freqüência, ou em amplitude, ou em fase ou em eventos, por segundo).

Para o funcionamento desse módulo *Bluetooth* é necessário uma tensão de 3,3 volts. Ele possui 14 pinos de entrada e saída de propósito geral. Vale destacar ainda que o módulo possui uma memória flash de 8 Mbit. A Figura 11 mostra a pinagem no chip KC-21 e suas dimensões físicas em milímetros [32].

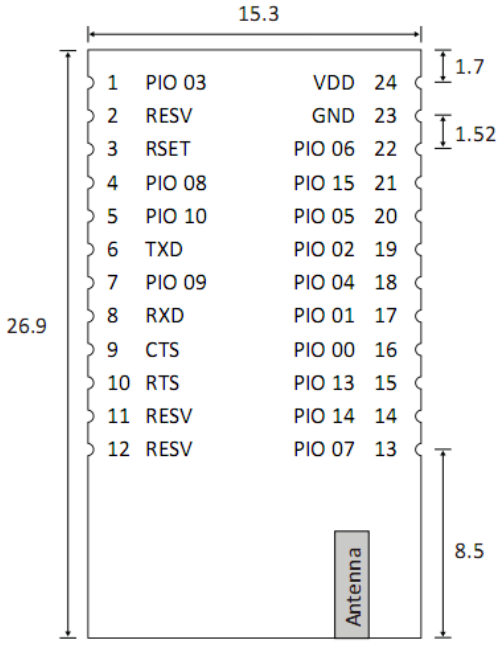


Figura 11. Pinagem e dimensões do *Bluetooth KCWirefree* KC-21 [32].

A Tabela 3 apresenta a descrição da pinagem do módulo *Bluetooth* utilizado no projeto proposto, que foi apresentado na Figura 11. O fabricante recomenda que pinos que não estão sendo utilizados não devem estar conectados.

Tabela 3. Descrição da funcionalidade dos pinos do *Bluetooth KCWirefree* KC-21.

Pinos	Legenda	Descrição
1, 4, 5, 7, 13, 14, 15, 16, 17, 18, 19, 20, 21 e 22	PIO	Pinos programáveis como de entrada e de saída de dados
2, 11 e 12	RESV	Reservados
3	RSET	Entrada de *reset*
6	TXD	Transmissão de dados
8	RXD	Recepção de dados
9	CTS	Utilizado para controle de fluxo – *Clear to Send*

10	RTS	Usado para controle de fluxo – *Request to Send*
23	GND	Terra
24	VDD	Alimentação

4.4 Ciclo de desenvolvimento do projeto

A metodologia proposta é subdividida em 7 etapas. Ela está apresentada na Figura 12.

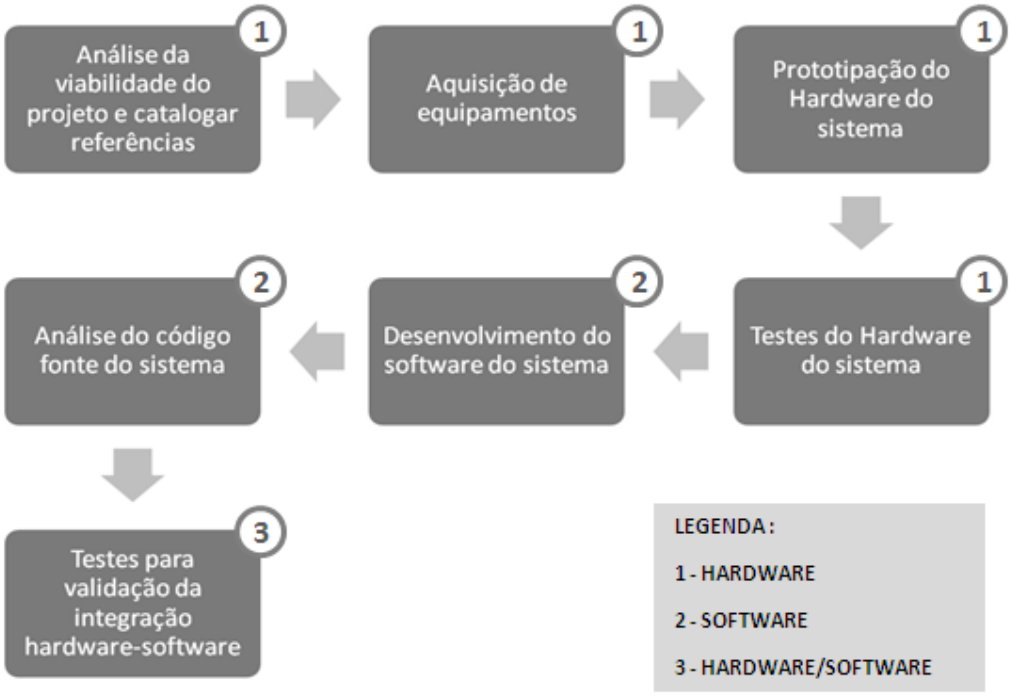


Figura 12. Fluxo de desenvolvimento do projeto.

Pode-se observar na Figura 12 que podemos agrupar as etapas do projeto em 3 partes. Primeiramente o desenvolvimento se concentrou na implementação do *hardware* do trabalho proposto. Logo após, na etapa do *software* necessário para comunicação dos módulos desenvolvidos na primeira etapa. E por fim, os testes para verificação da integração *hardware-software.*

4.5 Placa de Circuito Impresso

O principal objetivo da placa de circuito impresso é a circulação da corrente elétrica entre os componentes. O desenho da placa diz respeito à localização física dos componentes, trilhas e ilhas. A placa impressa proposta foi construída levando em consideração os seguintes aspectos:

- Disponibilidade de espaço na placa para todos os componentes do circuito;
- Proximidade entre as trilhas, evitando ocorrer interferência eletromagnética entre componentes do circuito;
- Trilhas abertas ou trilhas que se cruzam (curto-circuito).

A construção de uma plataforma física de um produto eletrônico consiste em obter componentes discretos, agrupá-los em nível esquemático e projetar o circuito de modo a fornecer uma interconectividade entre os componentes, e fornecer bases para conduzir a elaboração de uma placa de circuito impresso. O arquivo gerado pelo *TraxMaker* é então utilizado para gerar os arquivos necessários para fabricar e montar, fornecer uma documentação de concepção e fornecer uma infra-estrutura a concepção mecânica do produto. O esquemático desenvolvido para o módulo da Plataforma PIC pode ser visualizado na Figura 13. Esse circuito pode ser dividido em quatro partes, são elas: o circuito oscilador, o circuito de gravação *in-circuit*, o circuito de re-inicialização e a comunicação com o *Bluetooth*.

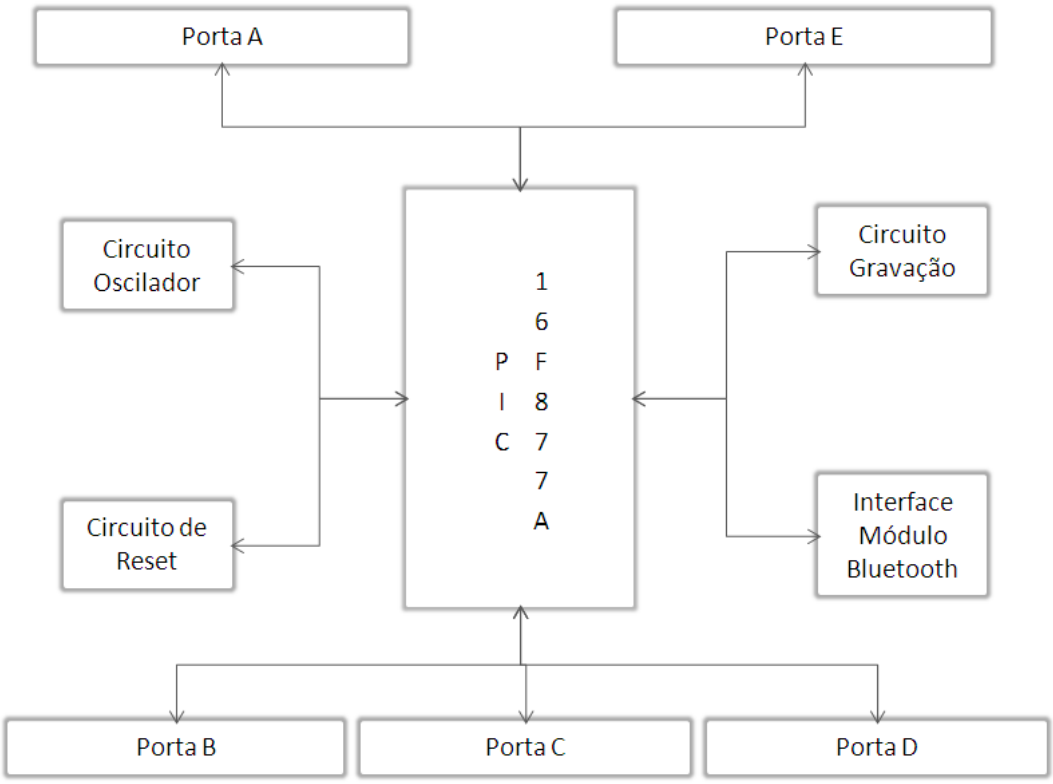


Figura 13. Esquemático da Plataforma PIC em diagrama de blocos.

A descrição do esquemático apresentado na Figura 13 será apresentada nas subseções 4.5.1, 4.5.2 e 4.5.3. O Apêndice C traz uma construção desse esquemático utilizando a ferramenta Proteus.

4.5.1 Módulo Plataforma PIC

O desenvolvimento do Módulo da Plataforma PIC seguiu normas recomendadas pelos fabricantes, tais como: organização de memória, características elétricas, configuração de portas de entrada e saída. Ela foi desenvolvida para que posteriormente fosse interligada a placa com o módulo *Bluetooth KCWirefree* KC-21. O desenho do circuito pode ser visualizado na Figura 14.

