CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS DIVINÓPOLIS
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

Apolo Malta

DESENVOLVIMENTO DE UM DISPOSITIVO DE BAIXO CUSTO VISANDO OTIMIZAR O ATENDIMENTO EM ESTABELECIMENTOS QUE NECESSITAM DA PRESENÇA DE UM ATENDENTE

Divinópolis-MG
2014

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS DIVINÓPOLIS
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

Apolo Malta

DESENVOLVIMENTO DE UM DISPOSITIVO DE BAIXO CUSTO VISANDO OTIMIZAR O ATENDIMENTO EM ESTABELECIMENTOS QUE NECESSITAM DA PRESENÇA DE UM ATENDENTE

Monografia de Trabalho de Conclusão de Curso apresentada ao colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Eixo de formação: Eletrônica e Computação.
Orientador: Prof. Dr. Sandro Trindade Mordente Gonçalves

Divinópolis-MG
2014

Centro Federal de Educação Tecnológica de Minas Gerais
CEFET-MG / Campus Divinópolis
Curso de Engenharia Mecatrônica

Monografia intitulada "*Desenvolvimento de um dispositivo de baixo custo visando otimizar o atendimento em estabelecimentos que necessitam da presença de um atendente"*, de autoria do graduando Apolo Malta, aprovada pela banca examinadora constituída pelos seguintes professores:

_____

Prof. Dr. Sandro Trindade Mordente Gonçalves - CEFET-MG / Campus Divinópolis - Orientador

_____

Prof. Dr. Renato de Sousa Dâmaso - CEFET-MG / Campus Divinópolis

_____

Prof. MSc. Alberto Pena Lara - CEFET-MG / Campus Divinópolis

_____

Prof. Dr. Valter Junior de Souza Leite
Coordenador do Curso de Engenharia Mecatrônica
CEFET-MG / Campus Divinópolis

Divinópolis - Fevereiro de 2014

Agradecimentos

Agradeço primeiramente a Deus pela oportunidade de aprimorar os conhecimentos;

Agradeço aos meus pais, pela herança espiritual, pelos ensinamentos de vida, pelos estímulos diários e confiança;

Agradeço aos meus amigos de sala pelo tempo que passamos juntos, principalmente àqueles que estiveram sempre presente nos momentos de aprendizagem;

Agradeço especialmente, ao professor Sandro pela orientação e por todo o auxílio durante esse percurso;

Agradeço a todos meus professores pelos ensinamentos passados, principalmente ao professor Alan Marotta que me deu total suporte durante o desenvolvimento deste projeto e que durante essa etapa se tornou um grande amigo;

Enfim, agradeço a todos que de alguma forma contribuíram para minha formação.

Resumo

Este trabalho consiste em desenvolver um dispositivo, de baixo custo, para auxiliar o atendimento em estabelecimentos do tipo bares e restaurantes. O cliente poderá realizar os pedidos no local em que se encontra no estabelecimento dispensando o auxílio de um atendente. O dispositivo não possui fios, o que permite uma maior mobilidade e capacidade de alocação do mesmo de acordo com a acomodação do cliente. Para realizar a comunicação entre o dispositivo e a central não se utiliza nenhum protocolo de comunicação já existente, ou seja, utiliza-se um protocolo de comunicação próprio, totalmente desenvolvido durante o projeto. Para realizar a transmissão dos dados é utilizado um módulo transmissor e receptor de comunicação.

Palavras-chave: automação de bares e restaurantes, microcontrolador, dispositivo de baixo custo.

Abstract

This work is to develop a device, low cost, to assist in the care of type establishments bars and restaurants. The customer can place your orders at the place where the property is dispensing the aid of an attendant. The device has no wires, which allows for greater mobility and ability to allocate the same according to accommodate the customer. To perform communication between the central device

and does not use any existing communication protocol, or a protocol used for communication own, fully developed during the design. To perform the data transmission a transmitter and receiver communication module is used.

Key-words: Automation of bars and restaurants, microcontroller, low cost device.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos e Notação

V: Volt, unidade de tensão elétrica
A: Ampère, unidade de corrente elétrica
mm: Milímetro, unidade de comprimento
Hz: Hertz, unidade de frequência
s: Segundo, unidade de tempo
TBJ: Transistor Bipolar de Junção
PIC: Programmable Interface Controller
ICD: In Circuit Debugger
LSB: Least Significant Bit
WDT: Watchdog Time

Capítulo 1

Introdução

1.1. Relevância

Atualmente, há empresas que desenvolvem produtos que tem como principal objetivo a automatização do atendimento de bares e restaurantes. Há produtos que oferecem soluções simples como o "*Smart Call*" (Mafra, 2013). Outros, apresentam soluções mais sofisticadas como os aplicativos para *smartphones* e *tablets* que visam um maior conforto e praticidade aos clientes em relação ao atendimento nos estabelecimentos.

A empresa Nextdata possui um aplicativo de automação de comandas, chamado Nextdata Restaurante. Este aplicativo é compatível com *palms*, *pockets* e *smartphones*. Atualmente é utilizado em bares, restaurantes, boates e pubs. Sem esta ferramenta um garçom atende uma média de 4 a 5 mesas por hora. Utilizando o Nextdata Restaurante, esse número sobe para 8 a 12 mesas por hora. Sendo assim, o aproveitamento do profissional pode ser otimizado em mais de 100%. O dispositivo integra o garçom com o caixa e com a cozinha, estabelecendo uma comunicação colaborativa entre ele via Wi-Fi (Nextdata, 2013).

A empresa Runze Tecnologia também possui um aplicativo desenvolvido para a automação de bares e restaurantes, JBarCliente . Para este aplicativo ser utilizado o cliente deve possuir um *smartphones* ou *tablet* com compatibilidade com o sistema operacional. O JBarCliente é instalado no dispositivo do cliente, permitindo-o controlar a conta da mesa, fazer pedidos e obter o extrato da conta em tempo real através de Bluetooth (Runze Tecnologia, 2013).

Já a empresa Mafra possui um sistema mais simples, porém de funcionalidade semelhante do que os sistemas que envolvem aplicativos para *smartphones* e similares. Em bares e restaurantes o sistema funciona da seguinte forma, um pequeno dispositivo eletrônico sem fio é instalado em todas as mesas do estabelecimento. Quando um cliente deseja chamar o garçom basta que ele pressione um botão e o número de sua mesa será enviado por ondas de rádio para um painel eletrônico que registra a chamada e sinaliza, via um aviso sonoro e a apresentação do número que chamou em seus displays que uma nova solicitação de atendimento foi feita (Mafra, 2013).

As opções que utilizam o sistema de aplicativos são mais completas. Em contrapartida, são de elevado custo de implementação. Primeiramente, o cliente deverá já possuir o dispositivo (*smartphones* ou *tablet*) para utilizar o sistema, caso contrário o dono do estabelecimento há de arcar com a compra de dispositivos caso ele desejar que todos os clientes tenham acesso a este tipo de atendimento. Além

do mais, o estabelecimento deverá contar com um serviço de internet, já que o sistema utiliza conectividade por Wi-Fi para enviar e receber informações.

Os outros sistemas já apresentam custos reduzidos ao serem implantados porém, eles são limitados. Estes não permitem que o cliente efetue o seu pedido diretamente do local onde se encontra no estabelecimento. Sempre haverá a necessidade de chamar o atendente quando precisar de algum serviço. Este tipo de sistema não permite uma grande liberdade ao cliente limitando a versatilidade do produto.

1.2. Motivação

O dispositivo pensado já teria a vantagem que o cliente poderia realizar o seu pedido diretamente da mesa em que se encontra e o pedido seria recebido pelo computador do estabelecimento, sem a necessidade de um atendente. Além disso, o proprietário do estabelecimento tem o auxílio de uma interface gráfica, com a qual pode controlar todos os pedidos realizados que são separados por mesa e pelo valor que cada mesa está consumindo no momento. Pode também realizar o fechamento de conta diretamente pela interface com possibilidade de impressão do extrato e no final de cada dia de funcionamento o proprietário tem acesso ao faturamento total diário, com uma lista de todos os itens que foram consumidos e o valor total arrecadado, o que facilita bastante o controle de estoque de mercadoria do estabelecimento.

O dispositivo irá realizar a comunicação com um computador utilizando uma banda livre de transferência de dados, não havendo a necessidade da utilização de fios e não acrescentando custos com uma rede Wi-Fi. Nenhum protocolo comercial já disponível no mercado será utilizado para realizar a comunicação entre o dispositivo e o computador. Com a utilização de microcontroladores, consegue-se criar um conjunto de regras para a transmissão ordenada e automática de dados. Isso permite que a central receptora de dados coordene e controle as operações de transferência de dados. Dessa maneira, todo o protocolo de comunicação será desenvolvido durante o projeto. O desenvolvimento do protocolo de comunicação é economicamente viável por utilizar microcontroladores, que são dispositivos de custo acessível, o que torna ainda mais viável o desenvolvimento do projeto. Desse modo, o objeto de pesquisa oferece vantagens em relação aos sistemas que utilizam aplicativos para *smartphones* e *tablets*.

Ao desenvolver o dispositivo, visando um custo reduzido de implementação em relação a outros sistemas já existentes que desempenham funções equivalentes, este será um instrumento que poderá atuar em diferentes áreas comerciais como: entretenimento (bares, restaurantes), industrial, área da saúde.

1.2.1. Bares, lanchonetes e restaurantes:

Os clientes poderão solicitar seus pedidos diretamente da mesa em que se encontram. Com isso garante rapidez no atendimento, conforto ao cliente e aumento nas vendas. Gera também a possibilidade do garçom realizar outros trabalhos simultâneos, já que o mesmo não precisará ficar observando o salão a todo o momento.

1.2.2. Clínicas de repouso ou asilos:

Os moradores poderão solicitar o atendimento aos enfermeiros e auxiliares de um modo mais fácil principalmente para as pessoas com limitações ou com dificuldades de locomoção e movimentos restritos.

1.2.3. Clínicas médicas, odontológicas, ou qualquer estabelecimento relacionado à área de saúde:

Durante o atendimento ao paciente, o profissional da área de saúde geralmente necessita de algum instrumento ou chamar um auxiliar para dar continuidade ao processo em questão. Nesta situação, o uso de um aparelho telefônico ou um sistema de mensagem via computador pode ser inviável devido à pressa ou por questões de higiene. A utilização do dispositivo garante agilidade ao solicitar a presença de auxiliares e evita que o profissional entre em contato com equipamentos que podem estar contaminados.

1.2.4. Setor de produção das indústrias:

Quando há uma linha de montagem, nas indústrias, os trabalhadores são distribuídos por setores, mesas ou estações de trabalho e geralmente os funcionários são supervisionados por controladores de qualidade responsáveis por administrar e gerenciar a produção. Com o auxílio do dispositivo, o funcionário pode solicitar a presença do supervisor, reposição de material, entre outros, quando necessário. Basta distribuir os dispositivos nos setores de produção e quando for requerido algum dessas necessidades, basta acionar o dispositivo que aparecerá no computador a solicitação do que foi pedido.

1.3. Objetivos

O objetivo deste trabalho é desenvolver um dispositivo, de baixo custo e que não utilize fios, que auxilie o atendimento em estabelecimentos do tipo bar e restaurante de modo que o cliente pode realizar os pedidos do local onde ele se encontra no estabelecimento.

1.4. Metodologia

O desenvolvimento do projeto foi inicialmente idealizado em etapas, descritas a seguir:

- Revisão bibliográfica;
- Pesquisa e verificação dos produtos já existentes no mercado e encontrar uma lacuna existente no mercado;
- Simulações em computador dos circuitos eletrônicos relacionados ao microcontrolador;
- Projeto e desenvolvimento do dispositivo eletrônico;
- Envio e recepção de dados pela porta serial e utilizando o módulo de comunicação;
- Projeto e desenvolvimento da interface gráfica;
- Análise do funcionamento do sistema e realização de ajustes;
- Testes finais.

1.5. Organização deste trabalho

O trabalho é dividido em seis capítulos. Neste, são apresentados os objetivos do trabalho proposto e a motivação para desenvolvê-lo. São apresentados ainda: a metodologia adotada, o escopo do trabalho e a caracterização do dispositivo desenvolvido e suas vantagens em relação aos produtos já existentes no mercado. No Capítulo 2 é apresentada uma fundamentação teórica dos assuntos relevantes com o objetivo de agregar uma maior quantidade de informações e facilitar o entendimento do trabalho. O Capítulo 3 traz detalhadamente como foi o desenvolvimento eletrônico do dispositivo bem como o detalhamento da programação realizada no microcontrolador. No Capítulo 4 é apresentado todo o desenvolvimento da interface gráfica e como ela está divida de planejada e os testes e validações são descritos e analisados no Capítulo 5. Finalmente, o Capítulo 6 trata das análises dos resultados finais, conclusões do trabalho e proposições para futuros estudos.

Capítulo 2

Fundamentação teórica

2.1. Microcontrolador PIC 16F877A

O microcontrolador utilizado para o desenvolvimento do projeto pertence a família PIC 16F e é fabricado pela Microchip.

