CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB
FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

“Carrinho de compras inteligente”

Autor:

REGIS LEVINO DE OLIVEIRA
2002508/8

Prof. M.C. Maria Marony Sousa Farias Nascimento
Orientadora

Brasília-DF, junho de 2009.

CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB
FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

“Carrinho de compras inteligente”

Brasília-DF, junho de 2009.

Agradecimentos

Agradeço a Deus em primeiro lugar pela saúde e possibilidade de trabalho.

Agradeço em especial à minha mãe e ao meu pai, que proporcionaram esta realização, pelo carinho, atenção, dedicação, incentivo e total apoio nos momentos mais difíceis.

Aos meus irmãos, por sempre acreditarem na minha vitória.

À minha namorada Tyessa por estar sempre ao meu lado.

Aos meus tios José Raimundo e Joaquim, pelos materiais emprestados.

À minha tia Lili, por suas orações e demonstração de carinho.

À Mestra Profª Maria Marony, pela dedicação, atenção e esforço em me orientar, e ao Mestre Profº Francisco Javier, por todo apoio e incentivo.

Resumo

O objetivo deste trabalho é atribuir melhorias a um instrumento, uma caneta ótica, capaz de registrar e contabilizar automaticamente o nome e valor de produtos, via código de barras, enquanto simultaneamente exibe-os na tela da caneta, bem como o somatório de todos os valores lidos. Ao término das compras, o instrumento é capaz de transferir os dados registrados para o computador central e no ato o cliente recebe a fatura.

O projeto desenvolvido decodifica a leitura do código de barras, extrai o valor da compra e o envia a um servidor por meio de uma porta serial Bluetooth. Os dados são armazenados em um servidor de banco de dados. Um software instalado nos computadores dos operadores de caixa recupera os dados no banco de dados e imprimem na tela. Dessa forma, o cliente pode fazer o pagamento. O projeto foi planejado e desenvolvido com essas ferramentas, em conformidade com o padrão do processo operacional. O protótipo foi construído e testado de maneira sistemática e racional, o que permite alcançar os objetivos propostos.

Palavras-chave: informática**,** banco de dados**,** Bluetooth, políticas de atendimento ao cliente.

Abstract

The objective of this study is to attribute improvements of a device, optical scanners, which is able of reading a barcode and automatically register the name of the product and its price. Simultaneously it computs the total amont of these registered values, whilst it displays in a built-in screen the last recorded variables. At the cashier the device is able to transfer all recorded data to the main frame, while the customer receives the bill without delay.

The developed program decodes and transfers the total sun of values to a serial Bluetooth port. The data are recovered into database server. A software installed in computers of services box recover the data in database and print at monitor. So, the client can make the payment. The project is planed and developed using these tools according to standard procedures. A prototype is developed and tested in a systematic and rational way, in order to accomplish the objectives.

Key words: information systems, database, Bluetooth, customer policy.

SUMÁRIO

Lista de Figuras

CAPÍTULO 1. INTRODUÇÃO

1.1 Motivação

Este projeto tem como base, um projeto anterior, implementado por Luciano Cortez Toledo, de Engenharia de Computação, nesta instituição, chamado *Caneta Ótica para registro e contabilização automática de produtos*, onde era realizada a contabilização automática de produtos de um supermercado. Esse projeto apresentava algumas limitações tais como: o envio dos dados contabilizados é feito de forma manual através de um *pendrive* e o dispositivo não apresentava relatórios estatísticos. Fazia-se, então, necessário automatizar a transmissão dos dados coletados na caneta para o usuário (com a implementação de rede *wireless*) e uso de um banco de dados para gerar relatórios estatísticos, o que é feito neste trabalho. Este projeto consta de 6 capítulos. No capítulo 1 é abordado a motivação e os objetivos do projeto. O capítulo 2 apresenta os problemas, ou limitações do projeto anterior. No capítulo 3, é mencionada a parte teórico-científica em que o projeto foi construído e serve de apoio para o capítulo 4. O capítulo 4 descreve o desenvolvimento do projeto e está dividido e duas etapas. A etapa 1 descreve o desenvolvimento do hardware, as ligações elétricas e o código que controla o sistema. A etapa 2 descreve o desenvolvimento do servidor e a criação e integração com o banco de dados. O capítulo 5 menciona os testes e o capítulo 6 apresenta a conclusão e propostas de estudos futuros.