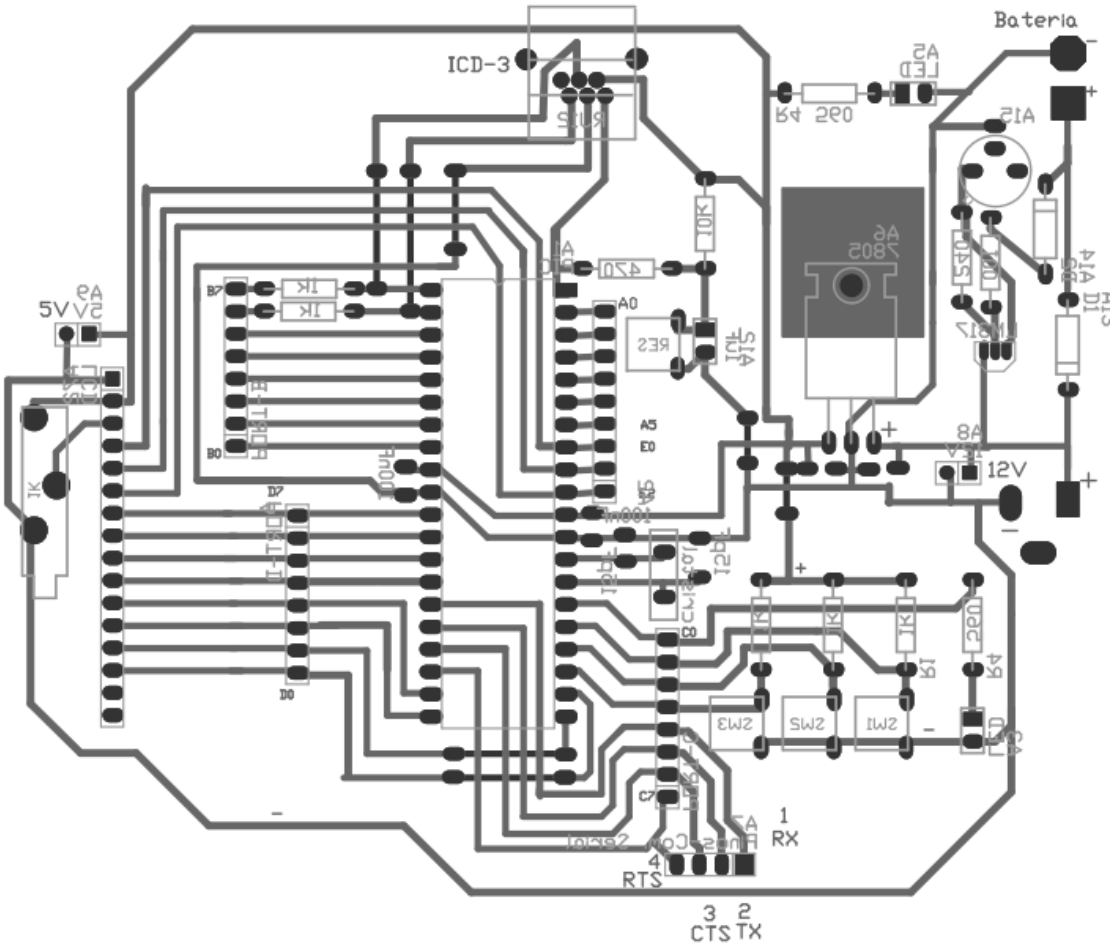


Figura 14. Desenho do circuito impresso da Plataforma de Desenvolvimento PIC

Para o desenvolvimento da Plataforma de Desenvolvimento para o PIC foram utilizados os componentes listados no Apêndice A. Partindo desse desenho do circuito impresso desenvolvido, foi produzida a placa, onde sua face dos componentes está apresentada na Figura 15 e das trilhas na Figura 16.

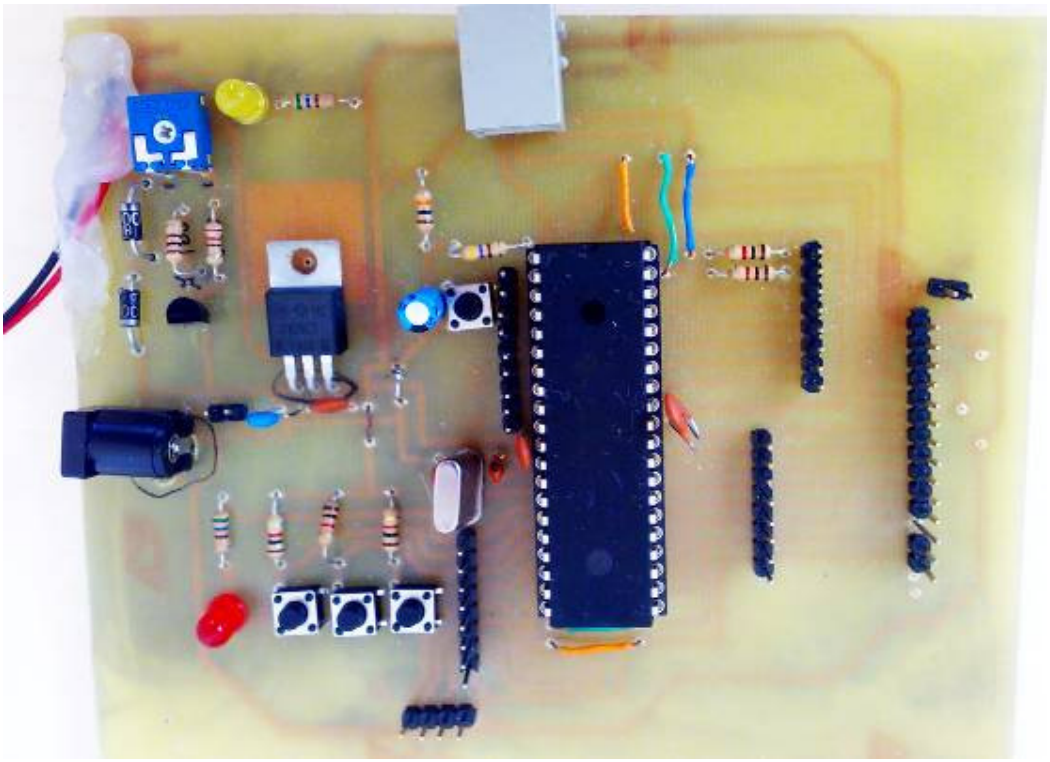


Figura 15. Face dos componentes do circuito desenvolvido para Plataforma PIC

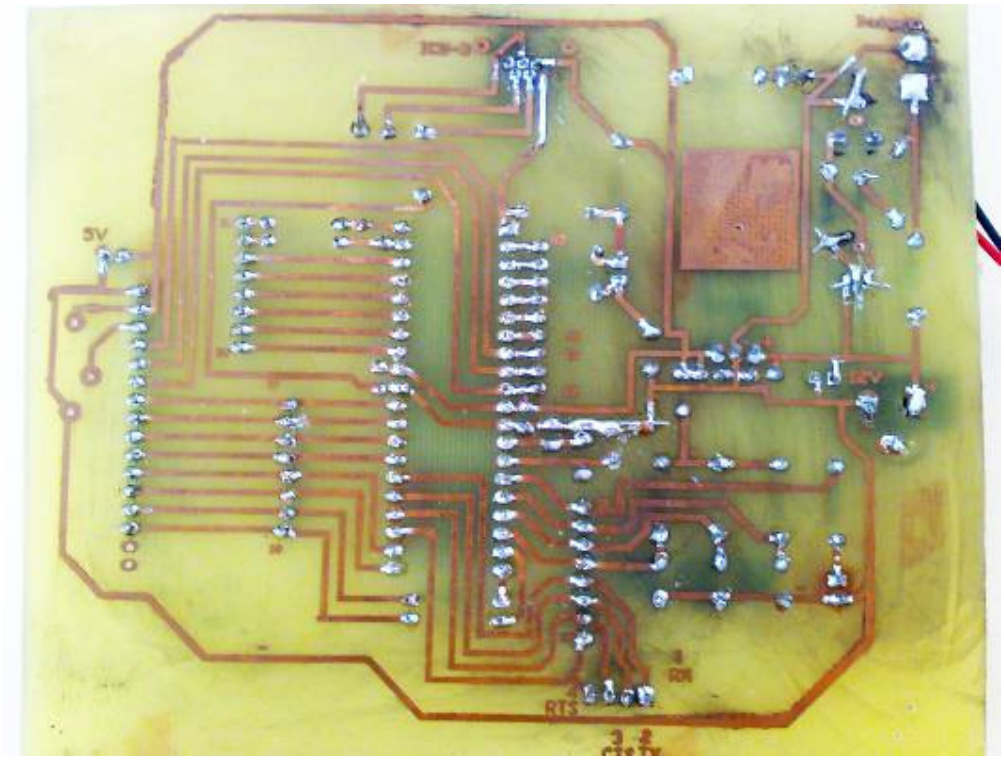


Figura 16. Face das soldas e trilhas da Plataforma PIC

4.5.2 Circuito Oscilador

O circuito oscilador da Placa da Plataforma PIC tem como objetivo fornecer o *clock* de entrada do circuito, ele possui um cristal de 20 MHz e dois capacitores cerâmicos de 15 pF. Esses capacitores servem para melhorar a estabilidade da oscilação. O fabricante recomenda que o circuito oscilador fique próximo ao microcontrolador, pois isso contribui para redução de interferências de outros componentes na placa. A Figura 17 exibe o esquema utilizado para a implementação do circuito de oscilação, nela é observado que as saídas do cristal são ligadas aos pinos 13 (OSC1/CLKIN) e 14 (OSC2/CLKOUT) do microcontrolador. O PIC possui em seu interior um sistema de oscilação do cristal. Devido a isso o cristal pode ser conectado diretamente aos pinos OSC1e OSC2.