O PIC16F877A é um microcontrolador que possui 40 pinos, sendo que 33 deles podem ser configurados como entrada ou saída. Ele possui um canal para comunicação serial RS232, um canal para comunicação I2C, 8 canais conversores analógico/digital AD, 2 canais PWM, dois Timers de 8 bits e um de 16 bits, além de 14,3K de memória de programa, 368 bytes de RAM e 256 bytes de EEPROM.

2.1.1. Ciclo de Máquina

Em um microcontrolador, todas as atividades necessitam de sincronização para que ocorra um correto funcionamento do sistema. O *clock* realiza justamente essa função, ou seja, atua como de sinal de sincronização. O *clock* é um pulso alternado de sinais de tensão, gerado pelos circuitos de relógio (composto de um cristal oscilador e circuitos auxiliares). Quando os dispositivos do microcontrolador recebem o sinal de executar suas atividades, dá-se a esse acontecimento o nome de "pulso de *clock*". Em cada pulso os dispositivos executam suas tarefas, param e vão para o próximo ciclo de *clock*.

Nos microcontroladores PIC, o sinal do *clock* é internamente dividido por quatro. Portanto, para um *clock* externo de 4 MHz, temos um *clock* interno de 1 MHz e desta forma, cada ciclo de máquina dura 1 µs. A divisão do *clock* por quatro forma as fases Q1, Q2, Q3 e Q4. O *program counter* é incrementado automaticamente na fase Q1 do ciclo de máquina e a instrução seguinte é buscada da memória de programa e armazenada no registrador de instruções no ciclo Q4. Ela é decodificada e executada no próximo ciclo, no intervalo de Q1 até Q4. Essa característica de buscar a informação num ciclo de máquina e executá-la no próximo ciclo é conhecida como PIPELINE. Ela permite que quase todas as instruções sejam executadas em apenas um ciclo, gastando assim 1 µs (para um *clock* de 4 MHz) e tornando o sistema muito mais rápido. As únicas exceções referem-se às instruções que geram “saltos” no *program counter*, como chamados de rotinas e retornos. Ao executar essas instruções, o PIPELINE deve ser primeiramente limpo para depois poder ser carregado novamente com o endereço correto, consumindo para isso dois ciclos de máquina. Esse PIPELINE é facilmente implementado devido à arquitetura Havard.

O diagrama da figura 2.1 representa o esquema do ciclo de máquina e demonstra claramente as divisões do ciclo nas quatro fases (Q1 a Q4) e o conceito de PIPELINE (SOUZA, 2013, p. 24).

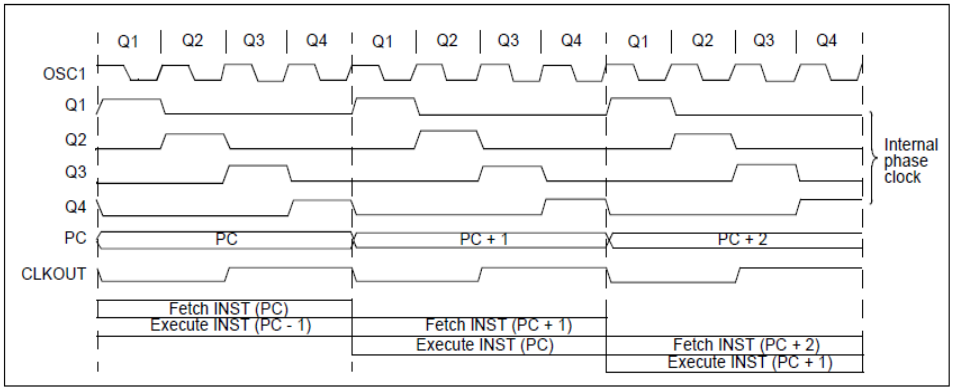

Figura 2.1 Esquema dos ciclos de máquina

2.1.2. USART

Em muitas aplicações microcontroladas, pode ser necessário realizar a comunicação entre o microcontrolador e um ou mais dispositivos externos. O microcontrolador utilizado possui um protocolo de comunicação, a USART.

O nome USART significa Universal *Synchronous Asynchronous Receiver Transmitter*. Esse é um protocolo universal e possui dois modos distintos de trabalho: o modo síncrono e o modo assíncrono. Como o dispositivo desenvolvido não utiliza o *clock* para o sincronismo da transmissão e recepção de dados, já que não há como determinar o momento quando o usuário irá utilizar o dispositivo, aplica-se o modo assíncrono para efetuar a comunicação entre o dispositivo e o meio externo.

Nos protocolos assíncronos, não se encontra uma linha específica para sincronização. Neste caso, a sincronização entre os elementos transmissor e receptor é garantida pela precisão dos *clocks* de cada elemento e também pela utilização de sinais marcadores de inicio (*start*) e fim (*stop*) da palavra transmitida (PEREIRA, 2009, p.263). A figura 2.2 representa a transmissão de um sinal serial assíncrono:

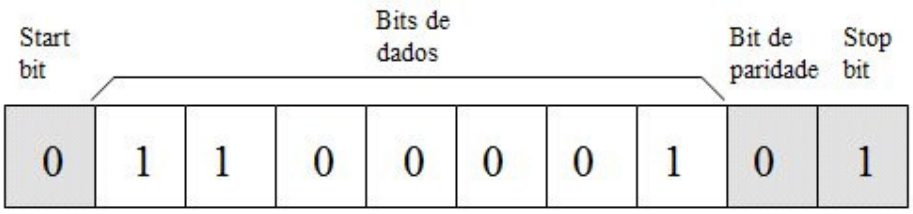

Figura 2.2 Transmissão de um sinal serial assíncrono

Observa-se que a transmissão começa por um sinal de ínico (start), seguido pelos bits de dados (no caso 8 bits), iniciando pelo bit LSB e após eles, um bit de parada (stop) para sinalizar o fim do caractere (PEREIRA, 2009, p.263). Ou seja, o sincronismo é efetuado pelo próprio conjunto dado, isto é, os sistemas conseguem determinar quando cada conjunto de dados começa, através da precisão no tamanho de cada bit transmitido (baud rate) (SOUZA, 2013, p.171).

O Baud Rate é a velocidade com que os dados trafegam pela comunicação. Essa velocidade define o tamanho de cada bit, e por isso foi padronizada a utilização da unidade BPS (bits por segundo). Os valores de Baud Rate também foram padronizados, por exemplo, em 2400, 4800, 9600 etc (SOUZA, 2013, p.171). Após estabelecer uma velocidade de comunicação para o sistema, tanto o transmissor quanto o receptor utilizarão a mesma velocidade.

Para calcular o tempo de cada bit trafegando nas linhas TX e RX utiliza-se a fórmula (2.1):

$$T_{BIT} = \frac{1}{Baud\ Rate} \tag{2.1}$$

Ao conhecer o T_{BIT} é necessário saber quando começa o primeiro bit, o Start Bit. O padrão para ambas as linhas de comunicação é nível lógico alto (5 Vcc). Quando a comunicação de um byte começa, a linha vai para nível lógico baixo (GND), permanecendo desta forma pelo tempo de 1 bit (T_{BIT}). Esse intervalo específico em zero (0) é chamado de Start Bit e serve para sincronizar a recepção do dado. Depois disso, 8 bits serão transmitidos, respeitando a relação lógica de tensão (0 = GND e 1 = 5V). No final, obrigatoriamente deve voltar ao nível alto, para que o próximo Start Bit seja reconhecido (SOUZA, 2013, p.172).

Ao final da transmissão, pode ser definido também que o sistema opte por um intervalo em nível alto, para ajudar a separar um dado do outro. Esse intervalo é chamado de Stop Bit (SOUZA, 2013, p.172).

Por último, existe ainda a possibilidade da transmissão de um 9º bit (entre o final do dado e o Stop Bit), que serve para checar a integridade da informação recebida. Esse bit é chamado de paridade, e pode ser configurado como ímpar ou par. A paridade em uma comunicação é opcional.

2.1.3. Interrupções

As interrupções servem para interromper o programa imediatamente. Desta maneira, podemos tomar atitudes instantâneas. As interrupções são ações tratadas diretamente pelo hardware, o que as torna muito rápidas e disponíveis em qualquer ponto do sistema. Assim sendo, quando uma interrupção acontece, o programa é paralisado, uma função específica (definida pelo programador) é executada e depois o programa continua a ser executado no mesmo ponto em que estava (SOUZA, 2013, p.31).

2.1.3.1 Timer 0

O Timer 0 é um contador de 8 bits que pode ser acessado diretamente na memória, tanto para leitura quanto para a escrita. A diferença entre ele e os demais registradores é que seu incremento é automático e pode ser feito pelo clock da máquina ou por um sinal externo. O estouro desse contador pode gerar uma interrupção (SOUZA, 2013, p.43).

A interrupção do Timer 0 ocorre sempre que o módulo contador ou temporizador timer 0 estoura a sua contagem, ou seja, na transição de 255 (8 bits em “1”) para 0 (ou 8 bits em “0”) do timer 0 (PEREIRA, 2012, p.52).

2.1.3.2 Interrupções de USART

Esse sistema, além de facilitar todo o processamento para entrada e saída de dados seriais, possui duas interrupções para informar o programa quando um dado foi recebido e quando a transmissão de outro dado já foi terminada (SOUZA, p.33, 2013).

2.1.4 Modo Sleep

O dispositivo desenvolvido deve possuir mobilidade de alocação, por isso utilizam-se pilhas e baterias para sua alimentação. No entanto, é desejável que não haja o desperdício desta fonte de energia. O microcontrolador presente no projeto possui um recurso exclusivo para dispositivos com esse tipo de alimentação, o modo *sleep*.

Os microcontroladores possuem um modo de operação exclusivo para economia de energia, o modo *sleep*. Este modo de operação é utilizado em sistemas que podem ficar paralisados temporariamente, principalmente quando é alimentado por pilhas ou baterias (SOUZA, 2013, p. 163). O microcontrolador possui um *driver* interno para tratar o sinal que recebe do oscilador. Quando entra em modo *sleep*, este *driver* é desligado, deste modo, o sinal do oscilador chega ao pino de entrada do PIC, mas não chega à CPU e o processamento é paralisado. O WDT não é paralisado (SOUZA e LAVINIA, 2010, p. 332) porque este timer possui um oscilador independente que não é afetado pela função ‘SLEEP’. As portas de E/S mantêm o estado em que se encontravam antes da execução da instrução SLEEP (MICROCHIP, 2007, p. 112), ou seja, as portas que estavam configuradas como entrada continuam como entrada e as portas configuradas como saída mantêm seu nível lógico (SOUZA, 2013, p. 163).

O consumo de energia do PIC, que pode chegar a 300 mA no modo de operação normal (MICROCHIP, 2003, p. 117), pode cair para até 1 µA em modo *sleep* (SOUZA, 2013, p. 163). Para garantir o mínimo consumo de energia em modo SLEEP é recomendável configurar todas as portas de E/S como entrada quando isso for possível (SOUZA, 2013, p. 163).

O microcontrolador pode ser “acordado’’ de três formas diferentes (SOUZA e LAVINIA, 2006, p. 332):

- Um reset externo (no pino 1 / MCRL);
- Estouro no Watchdog Timer;
- Interrupção externa ou mudança do estado dos pinos RB0, RB4, RB5, RB6 ou RB7.

Na configuração da figura 2.3 utiliza-se o primeiro caso, um reset de hardware, que possui um mesmo botão para RESET/WAKE UP que irá reinicializar o sistema independentemente de estar no modo *sleep* ou não.

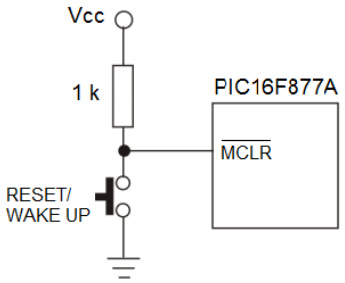

Figura 2.3 Circuito do Master Clear Reset

2.1.5 Teclado Matricial:

No projeto, é necessário que haja uma interação entre o objeto de estudo e o usuário. Para essa interação deve-se utilizar um dispositivo que seja capaz de identificar os dados de entrada do usuário para que o microcontrolador processe estes dados e possa enviá-los ao computador da central. Outros dispositivos normalmente utilizados para a interação entre o mundo exterior e os microcontroladores são as teclas que podem ser utilizadas para uma infinidade de aplicações: início e fim de processos, programação, alteração de parâmetros, sensoriamento, etc (PEREIRA, 2012, p. 300).

As técnicas para a interface entre as teclas e o microcontrolador variam de acordo com diversos parâmetros, mas principalmente em função do número de teclas. No caso do projeto, será utilizado um teclado com dezesseis teclas. Em teclados com um número de teclas superior a oito, utiliza-se normalmente a técnica de varredura de matriz. Nessa técnica as teclas componentes do teclado são arranjadas para formar uma matriz com linhas e colunas (PEREIRA, 2012, p. 302).

De forma a economizar pinos de entrada e saída, pode-se agrupar as teclas em uma estrutura de matriz, que é lida por meio de um sistema de varredura, de forma a verificar cada coluna separadamente, uma após outra. O teclado projetado possui doze teclas, totalizando a necessidade de utilizar doze pinos do microcontrolador. Ao utilizar a técnica de varredura, podem-se agrupar as teclas em uma matriz de quatro linhas e quatro colunas. De forma que se consegue reduzir a quantidade de dezesseis pinos, necessários para a conexão do teclado no microcontrolador, para sete pinos. Ao utilizar a técnica de varredura de teclado, (PEREIRA, 2006), consegue-se reduzir substancialmente a quantidade de pinos de entrada e saída que são necessárias para efetuar a conexão do teclado.