1.2 Objetivo geral do trabalho

O objetivo geral deste trabalho é implementar melhorias no projeto citado anteriormente, o qual apresentava dificuldades na transmissão dos dados coletados ao usuário (caixa de um supermercado) por ser de forma manual. No presente trabalho não há a guarda dos dados coletados em um sistema de banco de dados de forma automática. A proposta é implementar a transmissão dos dados de forma automática, criar um banco de dados e gerar relatórios com os dados coletados pela caneta ótica. Isso permitirá agilizar o processo de pagamento dos clientes e reduzir seu tempo no supermercado.

1.3 Objetivos específicos

O principal objetivo é acrescentar melhorias ao projeto, anteriormente citado, tais como:

1) Uma caneta ótica padrão utilizada em lojas comerciais (com leitura a laser SERIAL) com microcontrolador PIC16F876A.
2) Adaptar uma placa com sinal Bluetooth ao dispositivo que tem a caneta acoplada para o envio dos dados ao servidor.
3) Implementar uma solução em banco de dados utilizando o software Microsoft Access 2003 para a guarda das informações de forma centralizada, organizada e segura.
4) Implementar um software programado em Delphi que será usado pelo operador de caixa no supermercado. Esse software será usado para visualizar as compras realizadas pelo cliente.

5) Desenvolver um software em Delphi e instalá-lo no servidor para que esse computador possa receber os dados e armazená-los no banco de dados.
6) Desenvolver um software em basic para controlar o Hardware desenvolvido.

1.4 Justificativa e relevância do tema

Carrinho de compras de um supermercado inteligente é um projeto que visa diminuir o tempo de espera nas filas dos caixas de supermercado. Os supermercados atualmente não possuem métodos automáticos para agilizar os pagamentos dos produtos, o que contribui para a insatisfação dos compradores.

1.5 Escopo do trabalho

Este projeto tem como foco principal a implementação de uma solução em banco de dados, para o armazenamento e gerenciamento dos dados enviados pela caneta ótica e o teclado acoplado à caneta. Também tem como foco a integração da tecnologia Bluetooth para o envio dos dados lidos pela caneta ao servidor de banco de dados.

Este projeto não contempla uma solução de segurança para evitar que clientes de supermercado não passem os produtos no leitor de código de barras e assim consigam extraviar os produtos. Também não contempla um dispositivo capaz de detectar quais produtos não teve o código de barras informado à caneta ótica.

Neste projeto é proposta a automação de um carrinho de supermercado, de forma a permitir que o cliente possa realizar compras e contabilizá-la de forma

automática, enviando estes dados por meio da tecnologia Bluetooth até um servidor. Um software instalado no computador do caixa carregará os dados, armazenados no servidor, na tela e finalizará a compra do cliente. Na figura 1.1 é mostrada uma visão geral do produto final deste projeto.