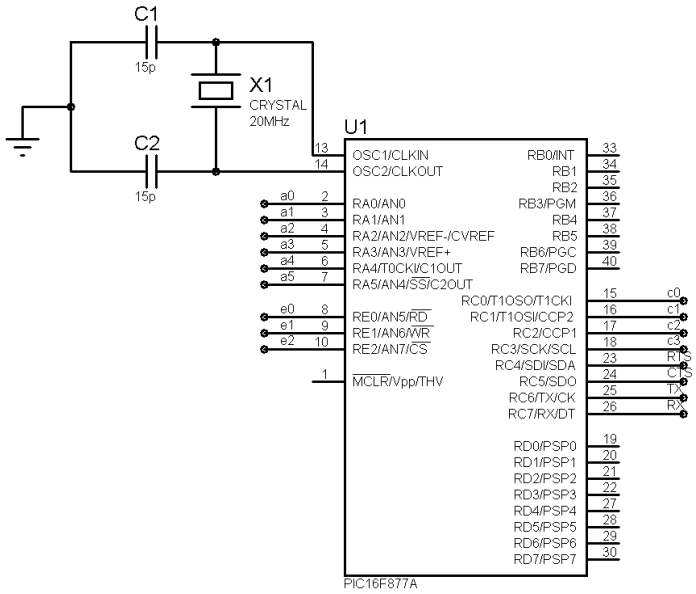


Figura 17. Circuito oscilador da Plataforma PIC.

4.5.3 Circuito *reset*

O circuito *reset* da Placa da Plataforma PIC tem como objetivo fornecer ao circuito a capacidade de restabelecer suas configurações pré-definidas no momento da gravação do código. A Figura 18 exibe o esquema utilizado para a implementação do circuito de reset. A função de *reset* no PIC está relacionada ao

pino 1, denominado MCLR (*Master CLeaR* – sigla para representar o *reset* principal do circuito).

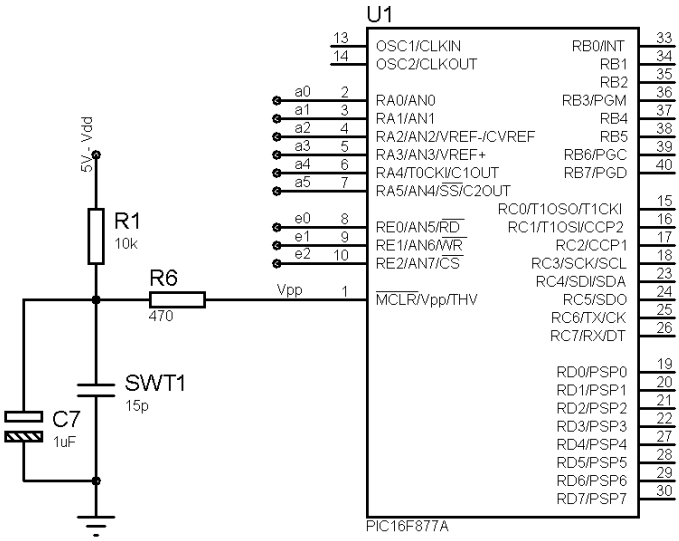


Figura 18. Circuito *reset* da Plataforma PIC.

4.5.4 Módulo *Bluetooth*

O módulo *Bluetooth* tem o objetivo de habilitar a capacidade da placa da Plataforma PIC a realizar transferência de dados utilizando o protocolo Bluetooth. Para o desenvolvimento da placa do módulo *Bluetooth* foram utilizados os componentes listados no Apêndice B. A comunicação serial foi utilizada para realização da comunicação desse módulo com a Plataforma PIC. Para o controle do *chip Bluetooth KCWirefree* KC-21 são utilizados comandos AT. Esses comandos servirão para um módulo interagir com outros dispositivos *Bluetooth*. Conforme apresentado na Figura 19.

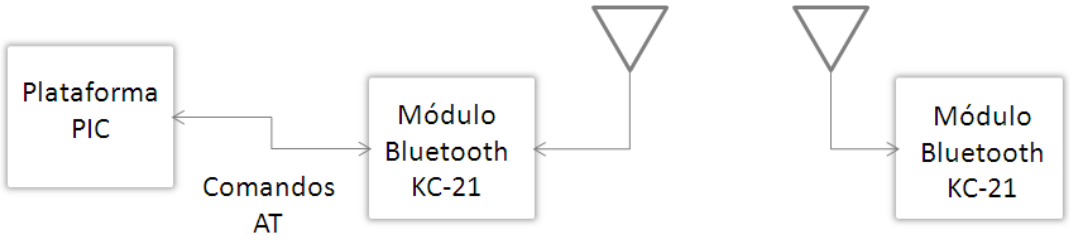


Figura 19. Comunicação da Plataforma PIC com Módulo KC-21 e outro módulo.

O desenho de circuito impresso da placa criada para o Módulo *Bluetooth* é apresentado na Figura 20, ele foi feito utilizando a ferramenta TraxMaker e atende as recomendações do fabricante do módulo.

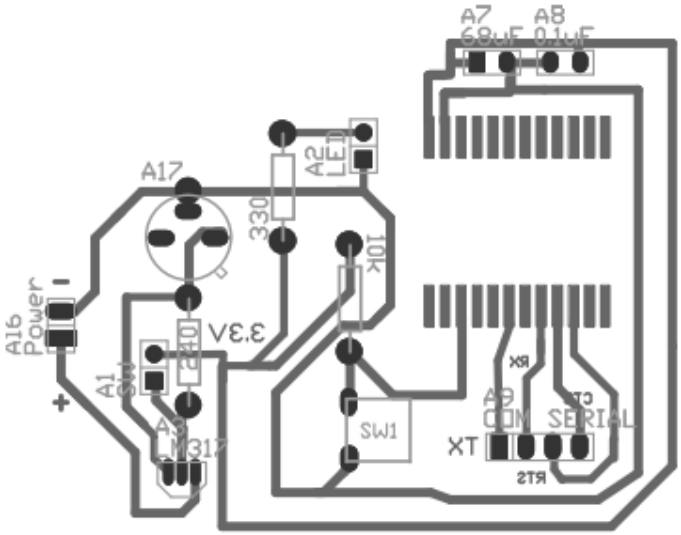


Figura 20. Circuito Elétrico do módulo *Bluetooth*

Partindo do desenho apresentado na Figura 20, foi confeccionada a placa de circuito impresso, onde a face com todos os componentes é mostrada na Figura 21 e das trilhas na Figura 22, com destaque para o *chip Bluetooth KCWirefree* KC-21.

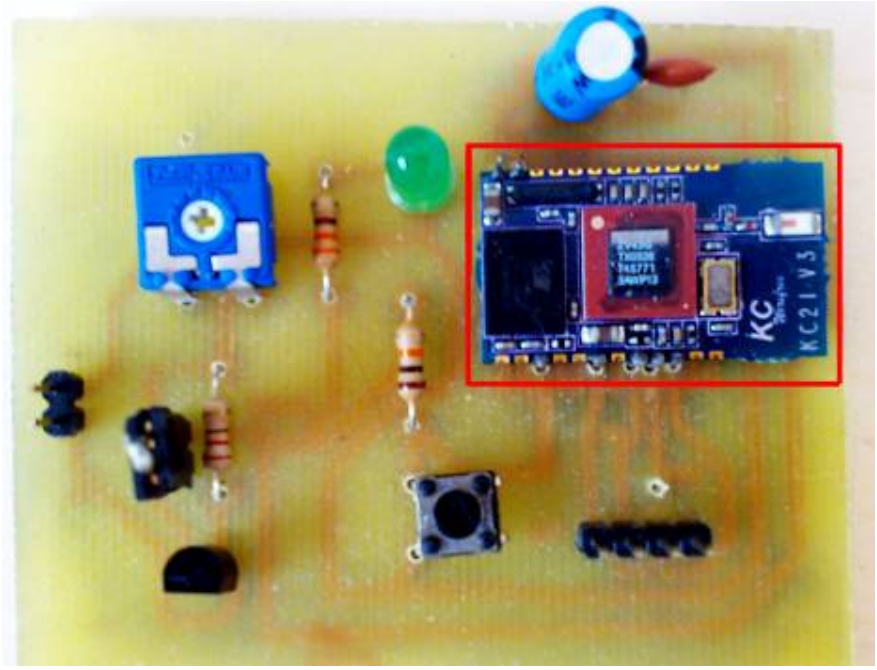


Figura 21. Face dos componentes do circuito desenvolvido para o módulo *Bluetooth.*

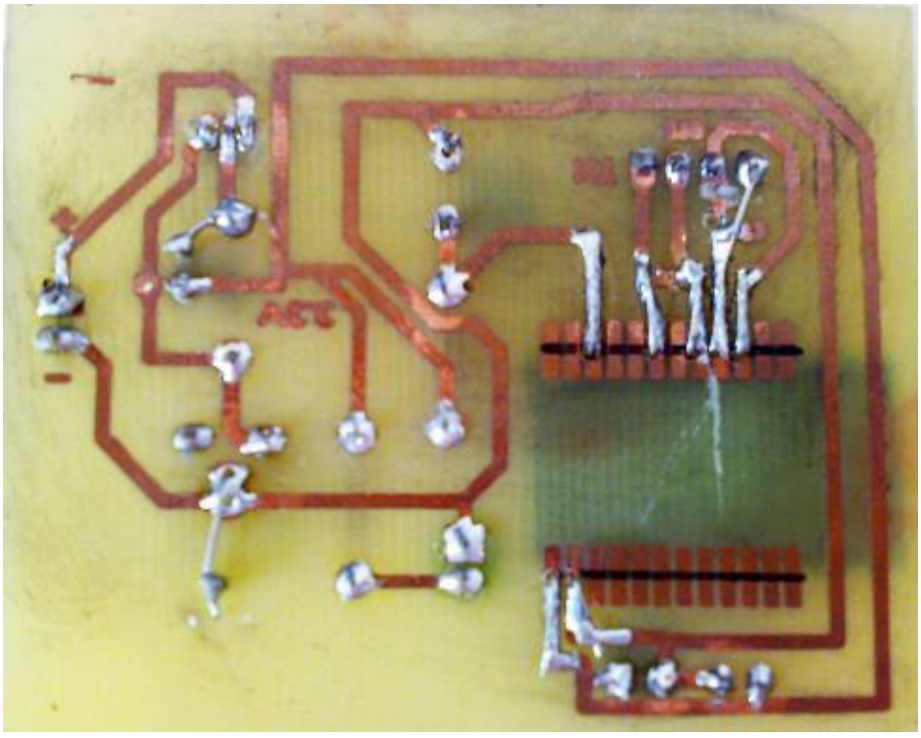


Figura 22. Face das soldas e trilhas do módulo *Bluetooth*

4.6 Processo de Gravação

O processo de gravação, do microcontrolador PIC, utilizada no projeto proposto será descrito nesta seção. Para essa etapa foi utilizado o CCS C Compiler, para o desenvolvimento do código-fonte, compilação e geração do arquivo binário com a extensão "*hex*".