O teclado matricial é um periférico de entrada de dados. Os botões, normalmente abertos, são organizados em linhas e colunas. Conforme uma tecla é pressionada, um curto-circuito é estabelecido entre o pino referente à sua linha e sua coluna.

2.1.6 Varredura de Teclado:

Durante a varredura de teclado, as linhas apresentam nível lógico “1” (VCC). Este “1” lógico é forçado através de um resistor de *pull-up*. O circuito de verificação deve então executar um loop inserido o nível lógico “0” (GND) na coluna que deverá ser lida e manter o estado “1” lógico nas demais colunas. No momento que o usuário pressionar uma tecla na coluna onde existe o estado “0” lógico, imediatamente a linha que faz referência à tecla pressionada assumirá o mesmo estado, no caso “0” lógico.

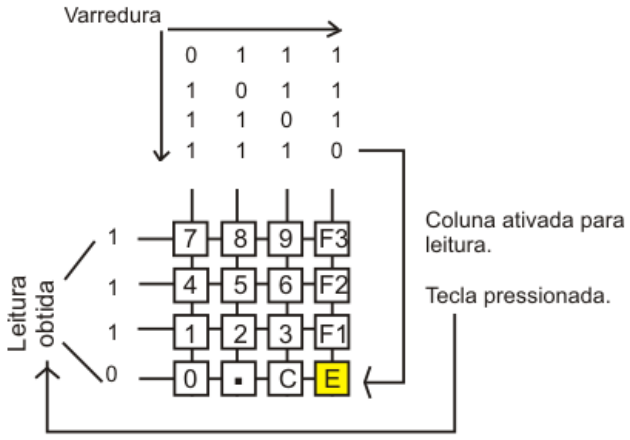

Figura 2.4 Varredura de teclado

A varredura insere nas colunas o estado lógico com uma velocidade muito superior ao que uma pessoa pressiona uma tecla. Primeiramente, insere-se o estado lógico “0” na primeira coluna, depois na segunda, e assim sucessivamente até chegar à última. Caso nenhuma tecla for pressionada, é iniciada uma nova varredura na busca de uma tecla. Assim o circuito de verificação ficará em um *loop* infinito aguardando uma tecla ser pressionada.

2.1.7 Circuito Elétrico

O circuito do teclado matricial é mostrado na Figura 2.5. Este circuito é bastante simples e trás a síntese do que foi explicado anteriormente: uma matriz com três colunas e quatro linhas com 12 teclas fazendo a interligação das mesmas quando pressionadas.

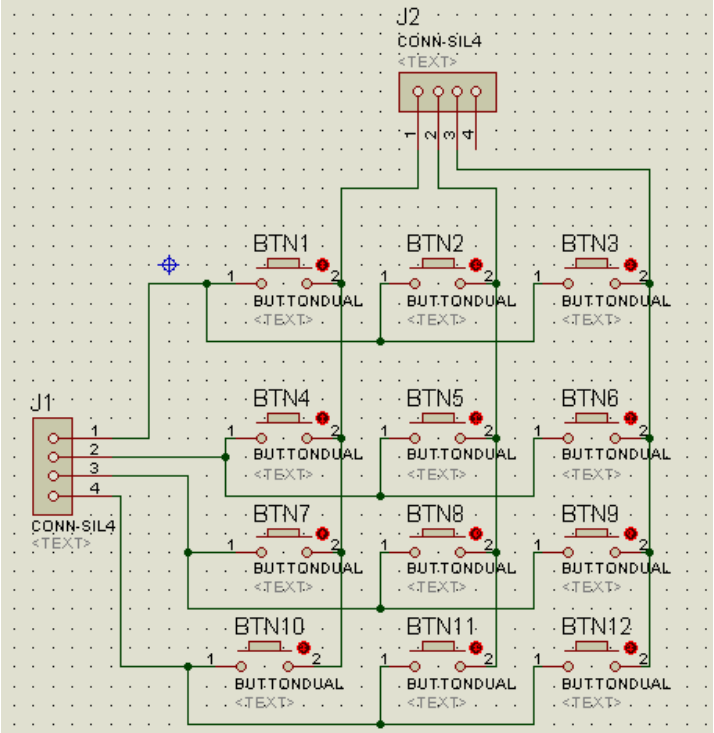

Figura 2.5 Circuito do Teclado 3x4

A figura 2.6 mostra o mesmo circuito, porém no modo PCB.

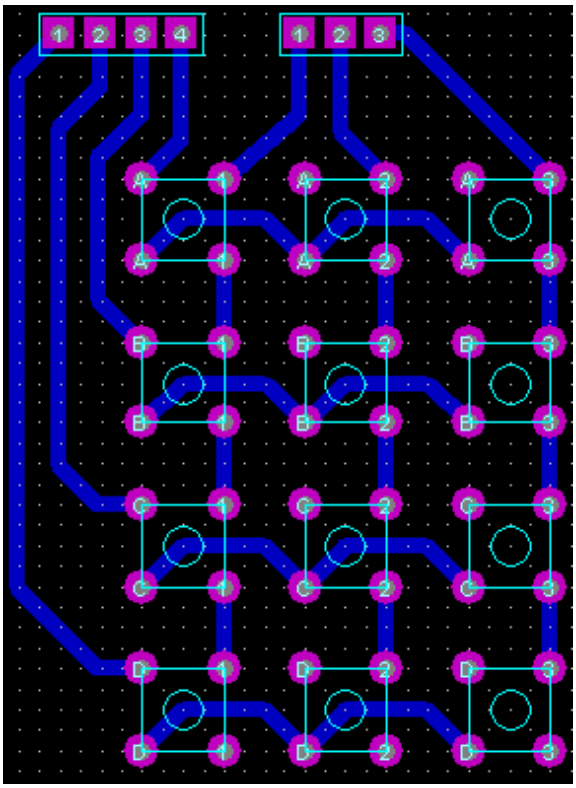

Figura 2.6 Circuito PCB do teclado 3x4

2.1.8 Display LCD:

Conforme visto na seção 2.1.5, o teclado é um modo muito utilizado de interação do usuário com o dispositivo, no entanto eles são unidirecionais e somente permitem o fluxo de informações flua no sentido do usuário para o dispositivo, mas não ao contrário. Para que a informação seja dirigida do dispositivo para o usuário, é necessária outra forma de interação, como os displays, por exemplo.

Os displays LCD permitem uma grande concentração de informações em um pequeno espaço físico. Além disso, apresentam um baixo consumo de energia (da ordem de 1 mA, quando em plena atividade e sem iluminação de fundo) (PEREIRA, 2012, p.308).

Ao utilizar um Display LCD, deve-se reservar apenas 7 pinos de controle. Este número é o mesmo para um display 16x2 (32 caracteres alfa numéricos) ou um display 20x4 (80 caracteres alfa numéricos). Ao realizar a comparação com um Display de 7 Segmentos, o qual utiliza 8 pinos de controle, observa-se uma grande vantagem na utilização do Display LCD quando se trata de concentração de informações. De modo que há a possibilidade de mostrar 32 caracteres por vez contra apena 1 do Display de 7 Segmentos.

O display LCD utilizado é do modelo 16x2 e apresenta 16 pinos de acesso, figura 2.7:

- 8 pinos de dados (D0 a D7), figura 2.8;
- pinos de controle (ENABLE, RS e RW), figura 2.9;
- 2 pinos para iluminação "*back-light*".

Figura 2.7 LDC visão geral

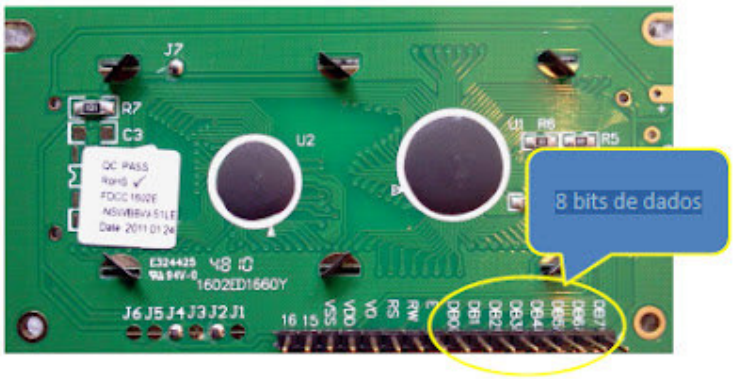

Figura 2.8 LDC pinos de dados

Figura 2.9 LDC pinos de controle

A comunicação no modo de 4 bits é realizada utilizando apenas as quatro linhas mais significativas de dados (D7 a D4), dividindo o byte em dois nibbles que são transferidos sempre iniciando pelo mais significativo seguido pelo menos significativo (PEREIRA, 2012, p.308).

A alimentação padrão é de 5V (de 4,5 à 6V), os pinos (1) Vss(GND) e (2) Vdd (Vcc) podem ser ligados juntamente com o PIC à mesma alimentação.

2.1.9 Ajuste do Contraste:

O pino (3) controla o contraste do LCD, para isso é necessário um resistor de um determinado valor para realizar a ligação entre este pino e o GND. Para determinar este resistor utiliza-se um potenciômetro entre 10 kΩ e 20 kΩ para este ajuste conforme a figura 2.10:

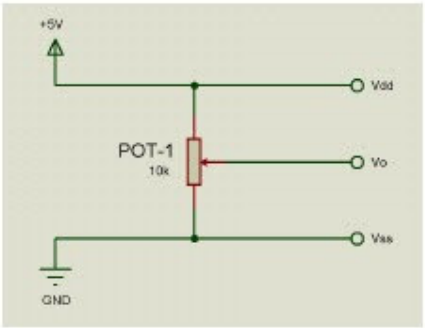

Fig 2.10 Ajuste do resistor de contraste

O valor encontrado neste teste foi de 983 Ω, logo será utilizado um resistor de 1 kΩ já que é o valor mais próximo encontrado para os resistores comercializados.

2.1.10 Módulo JY-MCU:

O JY-MCU, está mostrado na figura 2.11, é um módulo transmissor/receptor de porta serial sem fio Bluetooth, substituto para conexões seriais físicas, com um alcance de 25 metros sem obstáculos. Possui comunicação transparente de transmissão e recepção, ou seja, todos os dados enviados pelo transmissor do módulo são recebidos de maneira idêntica no destino de envio.

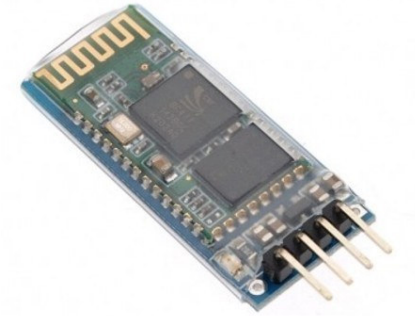

Figura 2.11 Módulo JY-MCU

É importante ressaltar que cada módulo efetua o pareamento uma porta COM diferente do sistema, não há possibilidade de dois módulos parearem com a mesma COM, como mostrado na figura 2.12:

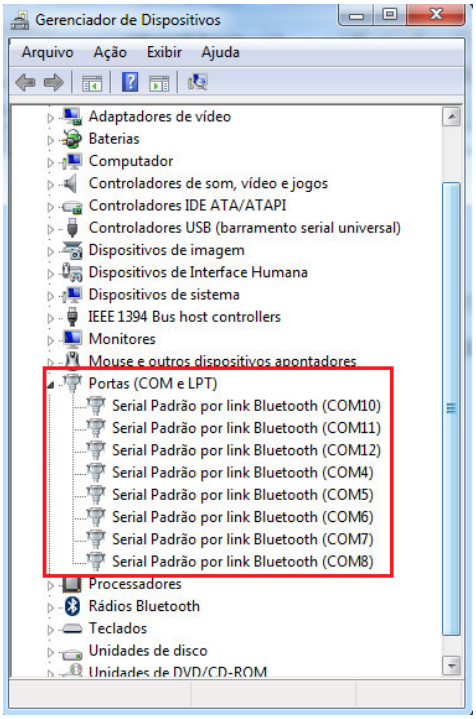

Fig 2.12 Pareamento dos módulos com as portas COM do computador

Isso é de extrema importância, pois assim os sinais enviados por um módulo não gera interferência no demais por usarem portas distintas de acesso.

2.1.11 Transistor de Junção Bipolar (TBJ), operação como chave:

Pode parecer uma visão muito limitada de um componente tão versátil, mas a verdade é que não é o objetivo deste trabalho um estudo teórico de eletrônica, e é difícil encontrar uma outra função para este componente que não seja melhor executada por um circuito integrado com preço acessível. O TBJ será utilizado como uma chave, nos modos de operação no corte e na saturação (SEDRA, 2007, p.261), por ser simples e barato.

Capítulo 3

Desenvolvimento do dispositivo eletrônico

3.1. Escolha do microcontrolador

O projeto do dispositivo iniciou-se a partir das definições das funções que seriam utilizadas no microcontrolador, que foram:

- Teclado Matricial, para realizar a interação usuário/dispositivo;
- Display LCD, para realizar a interação dispositivo/usuário;
- Modo de comunicação serial, USART;
- Modo Sleep, para a economia de energia;
- Interrupção Timer 0, no modo contador;
- Master Clear Reset, para gerar WAKE-UP no microcontrolador.