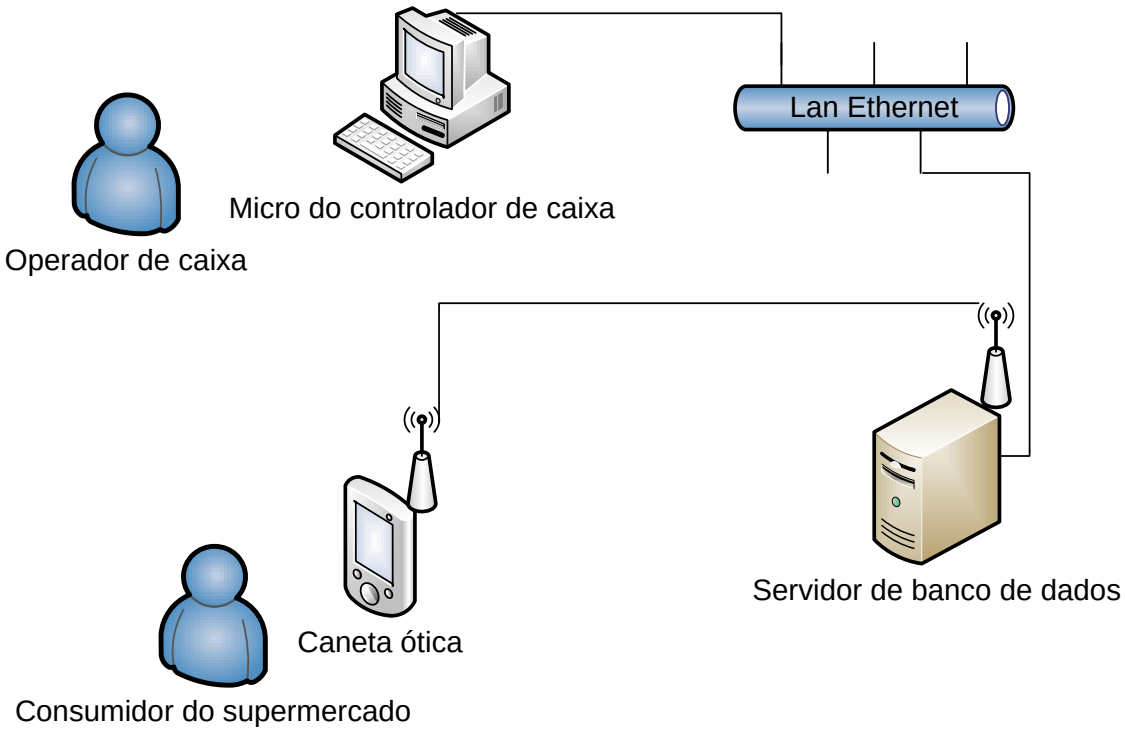

Figura 1. 1 – Visão geral do projeto

CAPÍTULO 2. APRESENTAÇÃO DO PROBLEMA

Pouca memória do micro-controlador escolhido para o projeto. Neste projeto, o micro-controlador foi trocado por um com memória maior;

Elevado tempo de espera por parte do cliente porque os dados coletados ficavam armazenados em um *pendrive,* os quais eram entregues ao caixa para verificar a contabilização. A transmissão dos dados do cliente para o caixa era feita de forma manual. Neste projeto, a transmissão é feita por meio de um dispositivo com transmissão Bluetooth;

O projeto anterior não podia gerar relatórios estatísticos como: o tempo do início de compras até o pagamento (tempo de espera no caixa), perfil de compras do cliente (caso o dono do estabelecimento comercial quiser presentear clientes que consomem certa quantidade do mesmo produto semanalmente, mensalmente) e etc. No projeto proposto, com a implementação do banco de dados, é possível ter histórico dos dados do cliente para formar as estatísticas de freqüência de compras, tempo de espera, produtos consultados e não comprados e etc, caso a empresa deseje usar estes dados com essa finalidade.

Com a continuação dos problemas citados, o equipamento não consegue gerar relatórios futuros dos itens comprados. O tempo que o cliente perde enquanto o caixa insere o *pendrive* para a coleta dos dados pode ser eficientemente resolvido com a transmissão de dados dinâmicos.

CAPÍTULO 3. REFERENCIAL TEÓRICO E BASES METODOLÓGICAS

3.1 Resumo

A idéia básica é acoplar um dispositivo de comunicação Bluetooth e substituir o uso de um *pendrive* no armazenamento dos dados coletados. Os dados, ao invés de serem armazenados em um arquivo.txt, são enviados imediatamente a um servidor por meio de comunicação Bluetooth. Neste projeto um servidor de banco de dados que armazena os dados gerados pelo carrinho de compras inteligente. Os componentes usados no projeto são:

- Kit Cerne Bluetooth com microcontrolador PICLAB16F876A;
- Scanner para leitura ótica de código de barras comum;
- Software mikrobasic para programar o microcontrolador;
- Software Delphi para criar o software do operador de caixa;
- Software Access 2003 para criar o banco de dados do servidor;
- Dispositivo Bluetooth USB acoplado ao servidor para a recepção dos dados.

3.2 Descrição dos componentes principais usados no projeto

3.2.1 Comunicação Bluetooth

A tecnologia Bluetooth é uma tecnologia de redes de comunicação sem fio para sistemas de curto alcance. Usa freqüência de rádio não licenciada 2.4 GHz ISM (Industrial Scientific Medical), freqüência também usada no padrão IEEE 802.11. Apesar de não ter sido criado pelo SIG (Special Interest Group), um consórcio de

empresas – Ericsson, IBM, Intel, Nokia, Toshiba -, foi padronizada pelo IEEE. Está classificada no padrão IEEE 802.15.

Como nos mostra o grande autor Tanenbaum, a arquitetura do Bluetooth funciona da seguinte forma: **"**A unidade básica de um sistema Bluetooth é uma piconet, que consiste em um nó mestre e até sete nós escravos, situados dentro de uma distancia de 10 metros".