Partindo do arquivo hexadecimal criado pela ferramenta CCS, é a hora de levá-lo para a ferramenta MPLAB IDE, que em conjunto com o gravador MPLAB ICD3 [7] foram fundamentais no processo de importação do arquivo e gravação do código no microcontrolador. Com o arquivo hexadecimal (arquivo que contém o código da aplicação) importado para IDE, o próximo passo é a seleção correta do dispositivo a ser gravado, que para este projeto essa ferramenta foi configurada para o PIC 16F877A. Já passada essa etapa, conecte o MPLAB ICD 3 ao computador e também a placa-alvo, a qual contém o microcontrolador que deseja utilizar, conforme representado na Figura 23. Com os equipamentos interligados, selecione opção programar o dispositivo através da MPLAB IDE.

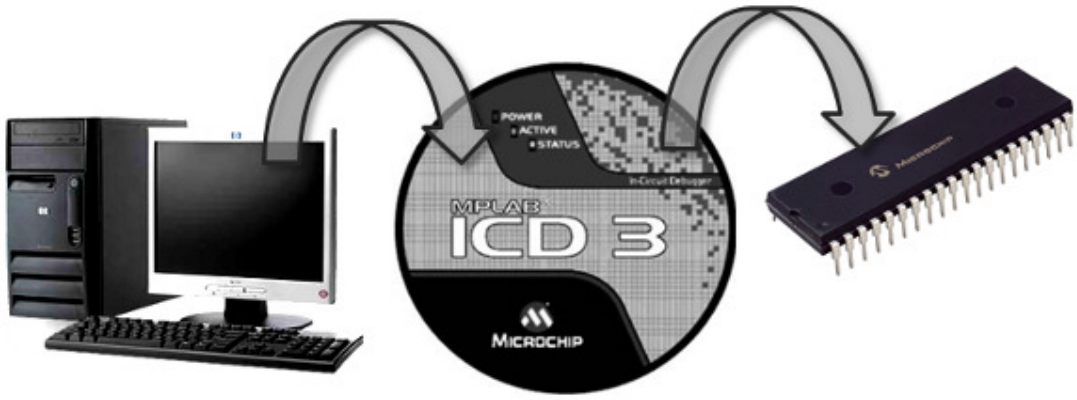


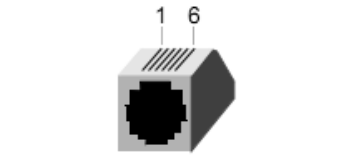
Figura 23. Esquema de transferência do código do computador para o microcontrolador

O método de gravação utilizado foi o *in-circuit*, isso serviu para que o PIC fosse gravado diretamente na placa. Essa gravação é feita de forma serial e utilizando somente 5 pinos de um conector RJ-12. Os pinos utilizados na gravação foram:

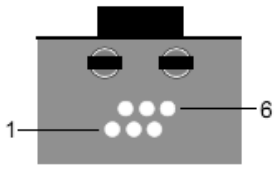
- **Vdd:** Alimentação de 5 volts.
- **Vss:** Terra.
- **MCRL:** Esse pino é necessário para informar ao microcontrolador que ele estará em modo de programação, no momento da gravação esse pino será alimentado por 13 volts, com isso houve a preocupação de proteger todo o resto do circuito dele.
- **RB6:** Pino responsável pelo *clock* da comunicação serial, e sua velocidade é dada pelo gravador.
- **RB7:** Pino responsável pelos dados na comunicação serial, esses são do gravador, no momento de escrita, mas caso esteja no momento de leitura eles serão PIC.

A Figura 24 representa a configuração do conector RJ-12 no processo de gravação utilizando o método *in-circuit*.

conector	PIC
6	não conectado
5	RB6
4	RB7
3	terra
2	VDD
1	MCLR



Visão superior do conector na placa



Visão inferior do conector na placa

Figura 24. Configuração do conector RJ-12 na Plataforma PIC (adaptado de [7])

4.7 Comunicação da Plataforma PIC com o Módulo *Bluetooth* KC-21

A comunicação entre a Plataforma PIC com o Módulo *Bluetooth* foi realizada utilizando uma comunicação serial, para isso foi criada uma interface utilizando os pinos C4 (RTS), C5 (CTS), C6 (TX) e C7 (RX). Esses pinos são responsáveis por enviar os comandos, via serial, para o Módulo *Bluetooth* que a Plataforma PIC irá controlar. A visualização dessa interface baseada no esquemático do PROTEUS pode ser vista na Figura 25. Na Figura 26 mostra como as placas ficaram interligadas.

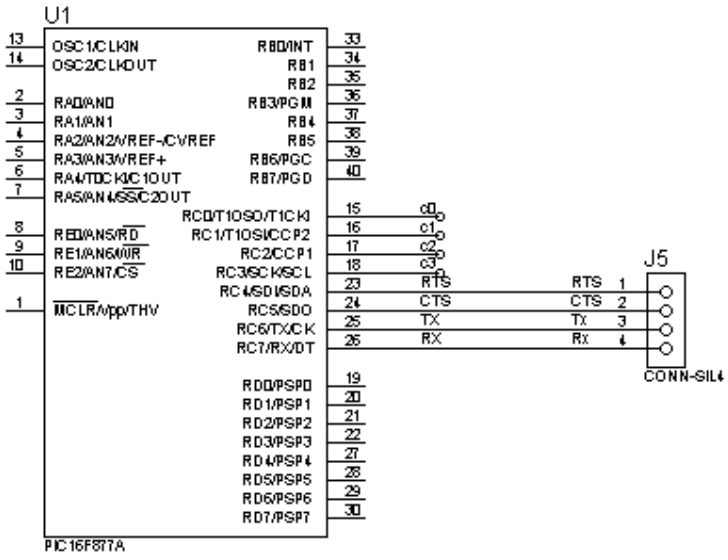


Figura 25. Interface criada para conectar a Plataforma PIC ao Módulo *Bluetooth.*

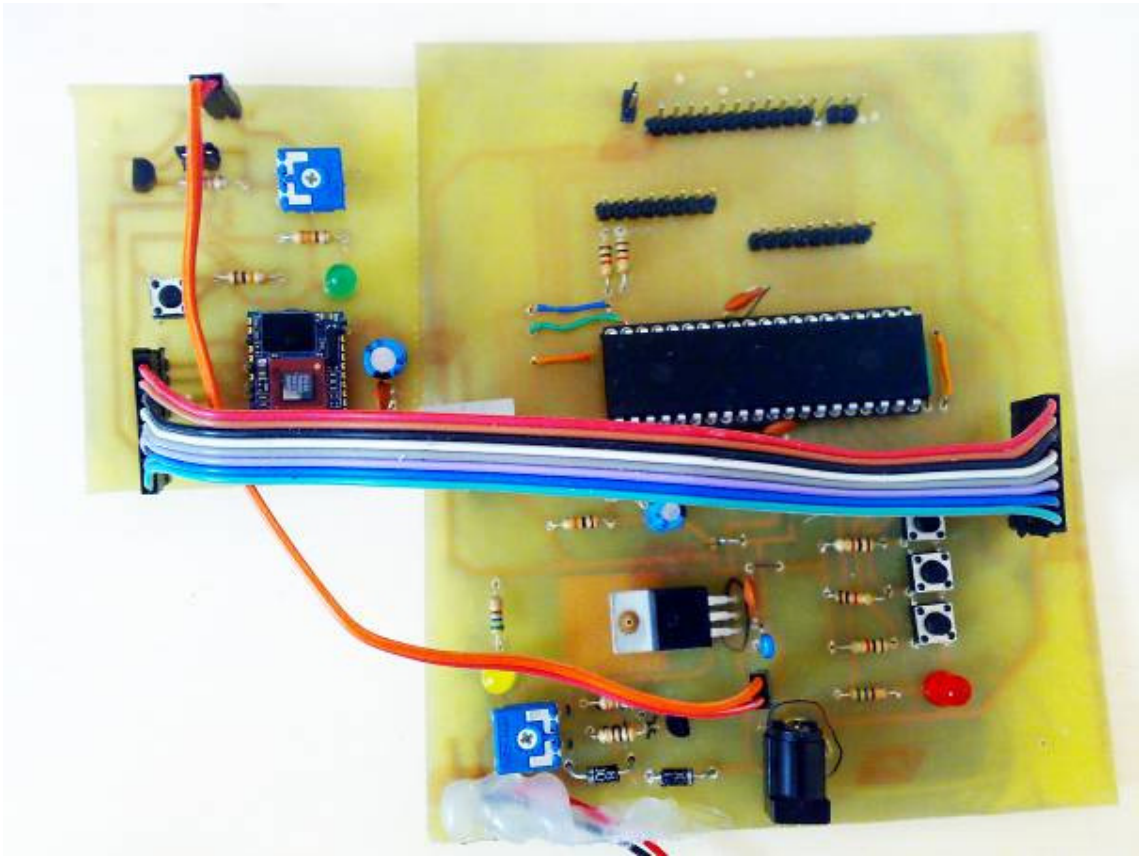


Figura 26. Vista real da interligação das Placas.