Ao selecionar todas as funções que seriam necessárias para a implementação do dispositivo, concluiu-se que o microcontrolador PIC16F628A, apesar de possuir todas essas características, não poderia ser utilizado, por não apresentar uma quantidade de pinos (um total de 18) suficiente. Então, foi utilizado o PIC16F877A, pois apresenta um total 40 pinos, além de possuir as características requeridas.

3.2. Desenvolvimento do circuito

Para a montagem e simulação do circuito foi utilizado o Proteus Design Suite, da empresa Labcenter Electronics, em conjunto com o CCS C Compiler, para realizar a programação do microcontrolador. Na figura 3.1 é apresentado o circuito projetado:

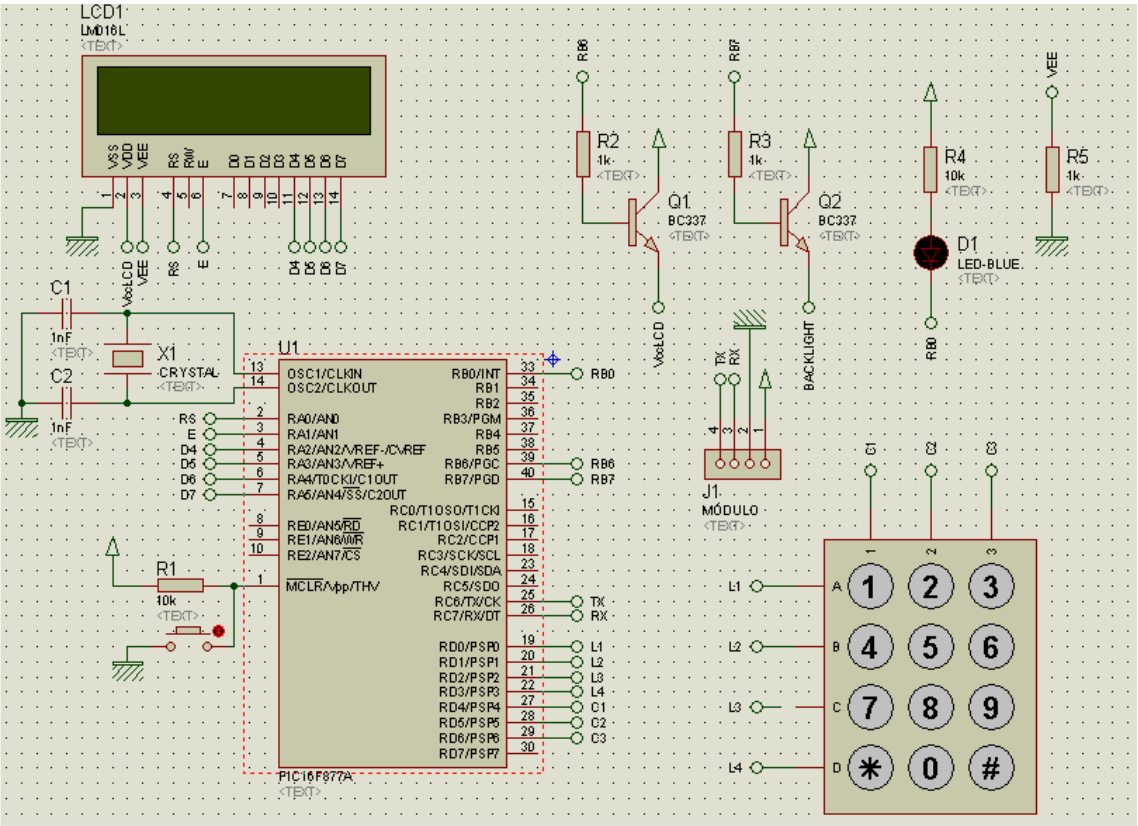

Figura 3.1 Circuito do dispositivo eletrônico

O circuito apresenta um Teclado Matricial 3x4, que são conectados no PORTD (lê-se port D) do microcontrolador. Nesta porta foram habilitados os resistores de *pull-up* internos, de modo que dispensa a utilização de resistores externos que são necessários para capturar os dados corretamente durante a varredura de teclado.

O *Display LCD* está conectado no PORTA (lê-se port A) do microcontrolador. Utilizam-se: os pinos de dados do LCD (D4 ao D7); o pino E (*Enable*), o que qual habilita a utilização do *LCD*; o pino RS, o qual permite a função de leitura dos dados do microcontrolador. Tanto as alimentações do *display*, quanto a do *back-light*, estão conectadas aos pinos emissores dos transistores BC 337. Os transistores estão operando no modo de chaveamento e durante o funcionamento do dispositivo os pinos RB6 e RB7 do microcontrolador ativa o nível lógico “1”, o que faz com que tanto o display quanto o *back-light* sejam alimentados. A utilização dos transistores é de extrema importância para que se consiga realizar o desligamento do display e do *back-light* quando o dispositivo entrar no Modo *Sleep*, pois assim associa-se o funcionamento do *Display LCD* com o funcionamento do microcontrolador.

O Módulo JY-MCU está conectado nos pinos da USART do microcontrolador, RC6 e RC7, os quais são responsáveis pela comunicação serial. É importante ressaltar que o pino TX do microcontrolador é ligado ao pino RX do módulo e vice-versa.

No pino 1 do microcontrolador está conectado um botão que realiza a função de WAKE-UP do dispositivo.

O LED no pino RB0 é utilizado para mostrar que o programa do microcontrolador foi inicializado e também sinaliza que o dispositivo está aguardando a chegada de um sinal externo pela porta RC7 (pino RX).

3.3. Funcionamento do programa

Todo o código do programa foi desenvolvido utilizando o CCS C Compiler, em linguagem C. Inicialmente, o programa inicializa todos os parâmetros necessários para a utilização dos recursos já citados em tópicos anteriores e atribui o valor zero para a variável de controle “n” e segue para a sub-rotina “A”. Essa variável é responsável pela seleção da qual sub-rotina que o programa irá acessar. A figura 3.2 mostra esse primeiro percurso que o programa segue:

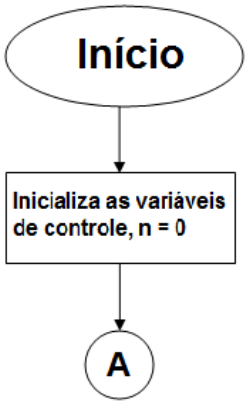

Figura 3.2 Fluxograma Ínicio

A sub-rotina “A” é o laço principal do programa e todas as outras sub-rotinas retornam à ela. Em “A” há uma condição que verifica o valor de “n” e dependendo do valor, o programa segue uma sub-rotina específica. A figura 3.3 mostra o fluxograma da sub-rotina “A”:

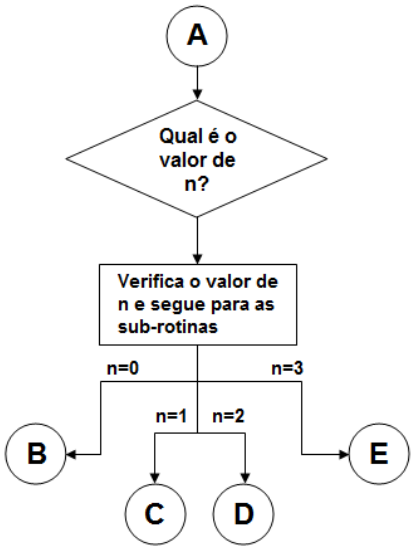

Figura 3.3 Fluxograma sub-rotina “A”

O valor da variável de controle é inicializado com zero, então a sub-rotina “B” é a acessada logo em seguida. Em “B” é realizado a Varredura de Teclas, aguardando o momento em que uma tecla é pressionada. Quando uma tecla é pressionada, uma variável auxiliar (aux1) recebe o valor referente a esta tecla e “n” recebe “1”, (n=1), e o programa retona para a sub-rotina “A”. A figura 3.4 mostra o fluxograma da sub-rotina “B”:

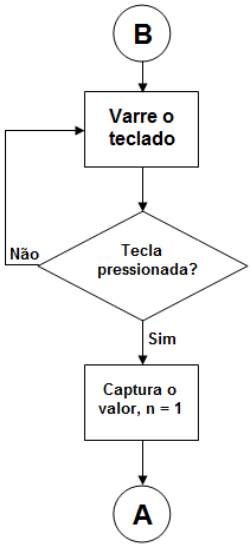

Figura 3.4 Fluxograma sub-rotina “B”

A sub-rotina “A” verifica o valor de “n” e chama a sub-rotina “C”. Em “C” repete-se o mesmo procedimento realizado em “B”. Porém, armazena o valor da tecla pressionada em uma variável auxiliar diferente (aux2) e “n” recebe “2”, (n=2), e o programa retona para a sub-rotina “A”. A figura 3.5 mostra o fluxograma da sub-rotina “C”:

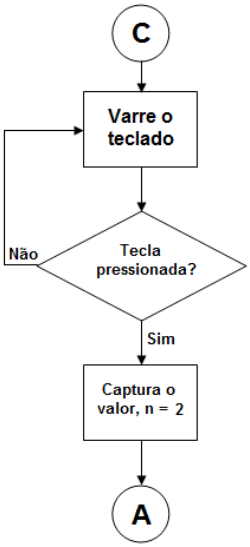

Figura 3.5 Fluxograma sub-rotina “C”

A sub-rotina “A” verifica o valor de “n” e chama a sub-rotina “D”. Em “D” o programa aguarda o acionamento de uma das seguintes teclas: “Envia” ou “Corrige”. Se a tecla “Corrige” for pressionada, a variável de controle recebe e as variáveis auxiliares recebem “0” (n=0, aux1=0, aux2=0) e o programa retorna para a sub-rotina “A”. Caso a tecla “Envia” for pressionada, “n” recebe “3” e o programa também retorna para a sub-rotina “A”. A figura 3.6 mostra o fluxograma da sub-rotina “D”:

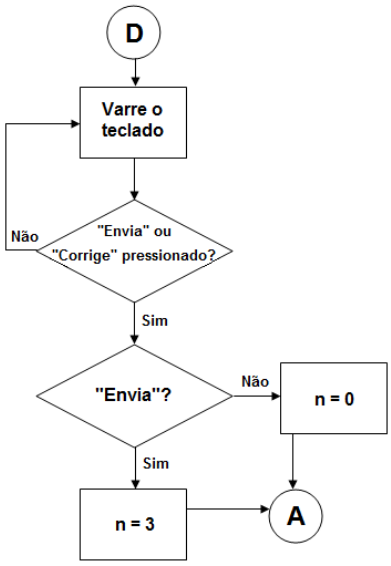

Figura 3.6 Fluxograma sub-rotina “D”

A sub-rotina “A” verifica o valor de “n” e chama a sub-rotina “E” que envia o pedido realizado pela porta serial do microcontrolador. Em seguida, o programa aguarda um sinal externo ser recebido pela porta RX do microcontrolador. Se o sinal não for recebido, o programa informa que houve uma falha no envio do pedido. Nos dois casos, “n” recebe “0” e o programa retorna para a sub-rotina “A” e o processo se repete. A figura 3.7 mostra o fluxograma da sub-rotina “E”:

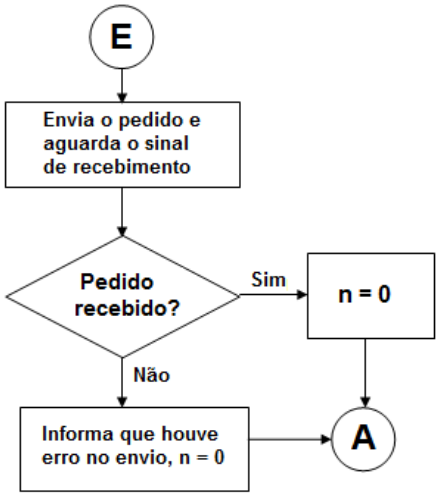

Figura 3.7 Fluxograma sub-rotina “E”

É importante ressaltar que, caso o botão de *MCLR* for pressionado, o programa se reinicia imediatamente, pois ele gera uma interrupção externa. Outro ponto de relevância é que há uma variável (cont) responsável pela contagem de tempo em que o programa está ocioso, ou seja, o programa está realizando a Varredura de Teclas sem que algum botão seja pressionado. Então quando “cont” atinge um determinado valor, ele aciona a interrupção de Timer 0 e coloca o dispositivo em Modo Sleep.