4.8 Biblioteca de funções para transmissão *Bluetooth*

A biblioteca de funções é um conjunto de funções extras que tornam transparente ao programador a manipulação do dispositivo *Bluetooth*. Nele existem funções de inicialização, transmissão e recepção de dados usando o protocolo de comunicação *Bluetooth*. Esta biblioteca, escrita em linguagem C, foi desenvolvida para uso em sistemas que usem o microcontrolador PIC 16F877A [28], módulo *Bluetooth KCWirefree* KC-21 [33] e o compilador CCS [29]. As seguintes funções foram implementadas:

- bt_localname_set(char *nome): Define o nome padrão com o qual outros dispositivos enxergarão o dispositivo;

- bt_reset(): Aplica um *reset* no dispositivo;
- bt_connect(char *bd_address): inicia conexão com outro dispositivo;
- bt_disconnect (): finaliza conexão com outro dispositivo;
- bt_send_string(char *string, char *bd_address): Envia uma string a um dispositivo.

Os códigos da implementação destas funções se encontram no Apêndice D.

Capítulo 5

Resultados

Este capítulo descreve as configurações, testes e resultados obtidos a partir da modelagem, implementação e aplicação dos módulos descritos no Capítulo 4. Para os testes foram utilizadas os dois pares de placas desenvolvidas, pode-se entender cada par como a Figura 26, ou seja, duas placas de Plataformas PIC e outras duas placas de Módulo *Bluetooth*.

5.1 Teste do *hardware* do projeto proposto

Após a etapa de desenvolvimento do *hardware* do projeto proposto, foram iniciados testes para sua verificação. Inicialmente foi checado se a placa possuía curtos-circuitos, o que não aconteceu. Logo após houve verificação dos níveis de tensão nas alimentações dos principais componentes (kC-21 *Wirefree* e Microcontrolador PIC), onde era necessário garantir que o KC-21 *Wirefree* seja alimentado por 3,3 volts e o microcontrolador por 5 volts. Posteriormente foi desenvolvido um código somente para validação da Plataforma PIC desenvolvida, com ele, pôde-se constatar que a placa Plataforma PIC estava com os módulos de gravação e portas funcionando corretamente. Nessa etapa não foram encontradas dificuldades.

5.2 Comunicação entre a Plataforma PIC e o módulo *Bluetooth.*

Após as duas etapas descritas na seção 5.1 foi iniciado o processo de tentativas de se estabelecer uma comunicação entre a Plataforma PIC e o módulo *Bluetooth.* Foram encontradas algumas dificuldades para realização de uma comunicação com sucesso. Principalmente, em tratar as informações recebidas do KC-21*wirefree*. Para um melhor entendimento de como funcionou a comunicação

entre a Plataforma PIC e o módulo *Bluetooth* observe a Figura 27. Ela apresenta o método de troca de mensagens no interior do KC-21 e dele com componentes externos ao módulo.

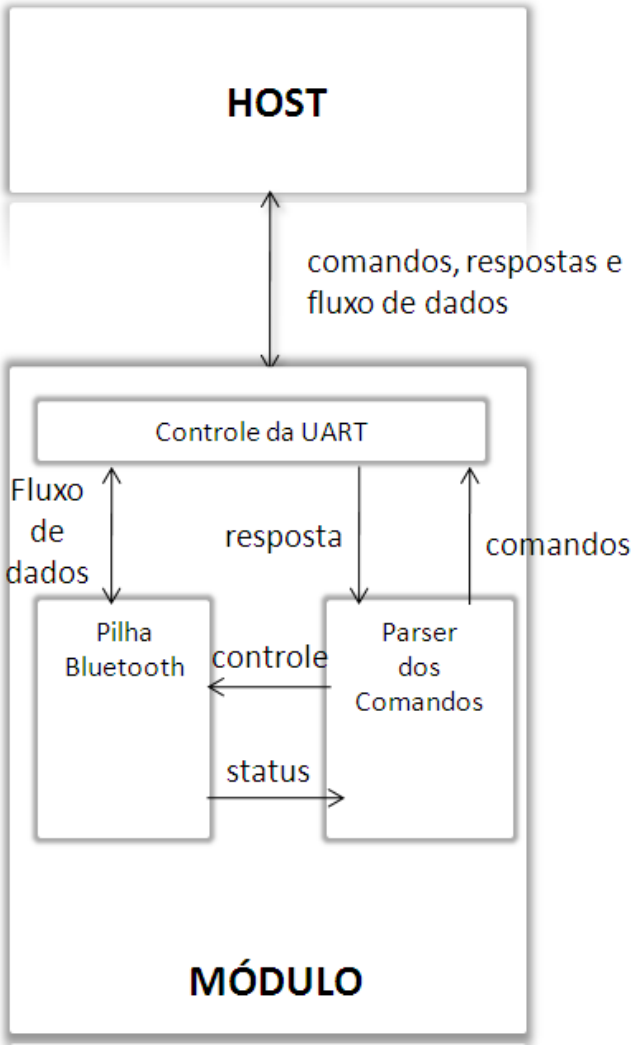


Figura 27. Comunicação HOST (Plataforma PIC) e o Módulo *Bluetooth.*

No módulo *Bluetooth*, os dados de entrada e saída da aplicação são de dois tipos. O primeiro é para comandos e respostas; vale lembrar que esses comandos são no formato AT e são fornecidos pelo fabricante do KC-21 *Wirefree*. Esses comandos e respostas somente são manipulados enquanto a aplicação estiver no modo de operação denominado *linha de comandos*. Já o segundo está relacionado ao módulo conhecido como *Bypass*, nele todo fluxo de dados que é recebido através da UART será propagado pela pilha *Bluetooth*.

Para critério de testes da comunicação da Plataforma PIC com o Módulo *Bluetooth*, foi inicialmente tentado modificar o nome padrão do dispositivo, que inicialmente vem com o nome "*KCWirefreeDevice*", O método utilizado para leitura do nome do dispositivo foi através de um celular habilitado com *Bluetooth.* A Figura 28 mostra a leitura que foi feita inicialmente pelo celular, apresentando o nome padrão do dispositivo.

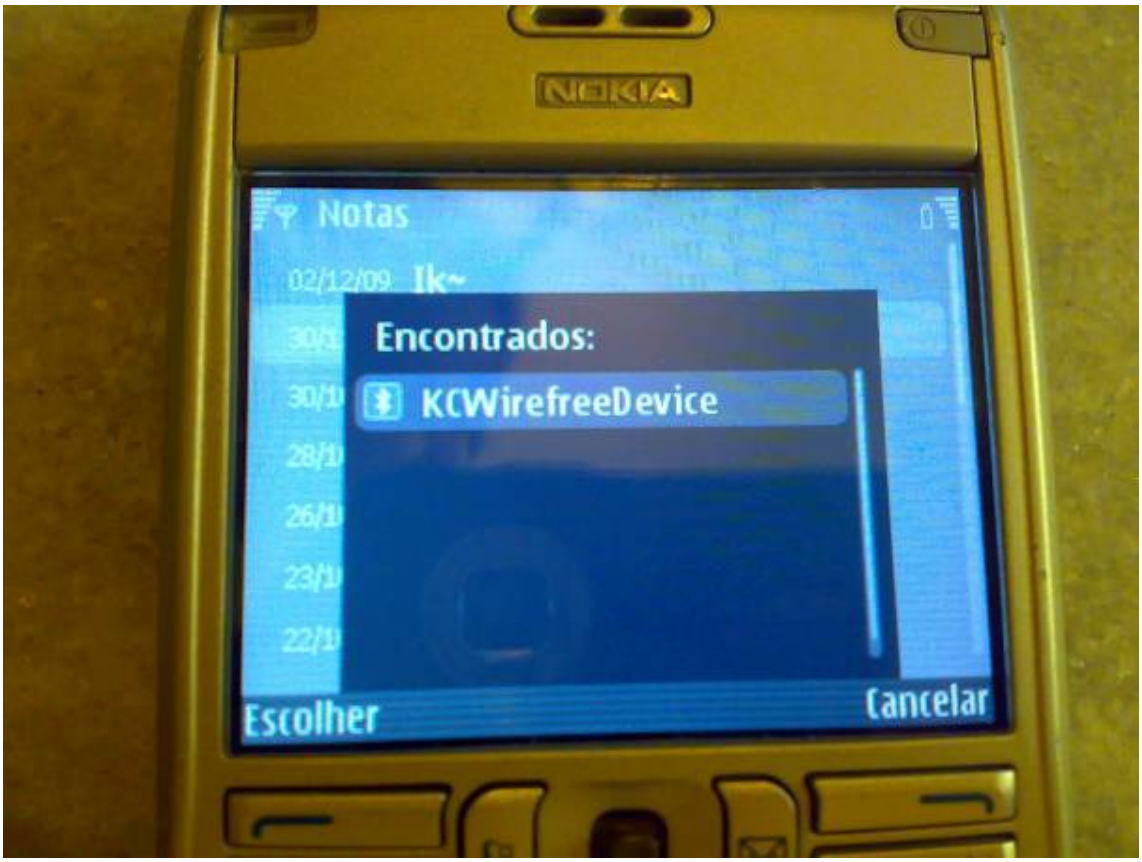


Figura 28. Leitura feita pelo celular que apresenta o nome padrão do dispositivo.

Para redefinir o nome padrão do dispositivo *Bluetooth* de modo permanente foi utilizado o seguinte comando AT:

- AT+ZV DefaultLocalName [nome]

Onde [nome] deve ser uma palavra com até 50 caracteres, onde também são permitidos inserção de caracteres de espaço. Um dos problemas para execução do comando foi definir da velocidade padrão da UART, que deve ser configurada para

115200 *baud.* Isto pode ser configurado através da diretiva responsável pela comunicação serial, conforme apresentado abaixo:

- #use rs232(baud=115200, xmit=PIN_C6, rcv=PIN_C7, parity = N, ERRORS)

Essa configuração foi feita inicialmente com o baud igual a 9600, e foi percebido que a modificação do nome não foi realizada com sucesso. Após a configuração correta do *baud* para o valor de 115200, novamente foi executado o comando para mudança do nome padrão, que no código-fonte está conforme apresentado abaixo:

- printf("AT+ZV DefaultLocalName modBT\n").

Após a execução e a tentativa de ler o nome do dispositivo, novamente através do celular, constatou-se a mudança do *DefaultLocalName,* conforme apresentado na Figura 29. O código completo está apresentado no Apêndice E.

Figura 29. Leitura feita pelo celular que apresenta o nome padrão do dispositivo modificado com sucesso.

Após o estabelecimento da comunicação entre a Plataforma PIC e o Módulo *Bluetooth*, foi iniciado o processo de tentativas de se estabelecer uma comunicação entre o módulo *Bluetooth* com outro dispositivo *Bluetooth* externo. Esta fase apresentou algumas dificuldades, principalmente no tratamento das mensagens e dados recebido pelo KC-21 *Wirefree*. Para um melhor entendimento de como funciona a comunicação entre a Plataforma PIC com um módulo *Bluetooth* com outro dispositivo KC-21 *Wirefree*, veja a Figura 30. Ela apresenta o método de troca de mensagens do KC-21 "mestre" com o KC-21 "escravo", onde inicialmente é criada uma comunicação remota do dispositivo mestre com o escravo, inicialmente configurando ambos os módulos no modo de comandos, logo após o mestre envia o comando abaixo:

- AT+ZV SPPConnect [BDAddrB],

Onde o [BDAddrB], apresentado no comando acima, corresponde ao endereço físico do dispositivo *Bluetooth* escravo. Uma vez a conexão estabilizada, a aplicação passa para o modo *Bypass,* e com isso todos os dados enviados para a porta serial do dispositivo mestre é também enviado para serial do dispositivo escravo.

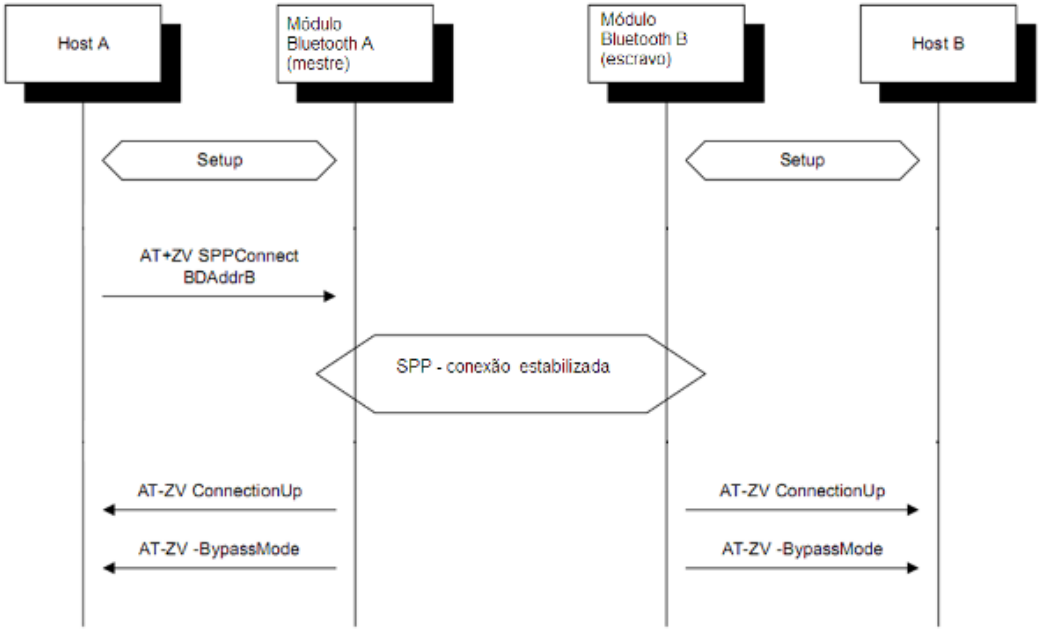


Figura 30. Processo de conexão estabilizada com sucesso.

Foram realizadas tentativas de estabelecimento de conexão com um *notebook* equipado com um *Bluetooth Dongle*. Onde o *notebook* tentou inicialmente enviar um simples arquivo e nenhum dado foi detectado no tratamento das interrupções. O inverso também foi realizado, enviando dados do Módulo *Bluetooth* conectado a Plataforma PIC para o notebook, e também a transferência de dados não foi bem sucedida. Não foram realizados testes com osciloscópio para verificação de sinais recebidos pelo módulo *Bluetooth*, uma das prováveis soluções é melhorar o tratamento da recepção de mensagens pela serial para facilitar uma melhor investigação do problema. O fabricante informa que uma provável desconexão pode ocorrer, caso o dispositivo fique fora do alcance de recepção. O que não se adéqua para o problema atual em questão. Outra possibilidade seria dividir o envio da *string* em blocos menores para verificar se há algum problema quanto ao tamanho do arquivo enviado.

Novas tentativas de estabelecimento de conexão foram feitas. Nessas tentou-se criar um simples pareamento entre dispositivos distintos (módulo *Bluetooth* e um *Bluetooth Dongle* conectado ao computador pessoal). Nessa tentativa o *KCWirefree KC-21* ao invés de utilizar o comando “AT+ZV SPPConnect [BDAddrB]” utilizou o “AT+ZV Bond [BDAddrB] [senha]”. O comando *Bond* faz com que o dispositivo faça uma requisição para o estabelecimento de conexão, depois o dispositivo “requisitado” autoriza a requisição e informa a senha, caso essa seja requerida.

Nessas novas tentativas foi identificado que o compilador CCS só gera os separadores de comandos AT <CR><LF> quando se escreve “putc (0x0d)” para o <CR> e “putc (0x0a)” para <LF>, no código C. A partir das modificações dos “printf(\r\n)” por “putc (0x0d)” e “putc (0x0a)”, nesta ordem, os comandos foram executados corretamente.

Após resolver o problema de envio de comandos, foi feita uma comunicação entre o *KCWirefree KC-21* e um computador pessoal na seguinte forma:

1. Habilitar o computador pessoal para receber requisições de conexão, conforme Figura 31;

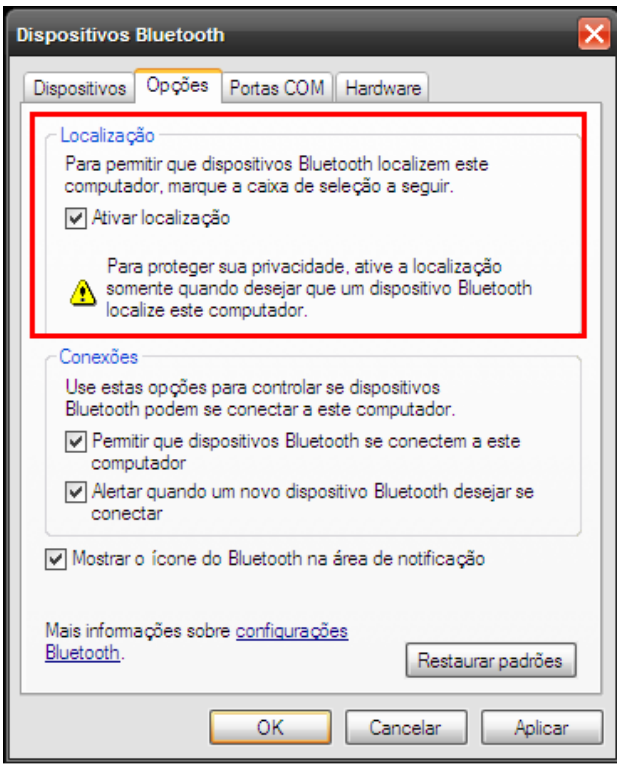


Figura 31. Habilitar o computador pessoal para que outros dispositivos o localizem.

2. Plataforma PIC envia o comando “AT+ZV Bond [bd_address] [senha]”, onde o [bd_address] é o endereço do dispositivo *Bluetooth* conectado ao computador pessoal e a [senha] é o código de acesso que o computador deve informar para autorizar a requisição;

3. Depois de enviado o comando o computador pessoal receberá o seguinte aviso, apresentado na Figura 32;

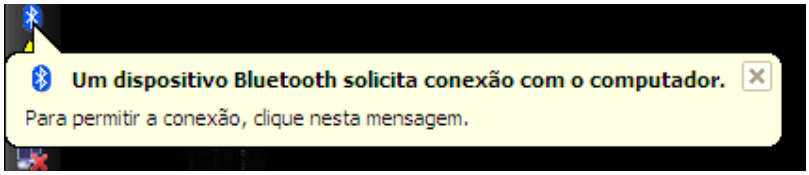


Figura 32. Recebimento de requisição pelo computador pessoal.

4. Depois de clicar no balão apresentado na Figura 32, será solicitada a senha de acesso conforme apresentado na Figura 33;

Figura 33. Solicitação de senha de acesso.