3.4. Plataforma de desenvolvimento

Para realizar a gravação e os testes com o microcontrolador, utilizou-se o PROGPIC 03 da empresa Datapool. O PROGPIC 03 permite o estudo da maioria dos microcontroladores da família PIC da MICROCHIP (12F, 16F e 18F). Pode funcionar nos módulos 2000, ou MTE. Comunica-se com o PC através de um cabo serial permitindo a gravação (ICD2) do programa utilizando o programa MPLAB da MICROCHIP. Possui alguns periféricos e permite o estudo de software e hardware(Datapool, 2013). A figura 3.8 mostra o kit PROGPIC 03:

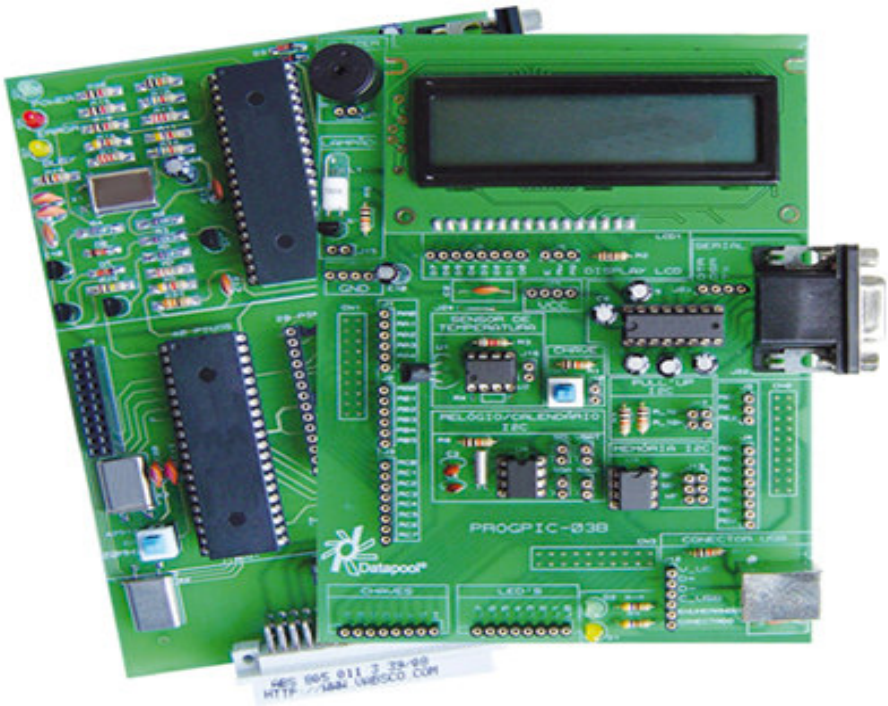

Figura 3.8 Kit de desenvolvimento PROGPIC 03

3.5. Protótipo do dispositivo eletrônico

A figura 3.9 mostra o dispositivo eletrônico após ser construído:

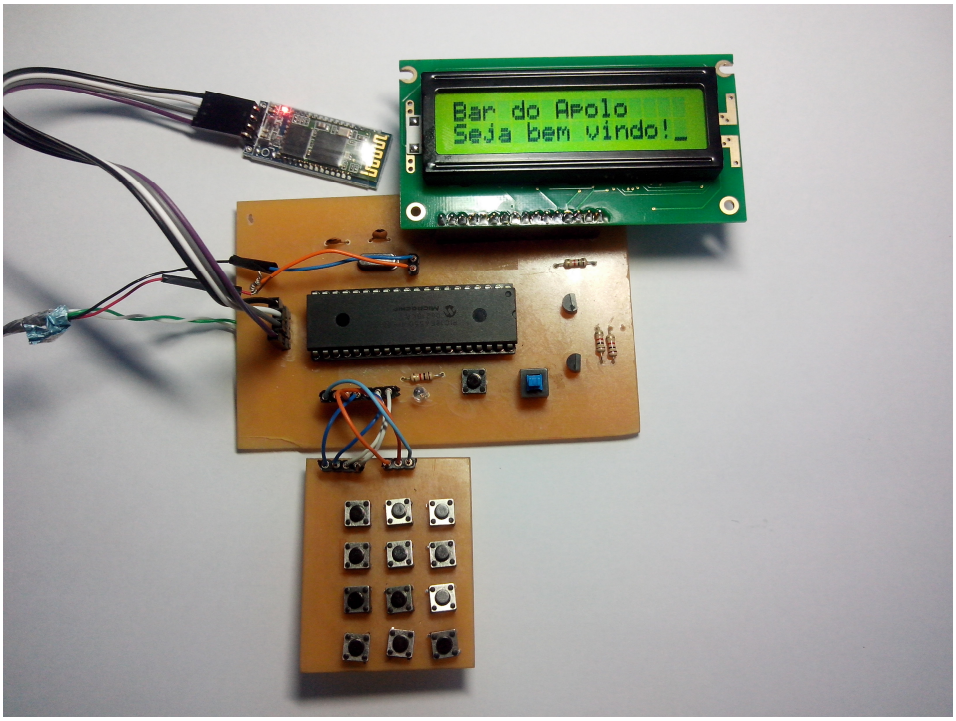

Figura 3.9 Dispositivo eletrônico

A Tabela 3.1 traz uma relação dos componentes utilizados neste Trabalho.

Tab. 3.1 - Componentes utilizados

Componente	Quantidade
PIC 16F877A	01
Display LCD	01
JY-MCU	01
LED	01
Capacitor 22p	02
Cristal 4 Mhz	01
Transistor BC 337	02
Resistor 10 kΩ	01
Resistor 1 kΩ	04
Chave táctil	13
Botão com retenção	01
Placa de fenolite 100x60	01
Placa de fenolite 50x40	01

Capítulo 4

Desenvolvimento da interface gráfica

4.1. Software de desenvolvimento – Delphi 7

Para o desenvolvimento da interface gráfica do projeto utilizou-se o Delphi 7 que utiliza a linguagem Pascal na sua programação.

4.1.1. Interface gráfica

A interface gráfica recebe as informações de todos os dispositivos e trata essas informações de modo que facilite a interpretação dos dados pelo usuário. A interface desenvolvida possibilita as seguintes funções:

- Recebe os pedidos de cada mesa, organizando-os separadamente em forma de lista. Cada mesa possui sua própria lista de pedidos;
- Apresenta o consumo atual de cada mesa do estabelecimento;
- Realiza o fechamento e impressão de conta;

- Realiza a adição de um novo produto no cardápio de estabelecimento;
- Gera um relatório de faturamento diário com o valor total arrecadado e com a lista de itens que foram consumidos.

A figura 4.1 apresenta a interface desenvolvida:

Figura 4.1 Interface gráfica

Ao ser inicializado o programa, primeiramente, é realizado o pareamento Bluetooth de todos dos dispositivos através do procedimento:

```
procedure TForm1.ApdComPort1PortOpen(Sender: TObject);
.
.
.
procedure TForm1.ApdComPortNPortOpen(Sender: TObject);
```

Momento após ocorrer o pareamento, um sinal é enviado para cada Módulo, para garantir que a conectividade está correta, através do procedimento "TForm1.FormShow(Sender: TObject);"

```
procedure TForm1.FormShow(Sender: TObject);
{
  i := Integer;
  begin
     for i := 1 to ApdComPort.Count do
     begin
     ApdComPort[i].Output := '!';
     end;
  end;
}
```

Após concluir esses procedimentos garante-se a comunicação entre a interface e os dispositivos.

O campo “Lista de Pedidos”, figura 4.2, é o local onde todos os pedidos recebidos são mostrados, independentemente de qual mesa realizou o pedido. Os pedidos são organizados em forma de lista e a descrição possui o número da mesa que efetuou o pedido, o número do pedido, a descrição do pedido e as horas que o pedido foi realizado. Deste modo facilita a verificação, pelo usuário, de qual mesa solicitou algum pedido, ao invés de conferir cada lista de mesa separadamente.

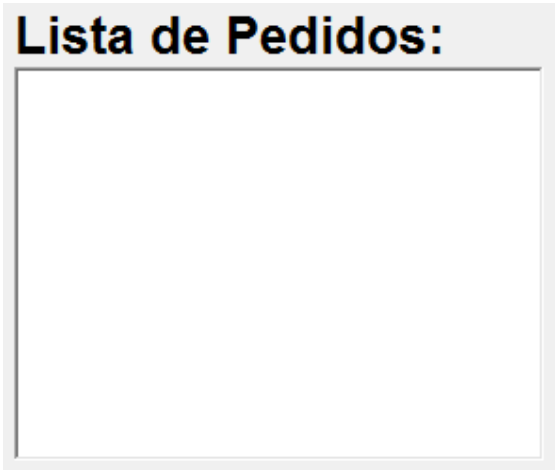

Figura 4.2 Campo “Lista de Pedidos”

O campo “Mesa”, figura 4.3, representa cada mesa do estabelecimento individualmente e todos os pedidos recebidos são armazenados, separadamente por mesa, em forma de lista. No canto superior direito é o local onde é mostrado o consumo atual da mesa que é automaticamente alterado quando um pedido é adicionado ou removido da lista.

Figura 4.3 Campo “Mesa”

O campo “Número da Mesa”, figura 4.4, é utilizado paralelamente com o funcionamento dos botões. Ao identificar a mesa neste campo, indica-se para qual mesa as ações devem ser direcionadas.

Figura 4.4 Campo “Número da Mesa”

O botão “Fechar Conta” realiza o fechamento de conta do seguinte modo: o programa gera um arquivo para ser impresso com o nome do estabelecimento, o número da mesa, a data e hora que o encerramento da conta foi solicitado, uma lista com todos os produtos consumidos e o valor total da conta. Caso a impressora do estabelecimento estiver conectada ao programa, a impressão é realizada automaticamente.

O botão “Nova Venda” é utilizado quando for necessário uma nova abertura de conta, pois é responsável por limpar a lista de todos os produtos e o valor total já consumidos na mesa em um momento anterior.

O botão “Cancelar Pedido” possui a função de cancelar e excluir um pedido de determinada mesa e quando ocorre a exclusão do pedido, o valor do pedido é decrementado diretamente do valor total consumido.

O botão “Adicionar Produto” é responsável por realizar a adição de novos produtos no cardápio do estabelecimento. Essa alteração é salva no mesmo arquivo do Excel de onde a lista dos produtos é carregada na inicialização do programa. Esta ação realiza a interação de dois formulários, o que fornece a opção de gerar uma nova janela, figura 4.5, para efetuar a função.

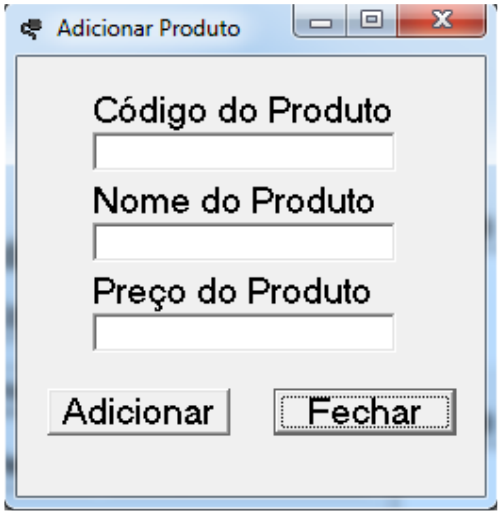

Figura 4.5 Adicionar Produto

O botão “Faturamento” retorna o faturamento total diário e a lista todos os produtos que foram consumidos naquele dia, o que facilita o processo de fechamento de caixa e controle de estoque do estabelecimento.

O botão “Limpar Mesas” possui a função de inicializar todas as mesas, ou seja, todos os campos das listas de pedidos e os campos dos valores consumidos são limpos.

Capítulo 5

Testes e Validação

5.1. Funcionamento do dispositivo eletrônico

Será apresentado o funcionamento do dispositivo desde que o momento que ele é ligado até quando é desligado automaticamente. Toda a interação entre o cliente/dispositivo e dispositivo/cliente é realizada através do Teclado apresentado na figura 5.1, e do *Display LCD* apresentado na figura 5.2, respectivamente.

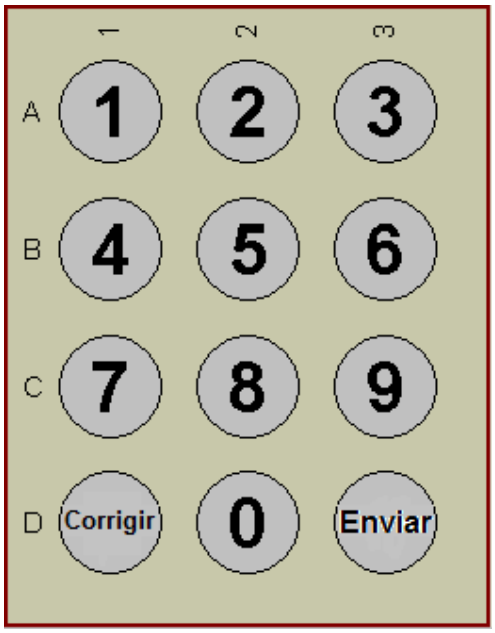

Figura 5.1 Teclado do dispositivo eletrônico

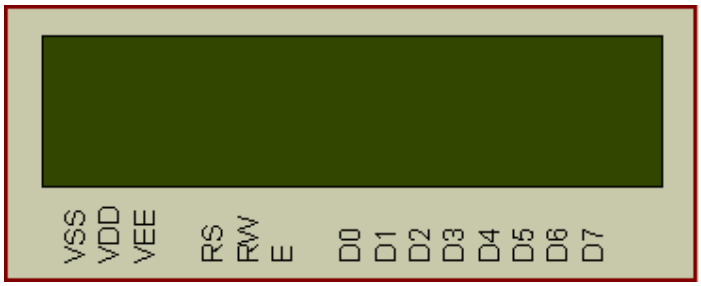

Figura 5.2 Display do dispositivo eletrônico

O processo se divide nas seguintes etapas:

- anúncio com o nome do estabelecimento juntamente com uma mensagem de boas vindas direcionada ao cliente;
- momento no qual o cliente realiza o pedido;
- confirmação de envio ou correção do pedido;
- envio do pedido para a central;
- aguardo do dispositivo pelo recebimento do pedido pela central;
- confirmação da central do recebimento do pedido;
- desligamento automático do dispositivo.