5. Apos inserir a senha “1111” (definida na programação do PIC) a comunicação é concluída como apresentada na Figura 34;

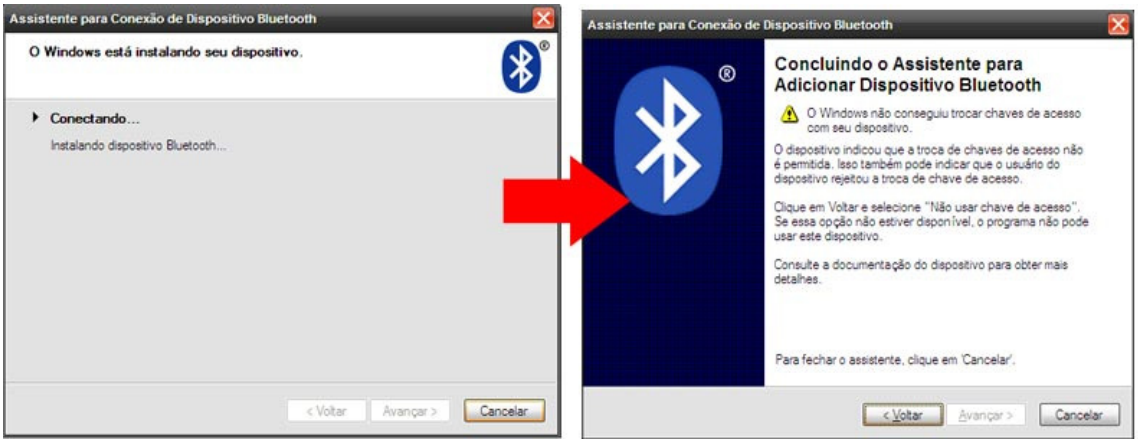


Figura 34. Conclusão estabelecimento de conexão.

6. Verifique a porta COM (serial) na qual o módulo *Bluetooth* está conectado, conforme destacado na Figura 35. Nesse teste o módulo se encontrava na porta COM24;

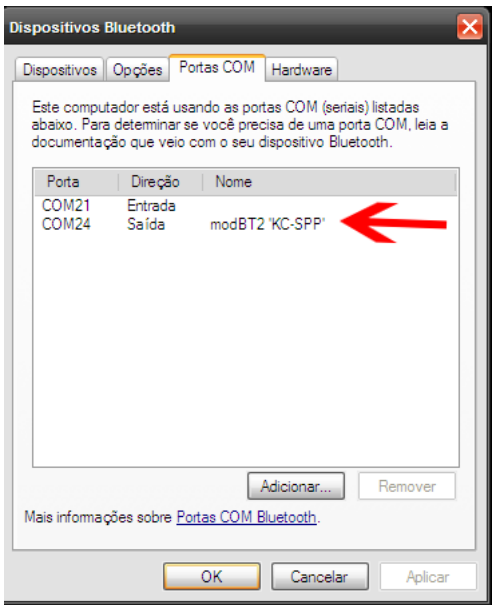


Figura 35. Portas COM do *Windows*.

7. Verifique os dispositivos *Bluetooth* que estão conectados ao computador pessoal. Selecione o dispositivo que corresponde ao módulo *KC-Wirefree* e depois clique em suas propriedades. Na janela de Propriedades do dispositivo, vá à aba serviços e marque o dispositivo como um serviço;

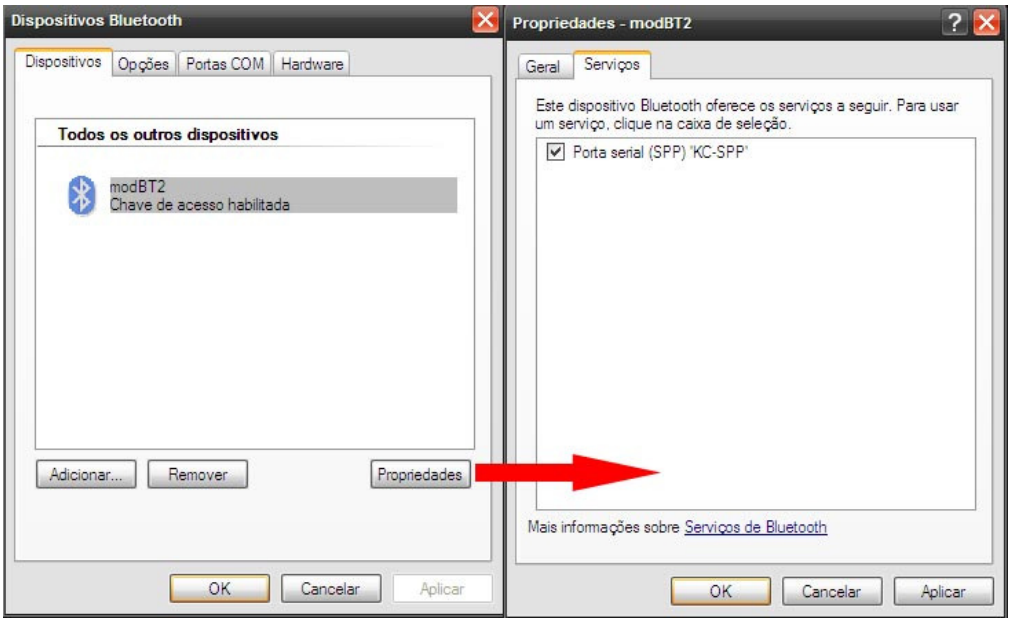


Figura 36. Configuração do dispositivo *Bluetooth* como serviço.

8. Abra o programa *Hyperterminal,* no computador pessoal com sistema operacional *Windows,* depois configure conforme Figura 37*;*

Figura 37. Configuração do *Hyperterminal*.

9. A partir deste momento, todos os dados que forem enviados ao *KCWireFree* KC-21, via comunicação serial, também serão enviados ao computador pessoal.

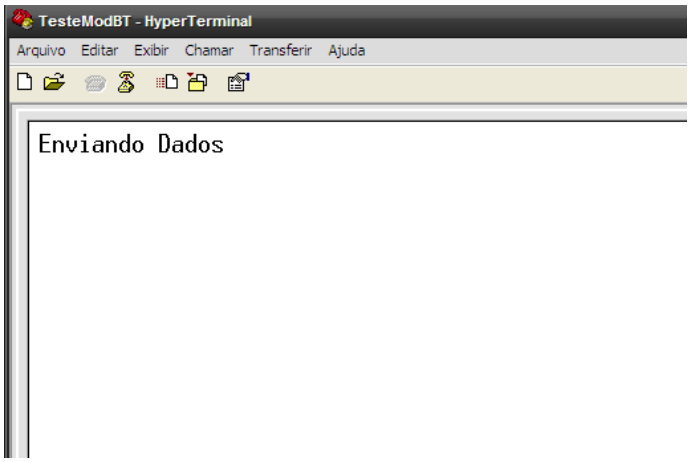


Figura 38. Dados enviados pelo módulo *Bluetooth* para o computador pessoal.

Capítulo 6

Conclusão e Trabalhos Futuros

O trabalho proposto foi desenvolvido com o objetivo de implementar um módulo de comunicação *Bluetooth* utilizando microcontrolador PIC. É esperado que ele contribua para o ensino dessa tecnologia para futuras turmas da Universidade de Pernambuco.

Além disso, o trabalho proposto espera que esse módulo proporcione uma fácil implementação, eliminando assim o tempo gasto para elaboração de um módulo base para comunicação *Bluetooth*.Assim, uma abstração dos detalhes de seu funcionamento é fornecida tornando-os transparentes ao desenvolvedor. Outra meta almejada é que o sistema proposto sirva para ferramenta de apoio a pesquisas relacionadas à comunicação sem fio na universidade.

A intenção futura é que o módulo desenvolvido seja uma opção para transmissão de dados sem fio em um ambiente de monitoramento de corrente de fuga em cadeias de isoladores de linhas de alta tensão. Existe a possibilidade de estender o modulo desenvolvido para que ele realize a comunicação e transmissão de dados entre ele e um computador pessoal, permitindo assim a interação dele com aplicações de PC.

Outros esforços futuros serão o sentido de analisar o desempenho entre as tecnologias de transferência de dados. Na plataforma apresentada poderão ser criadas interfaces avançadas como GPRS, ZigBee, entre outras. Além da comparação de desempenho, outra característica importante que poderá ser implementada através dessas interfaces é a tolerância a falhas. Caso um módulo escolhido como padrão de comunicação não funcionar de forma adequada, o sistema futuro estará habilitado a possibilitar outra forma de transmissão de dados. Ou ainda, provê a melhor tecnologia de transferência, baseada no desempenho, dentre as demais interfaces disponíveis na plataforma.

Outra estratégia de tolerância a falhas a ser implementada diz respeito a redundância por meio do *hardware*. A técnica a ser utilizada é denominada “modo espelho” [34]. Ela consiste na existência de no mínimo 2 microcontroladores PIC funcionando de forma sincronizada. Dessa forma não haverá interrupção na transferência de dados, caso um único microcontrolador pare de funcionar.

Bibliografia

[1] THE BLUETOOTH SIG STANDARD. Documentação oficial do Bluetooth. Disponível em: http://www.bluetooth.com/. Acesso em 25 de agosto de 2009.

[2] HAARTSEN, J. C.; MATTISSON, S. Bluetooth – A New Low-Power Radio Interface Providing Short-Range Connectivity. IEEE Proceedings of the IEEE. Versão 88. Número 10. Pág. 1651-1652. 2000.

[3] KUROSE, J. F. Redes de Computadores e a Internet: uma abordagem top-down. Edição 3ª. Editora Pearson Addison Wesley. 2006.

[4] KOBAYASHI, C. Y. A Tecnologia Bluetooth e Aplicações. USP. São Paulo. 2004

[5] SIQUEIRA, T. F.; MENEGOTTO , C. C.; WEBER, T. S.; NETTO, J. C.; WAGNER, F. R. Desenvolvimento de Sistemas Embarcados para Aplicações Críticas. UFRGS. 2003.

[6] SOUZA, D. J. Desbravando o PIC. Edição 8ª. Editora Érica, 2005

[7] Manual do usuário. MPLAB ICD 3 *In-Circuit Debugger.*

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

[9] PEDRALHO, A. Bluetooth: Da teoria à prática. O mundo sem cabos – Parte I. Número 3, Pág. 16-20, 2005.

[10] Specification of the Bluetooth System. Versão 1.0. Pág. 19. 1999.