Para realizar a escolha do pedido, o cliente deve se orientar pelos códigos referentes aos pedidos no cardápio. O código de cada item precede a descrição do pedido conforme é apresentado na figura 5.3:

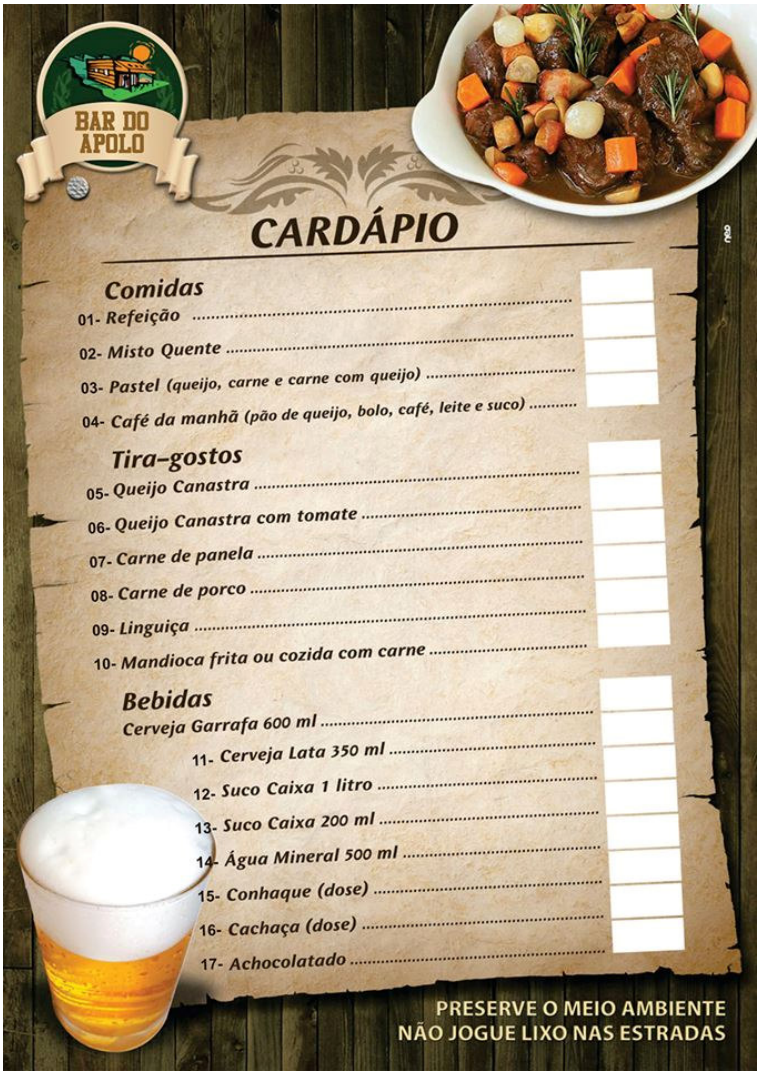

Figura 5.3 Cardápio com o código e a descrição dos pedidos

Ao ligar o dispositivo, é mostrado um texto com o nome do estabelecimento juntamente com uma mensagem de boas-vindas ao cliente, apresentado na figura 5.4,

Figura 5.4 LCD (Bar do Apolo/ Seja bem vindo!)

Em seguida, o dispositivo indica que o cliente está apto para realizar o seu pedido, apresentado na figura 5.5:

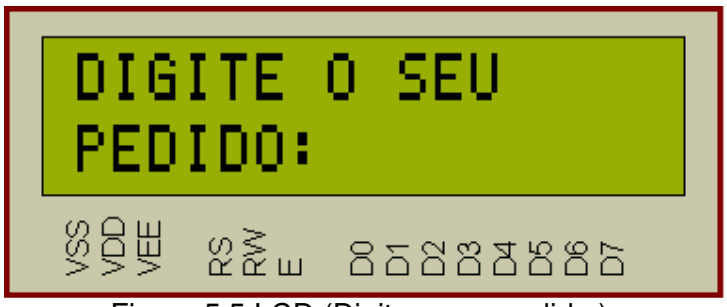

Figura 5.5 LCD (Digite o seu pedido:)

Este processo é iniciado e a entrada de dados é efetuada pelo teclado do dispositivo, apresentado na figura 5.6:

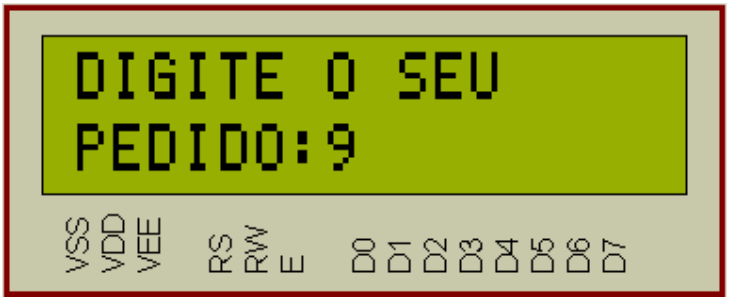

Figura 5.6 LCD (Digite o seu pedido: 9)

Em seguida, o cliente continua a efetuar a entrada de dados, apresentado na figura 5.7:

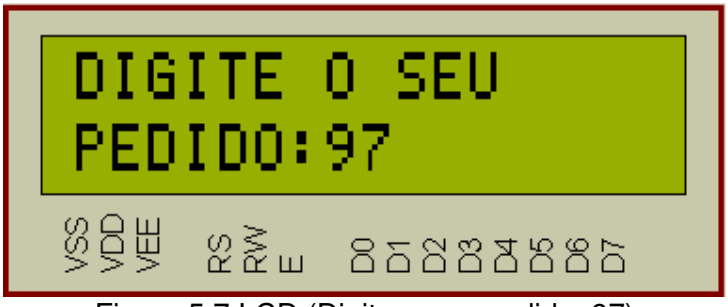

Figura 5.7 LCD (Digite o seu pedido: 97)

Neste momento, o cliente possui duas opções: Uma é a confirmação do pedido e a outra é a correção do pedido. Supondo que o pedido esteja equivocado, o cliente pode corrigir o pedido através do botão “Corrigir”, então poderá realizar o pedido novamente, apresentado na figura 5.8:

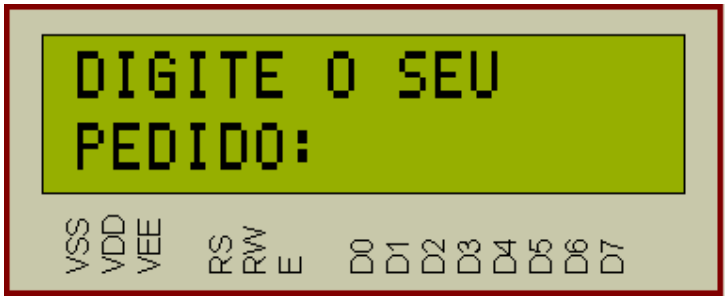

Figura 5.8 LCD (Digite o seu pedido:) – Opção “Corrigir”

Caso o pedido esteja correto, o cliente confirma o envio do pedido através do botão “Enviar”.

Neste momento, o dispositivo realiza o envio do pedido para a central e aguarda um sinal de *Acknowledge* para confirmar que a central recebeu o pedido de modo correto, apresentado na figura 5.9:

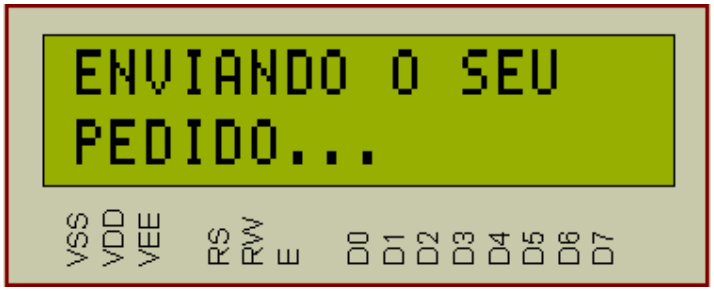

Figura 5.9 LCD (Enviando o seu pedido...)

Assim que o sinal de *Acknowledge* é recebido, então a confirmação de envio do pedido é realizada, apresentado na figura 5.10:

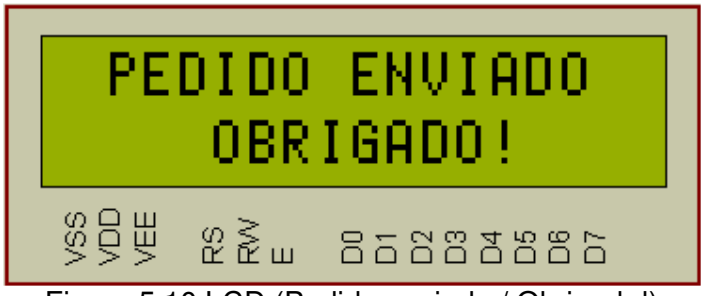

Figura 5.10 LCD (Pedido enviado / Obrigado!)

E o cliente pode continuar a realizar outros pedidos, repetindo o processo, apresentado na figura 5.11:

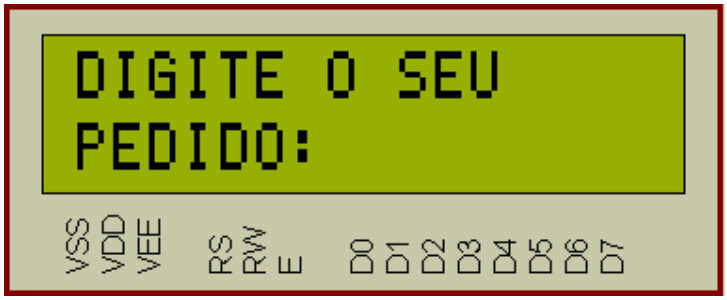

Figura 5.11 LCD (Digite o seu pedido:) - Continuação

Caso a central não conseguir receber o pedido, o dispositivo informa ao cliente que houve um erro ao enviar o pedido e indicando as providências que o cliente deve tomar, apresentado na figura 5.12:

Figura 5.12 LCD (Erro de envio / Chame o gerente!)

Após algum tempo sem ser utilizado, o dispositivo se desliga automaticamente e para iniciar o processo novamente, basta pressionar o botão “Iniciar”.

5.2. Funcionamento da interface

Será apresentado o funcionamento da interface gráfica do projeto realizando a comunicação com dois dispositivos. Primeiramente, a interface apresenta todos os seus campos vazios como foi mostrado na figura 4.1. Então, clientes de mesas diferentes realizam os seus pedidos e a interface se apresenta conforme mostrado na figura 5.13:

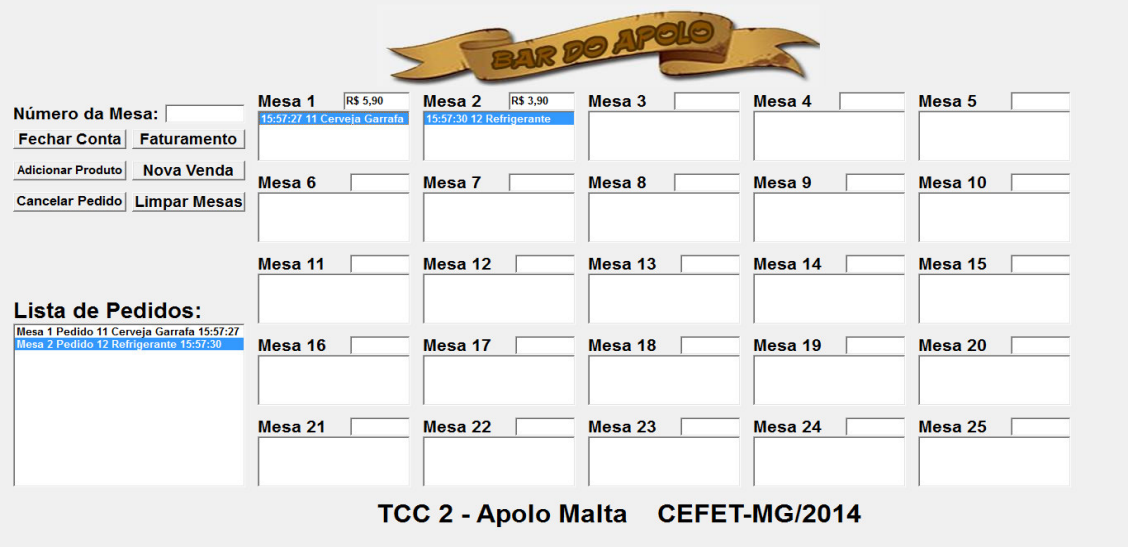

Figura 5.13 Interface – Pedidos 1

Pode-se perceber que os pedidos são organizados separadamente por mesa e que os preços dos itens são atualizados juntamente com o campo da mesa. O campo “Lista de Pedidos” recebe todos os pedidos realizados, de forma que o usuário da interface necessita apenas acompanhar este campo para checar os pedidos que foram enviados. Ao realizar outros pedidos ocorre o mesmo procedimento citado anteriormente. Após receber os pedidos a interface é preenchida conforme apresentado na figura 5.14:

Figura 5.14 Interface – Pedidos 2

Quando houver a necessidade de cancelar algum pedido deve-se preencher o campo “Número da Mesa” com o número da mesa que contém o pedido e selecioná-lo no campo onde se encontra como mostrado na figura 5.15:

Figura 5.15 Interface – Cancelar Pedido (pedido selecionado)

O segundo pedido da Mesa 2 foi selecionado e o campo “Número de Mesa” foi preenchido. Na figura 5.16 mostra quando o botão “Cancelar Pedido” é pressionado:

Figura 5.16 Interface – Cancelar Pedido (pedido cancelado)

O pedido é removido da lista da Mesa 2 e o preço do pedido é subtraído do valor atual consumido. Para iniciar uma nova venda, deve-se preencher o campo “Número da Mesa” com o número da mesa que irá iniciar a abertura de conta e pressionar o botão “Nova Venda”. Na figura 5.17 mostra quando o botão “Nova Venda” é pressionado:

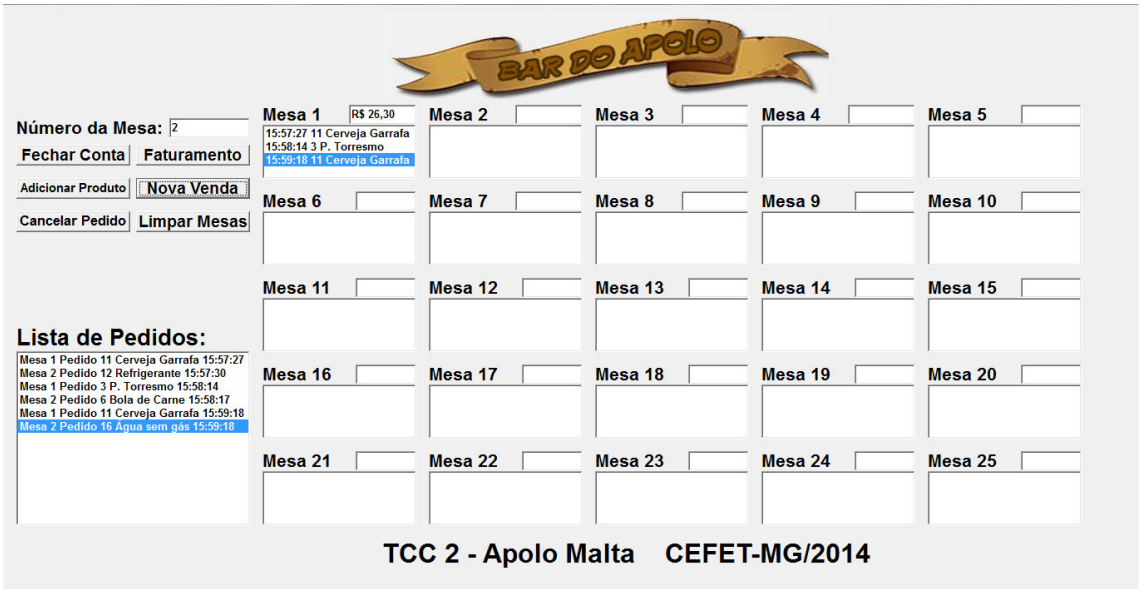

Figura 5.17 Interface – Nova Venda

Quando houver um encerramento de conta deve-se preencher o campo "Número da Mesa" com o número da mesa que se deseja fechar a conta e pressionar o botão "Fechar Conta". A figura 5.18 mostra o que acontece quando o botão "Fechar Conta" é pressionado:

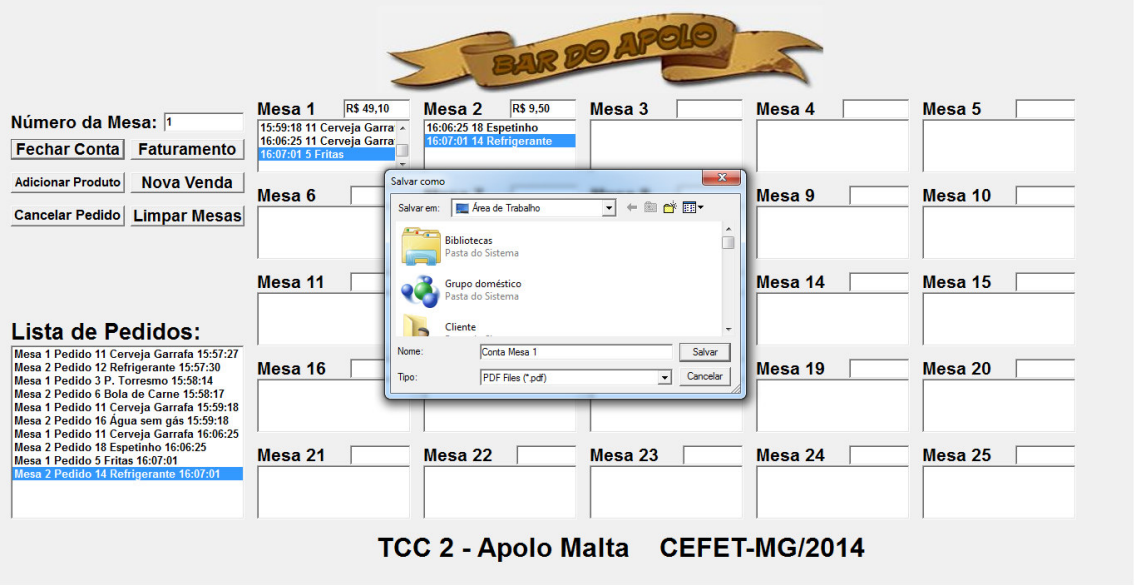

Figura 5.18 Interface – Fechar Conta

O programa gera um arquivo para ser impresso em formato de conta que é apresentado na figura 5.19:

Fechamento de conta: Mesa: 1
Data: 11/02/2014
Hora: 16:07:35

Qtd:	Desc:	P/Uni:	P/Tot:
3	Cerveja	5,90	17,70
1	P.Torres	14,50	14,50
1	Fritas	16,90	16,90

Valor: R$ 49,10

Figura 5.19 Arquivo de conta gerada

No momento em que se encerram as vendas e é inicializado o fechamento de caixa do estabelecimento, o usuário tem a opção de realizar um levantamento geral do valor total que foi arrecadado e de todos os itens que foram consumidos utilizando o botão “Faturamento”. Esta opção gera uma tabela no Excel com uma lista com a descrição de todos os itens que foram vendidos com a sua quantidade e com o valor total do faturamento daquele dia.

Capítulo 6

Conclusão e Propostas de Continuidade

Este trabalho proporcionou uma atividade com grande interdisciplinaridade, grande quantidade de experimentos práticos, além de toda a pesquisa que se fez necessária, caracterizando o que um Trabalho de Conclusão de Curso deve possuir.

6.1. Resultados

Um dos principais objetivos do projeto foi desenvolver um dispositivo de baixo custo e com qualidades equivalentes a outros produtos já existentes no mercado. Os preços citados nos orçamentos são referentes às empresas nacionais, ou seja, os impostos referentes aos produtos estão incluídos de acordo com as taxas cobradas no Brasil.

A tabela abaixo se refere ao valor total gasto para construir um dispositivo. Os preços foram extraídos de empresas revendedoras, não houve contato direto com o fabricante, ou seja, é possível ainda conseguir um valor inferior ao mostrado na tabela.

Tabela. 6.1 – Custo para a fabricação do dispositivo

Componente	**Quantidade**	**Preço/Uni**	**Preço**
PIC 16F877A	01	R$ 13,83	R$ 13,83
Display LCD	01	R$ 18,50	R$ 18,50
JY-MCU	01	R$ 12,50	R$ 12,50
LED	01	R$ 0,18	R$ 0,18
Capacitor 22p	02	R$ 0,04	R$ 0,08
Cristal 4 Mhz	01	R$ 0,37	R$ 0,37
Transistor BC 337	02	R$ 0,13	R$ 0,26
Resistor 10 kΩ	01	R$ 0,05	R$ 0,05
Resistor 1 kΩ	04	R$ 0,05	R$ 0,20
Chave táctil	13	R$ 0,20	R$ 2,60
Botão com retenção	01	R$ 0,23	R$ 0,23
Placa de fenolite 100x60	01	R$ 1,58	R$ 1,58
Placa de fenolite 50x40	01	R$ 0,60	R$ 0,60
Total			**R$ 50,98**

Na tabela 6.2, está o orçamento do produto “Smart Call’’ da Empresa Mafra,

Tabela. 6.2 – Orçamento "*Smart Call*", empresa Mafra

Produto	**Quantidade**	**Preço/Uni**	**Preço**
Display (três posições numéricas)	01	R$ 760,00	R$ 760,00
Campainhas	01	R$ 72,00	R$ 72,00
Total			R$ 832,00

O "*Smart Call*" funciona do seguinte modo: ao desejar ser atendido, o cliente aperta a Campainha, que envia um sinal digital para o Display onde é interpretado e exibido o número do local da chamada (mesa, *check-out*, etc.) Cada Display exibe simultaneamente até 3 números conforme a ordem de chamada.Vários Displays podem ser instalados no mesmo local (por zona) sem causar qualquer interferência (MAFRA,2013).

Percebe-se que este produto possui a seguinte limitação: o cliente não consegue realizar o pedido do local onde se encontra, sempre havendo a necessidade de um atendente para efetuá-lo. O dispositivo desenvolvido neste projeto é bem mais versátil e consegue preencher essa lacuna que o "Smart Call" possui, garantindo mais autonomia ao cliente.

A tabela abaixo apresenta o orçamento de um *tablet*, caso o proprietário do estabelecimento desejar implantar um sistema de aplicativos para Android,

Tabela. 6.3 – Orçamento *tablet*

Produto	**Quantidade**	**Preço/Uni**	**Preço**
Tablet Space BR Tela 7"	01	R$ 249,00	R$ 249,00
Total			R$ 249,00

Apesar de não haver o custo do sistema para aplicativos de *smartphones* e *tablets*, pode-se fazer as seguintes observações:

- O proprietário deve arcar com a despesa de uma rede wireless, já que os aplicativos se conectam através da internet;

- O proprietário deverá efetuar a compra de *tablets,* mesmo havendo clientes que já possuam dispositivos compatíveis com o sistema de aplicativos, para que não haja a exclusão de clientes que não possuam ou não estejam com os seus dispositivos.

- O proprietário deverá incluir o valor da implantação de um sistema para aplicativos no estabelecimento.

O dispositivo desenvolvido possui a vantagem de ter o próprio protocolo de comunicação, o que não gera custos adicionais com outras redes de comunicação.

Através das comparações e análises, percebe-se que o dispositivo desenvolvido possui vantagens operacionais sobre os dois tipos de produtos, além de apresentar um preço competitivo.

6.2. Propostas de continuidade

Como propostas de continuidade, existe a possibilidade de implementar um cardápio digital no dispositivo, assim o cliente poderá fazer a escolha o pedido diretamente da tela. Para esta implementação é sugerido utilizar um *Display LCD* 20x04 (4 linhas com 20 caracteres cada), pois possibilita uma quantidade maior de informação.

Outra ideia seria a implementação de um sistema de alimentação recarregável no dispositivo. Isso se faz necessário para proporcionar uma maior economia para o cliente que for adquirir o produto, evitando a compra de pilhas e baterias.

Também há a necessidade de desenvolver uma estrutura que seria a carcaça do dispositivo, deixando-o comercialmente mais estético e com fácil utilização para o usuário.

Bibliografia

MAFRA. http://www.mafrainformatica.com.br/novo_site/garcom.asp. Acesso em: 06 Agosto 2013.

MANZANO, J. A. N. G. **Programação de Computadores com C++**. 1. ed. [S.l.]: Érica, 2010.

MASTERING INFORMÁTICA. http://masteringinformatica.wordpress.com/2012/03/15/chama-atendente-2/ Acesso em: 06 Agosto 2013.

MICROCHIP. **PIC16F87X Data Sheet**. [S.l.]: [s.n.], 2001.

MICROCHIP. **PIC16F627A/628A/648A Data Sheet**. [S.l.]: [s.n.], 2007.

MICROCHIP. http://www.microchip.com/. Acesso em: 04 Agosto 2013.

MIYADAIRA, A. N. **Microcontroladores PIC18 - Aprenda e Programe em Linguagem C.** 4ª Revisada e Atualizada. ed. [S.l.]: Érica, 2012.

NEXTDATA. http://www.nextdata.com.br/produtos_restaurante.aspx. Acesso em: 06 Agosto 2013.

OLIVEIRA, J. F. D.; MANZANO, J. A. N. G. **Algoritmos - Lógica para Desenvolvimento de Programação de Computadores**. 22. ed. [S.l.]: Érica, 2009.

PEREIRA, F. **Microcontroladores PIC Técnicas Avançadas**. 4. ed. São Paulo: Érica, 2006.

PEREIRA, F. **PIC Programação em C**. 7. ed. São Paulo: Érica LTDA, 2012.

PINHEIRO, F. A. C. **Elementos de Programação Em C**. [S.l.]: Bookman, 2012.