[11] CONTI, M. e MORRETI, D. System Level Analysis of the Bluetooth Standard. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition. 2005.

[12] BOER, J. Direct Sequence Spread Spectrum Physical Layer Specification IEEE 802.11, 1996.

[13] LANGTON, C. Intuitive Guide to Principles of Communications. Orthogonal Frequency Division Multiplex (OFDM) Tutorial.

[14] Especificação do protocolo 802.11. Disponível em http://www.ieee802.org/11/. Acessado em 20 de outubro de 2009.

[15] Especificação do protocolo 802.15. Disponível em http://www.ieee802.org/15/. Acessado em 20 de outubro de 2009.

[16] REZENDE, J. F. Notas de aula. UFRJ. Disponível em http://www.gta.ufrj.br/~rezende/cursos/eel879/trabalhos/80211/FHSS.htm. Acesso em 10 de outubro de 2009.

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

[18] PERROTT, M. H. High Speed Communication Circuits and Systems, Spring, 2003.

[19] CHOMIENNE, D. EFTIMAKIS, M. Bluetooth Tutorial. Disponível em http://www.newlogic.com/products/Bluetooth-Tutorial-2001.pdf. Acesso em 15 de outubro de 2009.

[20] KARDACH, J. Bluetooth Architecture Overview. Intel Technology Journal. 2000.

[21] MCDERMOTT-WELLS, P. Bluetooth Overview. IEEE - *Potencials Magazine*. Pág. 33-35. 2004.

[22] SOUZA, D. J. Conectando o PIC – Recursos Avançados. Editora Érica. Edição 3. 2000.

[23] SCHILDT, H. C Completo Total. Editora McGraw Hill, 1990.

[24] SENNE, E. L. F. Primeiro Curso de Programação em C. Editora Visual Books. 2006.

[25] RITCHIE, D. M.; KERNIGHAN, B. W. The C programming Language. Editora Prentice-Hall, 1989.

[26] LOUDEN, K. Compiladores: principios e praticas. Editora Thomson. Pág. 1. 2004.

[27] CCS Inc. Home. Site Oficial do compilador CCS. Disponível em http://www.ccsinfo.com/. Acessado em 08 de junho de 2009.

[28] *Data Sheet* do microcontrolador PIC16F877A disponível em ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf. Acessado 1º de outubro de 2009.

[29] CCS Inc. Manual CCS C Compiler. 2007.

[30] Site do fabricante do PROTEUS. Disponível em http://www.labcenter.co.uk/. Acessado em 10 de novembro de 2009.

[31] CAMPELLO, S. *Tutorial PROTEUS.* Recife, PE. Versão 7.1. 2007. Pág. 1.

[32] SANTOS, L. Sistema de comunicação USB com microcontrolador. Monografia para Graduação em Engenharia da Computação, UPE. Pernambuco. 2009.

[33] *Data Sheet* do módulo *KC-21 Bluetooth OEM.* Disponível em www.kcwirefree.com. Acessado em 1º de outubro de 2009.

[34] FERNANDES, S. M. M. *Estudo e Implementação do Mecanismo de Tolerância a Falhas em software por meio de Blocos de Recuperação*. 1995. 26 f, Tese de Mestrado em Engenharia Eletrônica, Universidade Federal de Pernambuco, Pernambuco. 1995.

Apêndice A

Lista dos Materiais da Placa da Plataforma

Para o desenvolvimento da Plataforma de Desenvolvimento para o PIC, conforme ilustrada na Figura 14, foram utilizados os seguintes componentes:

- 1 PIC 16F877A;
- 1 Soquete DIP 40 pinos;
- 1 Conector RJ-12;
- 1 Regulador de tensão tipo 7805 e 1 LM317 TO-92B;
- 2 Diodos 1N4001;
- 4 Chaves Push-Button tipo NA;
- 2 LEDs;
- 4 Conectores tipo SIP de 8 pinos, 2 conectores SIP de 2 pinos, 1 Conector SIP de 4 pinos e 1 Conector SIP de 14 pinos e 1 conector para fonte do tipo *Jack*;
- 1 Capacitor de 0.1uF, 1 Capacitor de 0.33uF, 1 Capacitor de 1uF, 2 Capacitores de 15pF e 2 Capacitores de 100nF;
- 2 TIMPOTs, 1 Clip de bateria e 1 Cristal de 20Mhz;
- 5 resistores de 1k 1/8W, 2 resistores de 560 1/8W, 1 resistor de 100 1/8W, 1 resistor de 240 1/8W, 1 resistor 470 1/8W e 1 resistor de 10k.

Apêndice B

Lista dos Materiais da Placa do Módulo *Bluetooth*

- 1 Módulo Bluetooth KC-21 v3;
- 1 Resistor 10k 1/8w, 1 resistor 330 1/8w e 1 resistor 220 1/8w;
- 1 LED;
- 1 LM317 TO-92B;
- 1 TRIMPOT;
- 1 Capacitor 68uF e 1 Capacitor 0.1uF;
- 1 Conector SIP 4 pinos e 1 Conector SIP 2 Pinos;
- 1 Chave.

Apêndice C

Esquemático da Plataforma PIC desenhado no Proteus

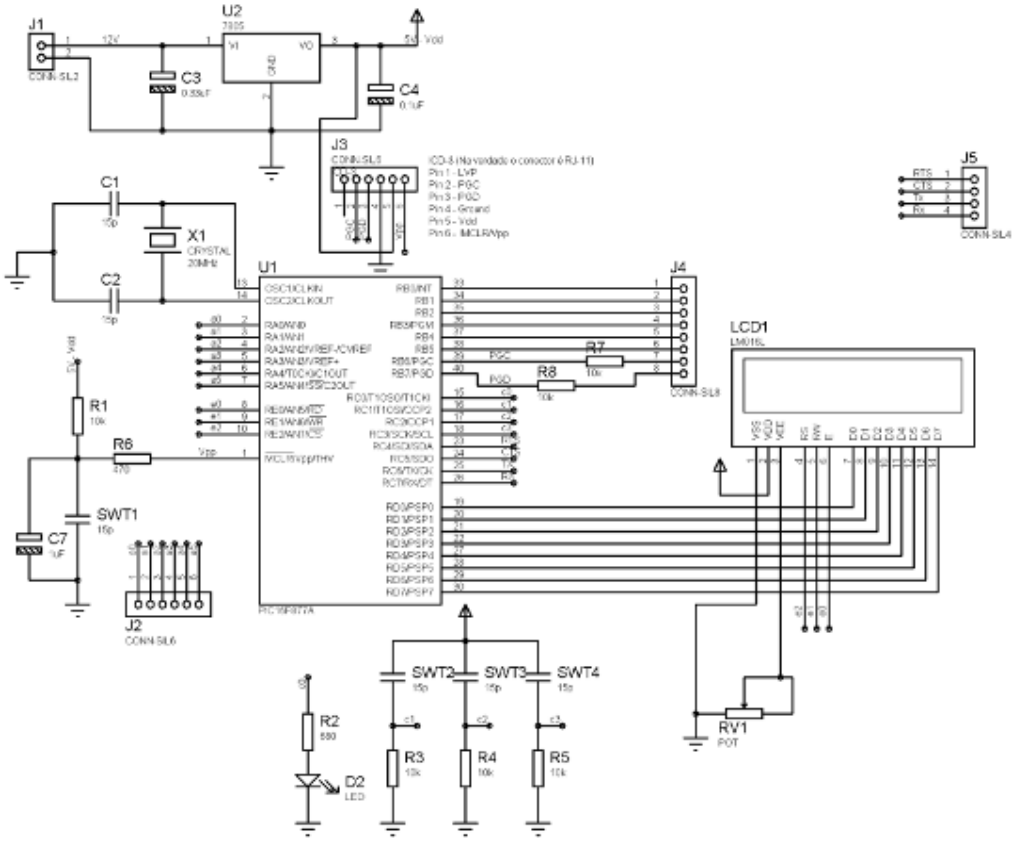


Figura 39. Esquemático da Plataforma PIC desenvolvido no Proteus.

Apêndice D

Códigos da API proposta

Os códigos seguintes pertencem à implementação do pacote bt_pic_api.

```
/**
 * Define o nome com o quais outros dispositivos enxergarão o dispositivo
 * @package bt_pic_api
 */
void bt_localname_set(char *novoNome)
{
        char buffer[37];
        int n;
        n=sprintf (buffer, "AT+ZV DefaultLocalName %s", novoNome);
        printf("%s", buffer);
        putc(0x0d); //CR
        putc(0x0a); //new line
}
/**
* inicia conexão com outro dispositivo
* @package bt_pic_api
*/
void bt_connect(char *bd_address, char *senha)
{
        char buffer[40];
        int n;
```

```
        n=sprintf (buffer, " AT+ZV Bond %s %s", bd_address, senha);
        printf("%s", buffer);
        putc(0x0d); //CR
        putc(0x0a); //new line
}
/**
* Envia uma string a um dispositivo
* @package bt_pic_api
*/
void bt_send_string(char *string)
{
        printf("%s", string);
}
/**
* Aplica um reset no dispositivo
* @package bt_pic_api
*/
void bt_reset()
{
        printf("AT+ZV Reset");
        putc(0x0d); //CR
        putc(0x0a); //new line
}
```

Apêndice E

Códigos do programa para chamada das funções da API proposta

```
#include <16F877a.h>  // inclui arquivo de bibliotecas do dispositivo

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

#use delay (clock = 20000000)

#fuses HS, NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP

#use rs232(baud=115200, xmit=PIN_C6, rcv=PIN_C7, parity = N, ERRORS) //configuração da rs232

#int_rda

void reception ()

{

  char temp;

  temp = getc();

  printf("Recebeu: %s", temp);

}

void main(void)

{

  /* Habilita a interrupção serial */

  enable_interrupts(GLOBAL);

  enable_interrupts(int_rda);

 //aqui chamar funções da API

}
```