RUNZE. http://www.runze.com.br/o-que-fazemos/jbar-automacao-para-bares-e-restaurantes. Acesso em: 6 Agosto 2013.

SEDRA, A. S. **Microeletrônica. 5. ed**. [S.l.]: Prentice Hall - Br, v. Único, Sedra, Adel S.

SILVA, R. A. **Programando Microcontroladores PIC Linguagem "C"**. 1. ed. São Paulo: Ensino Profissional, 2006.

SOUZA, D. J. D. **Desbravando o PIC**. 12. ed. São Paulo: Érica LTDA, 2013.

SOUZA, D. J. D.; LAVINIA, N. C. **Conectando o PIC Recursos Avançados**. 3. ed. São Paulo: Érica LTDA.

SUSAN A. RIEDEL, J. W. N. **Circuitos Elétricos**. 8. ed. [S.l.]: PRENTICE HALL (BRASIL), 2008.

TOCCI, R. J. Sistemas **Digitais Princípios e Aplicações**. 11. ed. [S.l.]: Pearson, 2011.

LEÃO, M. **Borland Delphi 7 Curso Completo**. 5. ed. Axcel Books do Brasil, 2008.

Anexos

MOD_LCD.C - Biblioteca de manipulação de módulo LCD
Autor: Fábio Pereira

```
/************************************************************************/
/*  MOD_LCD.C - Biblioteca de manipulação de módulo LCD                 */
/*                                                                      */
/*  Autor: Fábio Pereira                                                */
/*                                                                      */
/************************************************************************/

// As definições a seguir são utilizadas para acesso aos pinos do display
// caso o pino RW não seja utilizado, comente a definição lcd_rw
#ifndef lcd_enable
   #define lcd_enable     pin_b1      // pino enable do LCD
   #define lcd_rs         pin_b0      // pino rs do LCD
   //#define lcd_rw       pin_e2      // pino rw do LCD
   #define lcd_d4         pin_b4      // pino de dados d4 do LCD
   #define lcd_d5         pin_b5      // pino de dados d5 do LCD
   #define lcd_d6         pin_b6      // pino de dados d6 do LCD
   #define lcd_d7         pin_b7      // pino de dados d7 do LCD
#endif

#define lcd_type 2           // 0=5x7, 1=5x10, 2=2 linhas
#define lcd_seg_lin 0x40    // Endereço da segunda linha na RAM do LCD

// a constante abaixo define a seqüência de inicialização do módulo LCD
byte CONST INI_LCD[4] = {0x20 | (lcd_type << 2), 0xf, 1, 6};

byte lcd_le_byte()
// lê um byte do LCD (somente com pino RW)
{
   byte dado;
   // configura os pinos de dados como entradas
   input(lcd_d4);
   input(lcd_d5);
   input(lcd_d6);
   input(lcd_d7);
   // se o pino rw for utilizado, coloca em 1
   #ifdef lcd_rw
      output_high(lcd_rw);
   #endif
   output_high(lcd_enable); // habilita display
   dado = 0;   // zera a variável de leitura
   // lê os quatro bits mais significativos
   if (input(lcd_d7)) bit_set(dado,7);
   if (input(lcd_d6)) bit_set(dado,6);
```

```
  if (input(lcd_d5)) bit_set(dado,5);
  if (input(lcd_d4)) bit_set(dado,4);
  // dá um pulso na linha enable
  output_low(lcd_enable);
  output_high(lcd_enable);
  // lê os quatro bits menos significativos
  if (input(lcd_d7)) bit_set(dado,3);
  if (input(lcd_d6)) bit_set(dado,2);
  if (input(lcd_d5)) bit_set(dado,1);
  if (input(lcd_d4)) bit_set(dado,0);
  output_low(lcd_enable);   // desabilita o display
  return dado;   // retorna o byte lido
}

void lcd_envia_nibble( byte dado )
// envia um dado de quatro bits para o display
{
  // coloca os quatro bits nas saidas
  output_bit(lcd_d4,bit_test(dado,0));
  output_bit(lcd_d5,bit_test(dado,1));
  output_bit(lcd_d6,bit_test(dado,2));
  output_bit(lcd_d7,bit_test(dado,3));
  // dá um pulso na linha enable
  output_high(lcd_enable);
  output_low(lcd_enable);
}


void lcd_envia_byte( boolean endereco, byte dado )
{
  // coloca a linha rs em 0
  output_low(lcd_rs);
  // aguarda o display ficar desocupado
  //while ( bit_test(lcd_le_byte(),7) ) ;
  // configura a linha rs dependendo do modo selecionado
  output_bit(lcd_rs,endereco);
  delay_us(100);   // aguarda 100 us
  // caso a linha rw esteja definida, coloca em 0
  #ifdef lcd_rw
    output_low(lcd_rw);
  #endif
  // desativa linha enable
  output_low(lcd_enable);
  // envia a primeira parte do byte
  lcd_envia_nibble(dado >> 4);
  // envia a segunda parte do byte
  lcd_envia_nibble(dado & 0x0f);
}
```

```
void lcd_ini()
// rotina de inicialização do display
{
  byte conta;
  output_low(lcd_d4);
  output_low(lcd_d5);
  output_low(lcd_d6);
  output_low(lcd_d7);
  output_low(lcd_rs);
  #ifdef lcd_rw
    output_high(lcd_rw);
  #endif
  output_low(lcd_enable);
  delay_ms(15);
  // envia uma seqüência de 3 vezes 0x03
  // e depois 0x02 para configurar o módulo
  // para modo de 4 bits
  for(conta=1;conta<=3;++conta)
  {
    lcd_envia_nibble(3);
    delay_ms(5);
  }
  lcd_envia_nibble(2);
  // envia string de inicialização do display
  for(conta=0;conta<=3;++conta) lcd_envia_byte(0,INI_LCD[conta]);
}

void lcd_pos_xy( byte x, byte y)
{
  byte endereco;
  if(y!=1)
    endereco = lcd_seg_lin;
  else
    endereco = 0;
  endereco += x-1;
  lcd_envia_byte(0,0x80|endereco);
}

void lcd_escreve( char c)
// envia caractere para o display
{
  switch (c)
  {
    case '\f'    :   lcd_envia_byte(0,1);lcd_envia_byte(0,0x0c);
                     delay_ms(2);
                   break;
    case '\n'   :
    case '\r'    :   lcd_pos_xy(1,2);
                     break;
    case '\b'    :   lcd_envia_byte(0,0x10);
```

```
            break;
    default   :  lcd_envia_byte(1,c);
            break;
  }
}

char lcd_le( byte x, byte y)
// le caractere do display
{
  char valor;
  // seleciona a posição do caractere
  lcd_pos_xy(x,y);
  // ativa rs
  output_high(lcd_rs);
  // lê o caractere
  valor = lcd_le_byte();
  // desativa rs
  output_low(lcd_rs);
  // retorna o valor do caractere
  return valor;
}
```

Programa que realiza a varredura do teclado 4x4 e escreve o valor no display:

```
/*****************************************************************************
* Varredura do Teclado Matricial e Display de 7 segmentos*
*****************************************************************************
* RAM: 2% ROM: 6% *
*****************************************************************************
* Exibe o número ou letra em um display 7 segmentos de acordo com o *
* botão pressionado no teclado matricial de 0 a 9, A,C,E,F,H,I *
*****************************************************************************/
#include <16f877a.h> // PIC 16F877A
#device adc=10 // Configura conversor AD para 10 bits
#fuses    XT,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT    //
Configs internas
#use delay (clock=4000000) // Seleciona delay para 4Mhz de clock
/*****************************************************************************
*****************************************************************************/
// Definições de pinos
#define c0 pin_b4 // Primeira coluna do teclado
#define c1 pin_b5 // Segunda coluna do teclado
#define c2 pin_b6 // Terceira coluna do teclado
#define c3 pin_b7 // Quarta coluna do teclado
#define l0 pin_b0 // Primeira linha do teclado
#define l1 pin_b1 // Segunda linha do teclado
#define l2 pin_b2 // Terceira linha do teclado
#define l3 pin_b3 // Quarta linha do teclado
```

```
/*************************************************************************
*************************************************************************/
// Variaveis globais
int cont; // Variável auxiliar para exibir o número ou letra no display
/*************************************************************************
*************************************************************************/
// Tabela para converter o valor da contagem para exibir no display
byte                                                                    const
tabela[]={0xbf,0x86,0xdb,0xcf,0xe6,0xed,0xfd,0x87,0xff,0xe7,0xf7,0xb9,0xf9,0xf1,0xf
6,0xb0};
//                {0,1,2,3,4,5,6,7,8,9,A,C,E,F,H,I}
/*************************************************************************
*************************************************************************/
void exibe(int num);
void varredura1();
void varredura2();
void varredura3();
void varredura4();

// Rotina principal do programa
main(){ // Rotina principal
  setup_adc_ports(no_analogs); // Configura todas as entradas como digitais
  set_tris_a(0b00000000); // configuração da direção dos pinos de I/O
  set_tris_b(0b11110000);
  set_tris_c(0b00000000);
  set_tris_d(0b00000000);
  set_tris_e(0b00000000);
  while(1){ // Laço infinito
    varredura1(); // Chama primeira varredura
    varredura2(); // " segunda "
    varredura3(); // " terceira "
    varredura4(); // " quarta "
  } // Fim do laço
} // Fim do programa

// Exibe no display o valor do botão pressionado
void exibe(int num){
  delay_ms(250);
  output_d(tabela[num]); // Mostra tabela[cont] no display
}

// Fim da sub-rotina
/*************************************************************************/
// Varredura de teclas
void varredura1(){ // Faz primeira varredura
  int aux=50; // Tempo de varredura
  while(aux != 0){ // Laço para varredura
    output_low(c0); // Joga 0 na coluna a ser varrida e 1 nas demais
    output_high(c1);
    output_high(c2);
```

```
      output_high(c3);
      if(!input(l0)) // Testa primeira linha
        exibe(1);   // Chama a sub-rotina correspondente a tecla

      if(!input(l1)) // Testa segunda linha
        exibe(2);   // Chama a sub-rotina corresponde a tecla

      if(!input(l2)) // Testa terceira linha
        exibe(3);   // Chama a sub-rotina corresponde a tecla

      if(!input(l3)) // Testa quarta linha
        exibe(10); // Chama a sub-rotina corresponde a tecla A

      aux--; // Decrementa 1 de aux
      delay_us(aux); // Tempo de atraso da varredura é igual a aux
    }
} // Fim da sub-rotina
/*************************************************************************/
void varredura2(){ // Faz segunda varredura
   int aux=50; // Tempo de varredura
   while(aux != 0){ // Laço para varredura
      output_high(c0); // Joga 0 na coluna a ser varrida e 1 nas demais
      output_low(c1);
      output_high(c2);
      output_high(c3);
      if(!input(l0)) // Testa primeira linha
        exibe(4);  // Chama a sub-rotina correspondente a tecla

      if(!input(l1)) // Testa segunda linha
        exibe(5);   // Chama a sub-rotina corresponde a tecla

      if(!input(l2)) // Testa terceira linha
        exibe(6);   // Chama a sub-rotina corresponde a tecla

      if(!input(l3)) // Testa quarta linha
        exibe(11); // Chama a sub-rotina corresponde a tecla C

      aux--; // Decrementa 1 de aux
      delay_us(aux); // Tempo de atraso da varredura é igual a aux
   }
} // Fim da sub-rotina
/*************************************************************************/
void varredura3(){ // Faz terceira varredura
   int aux=50; // Tempo de varredura
   while(aux != 0){ // Laço para varredura
      output_high(c0); // Joga 0 na coluna a ser varrida e 1 nas demais
      output_high(c1);
      output_low(c2);
      output_high(c3);
      if(!input(l0)) // Testa primeira linha
```

```
      exibe(7);   // Chama a sub-rotina correspondente a tecla

    if(!input(l1)) // Testa segunda linha
      exibe(8);   // Chama a sub-rotina corresponde a tecla

    if(!input(l2)) // Testa terceira linha
      exibe(9);   // Chama a sub-rotina corresponde a tecla

    if(!input(l3)) // Testa quarta linha
      exibe(12);  // Chama a sub-rotina corresponde a tecla E

    aux--; // Decrementa 1 de aux
    delay_us(aux); // Tempo de atraso da varredura é igual a aux
  }
} // Fim da sub-rotina
/*************************************************************************/
void varredura4(){ // Faz quarta varredura
  int aux=50; // Tempo de varredura
  while(aux != 0){ // Laço para varredura
    output_high(c0); // Joga 0 na coluna a ser varrida e 1 nas demais
    output_high(c1);
    output_high(c2);
    output_low(c3);
    if(!input(l0)) // Testa primeira linha
      exibe(13);  // Chama a sub-rotina correspondente a tecla F

    if(!input(l1)) // Testa segunda linha
      exibe(0);   // Chama a sub-rotina corresponde a tecla

    if(!input(l2)) // Testa terceira linha
      exibe(14);  // Chama a sub-rotina corresponde a tecla H

    if(!input(l3)) // Testa quarta linha
      exibe(15);  // Chama a sub-rotina corresponde a tecla I

    aux--; // Decrementa 1 de aux
    delay_us(aux); // Tempo de atraso da varredura é igual a aux
   }
} // Fim da sub-rotina
/*************************************************************************
*************************************************************************/
```