Os nós escravos só se comunicam com o mestre. Não é possível haver comunicação diretamente entre nós escravos. As piconets podem ser conectadas umas as outras por um nó de ponte. Essa interconexão de piconets é chamada de scatternet. Em geral, os protocolos de comunicação entre máquinas não se preocupam com as aplicações que irão comunicar-se. No caso do Bluetooth, foram projetados 13 aplicações. Essas aplicações são conhecidas como "profiles". A figura 3.1 mostra as 13 aplicações:

Name	Description
Generic access	Procedures for link management
Service discovery	Protocol for discovering offered services
Serial port	Replacement for a serial port cable
Generic object exchange	Defines client-server relationship for object movement
LAN access	Protocol between a mobile computer and a fixed LAN
Dial-up networking	Allows a notebook computer to call via a mobile phone
Fax	Allows a mobile fax machine to talk to a mobile phone
Cordless telephony	Connects a handset and its local base station
Intercom	Digital walkie-talkie
Headset	Allows hands-free voice communication
Object push	Provides a way to exchange simple objects
File transfer	Provides a more general file transfer facility
Synchronization	Permits a PDA to synchronize with another computer

Figura 3. 1 – Aplicações do Bluetooth. Fonte: TANENBAUM, Andrew S., 2003

O perfil acesso genérico é usada como um modelo para a criação das aplicações que usarão o Bluetooth. Tanenbaum fala que a principal função do perfil acesso genérico “é fornecer um meio para estabelecer e manter enlaces seguros (canais) entre o mestre e os escravos”. O perfil descoberta de serviço é usado para localizar quais serviços estão disponíveis no dispositivo. O perfil de porta serial é usado para simular, às aplicações que se utilizam de porta serial, uma comunicação em série. O perfil de intercâmbio genérico de objetos – Tanenbaum – define um relacionamento cliente/servidor para movimentação de dados. Os clientes iniciam operações, mas um escravo pode ser um cliente ou um servidor. O perfil acesso de LAN permite um dispositivo acessar a rede LAN. O perfil rede dial-up permite discar a um telefone móvel. O perfil de fax que dispositivos de fax troquem mensagem sem fio. O perfil de telefonia sem fio permite a conexão dos telefones fixos à estação base. Tanenbaum afirma que “o perfil de intercomunicação permite que dois telefones se conectem como intercomunicadores. O perfil de fone de ouvido é usado para trafegar voz, usado em dispositivos telefônicos. O perfil push de objetos é usado para o envio e dados como imagens, arquivos de dados comuns etc. O perfil de transferência de arquivos define como arquivos e pastas em um dispositivo pode ser pesquisado por um dispositivo cliente. O perfil de sincronização é usado em conjunto com o acesso genérico para permitir a sincronização de calendários, informações pessoais entre dois dispositivos Bluetooth.

A tecnologia Bluetooth apresenta muitos protocolos. O IEEE tem aprimorado-o para se adaptar ao modelo IEEE 802. A figura 3.2 mostra a arquitetura básica de protocolos do Bluetooth do padrão IEEE 802.15.

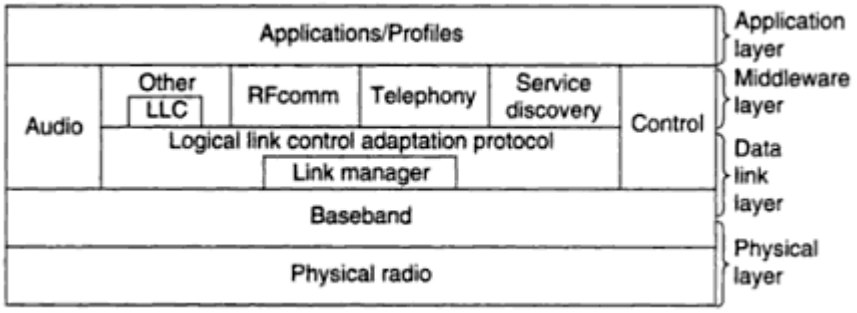

Figura 3. 2 – Arquitetura dos protocolos. Fonte: TANENBAUM, Andrew S., 2003

A camada física trabalha com o modo de transmissão e modulação. O sistema de rádio usado no Bluetooth opera na banda ISM de 2,4 GHz, a mesma usada no padrão IEEE 802.11. A banda é dividida em 79 canais de 1 MHz, cada uma, afirma Tanenbaum. A camada de banda-base tem a função parecida com o MAC (controle de acesso ao meio). Dois ou mais dispositivos compartilham o mesmo canal de um piconet. Um dispositivo é o mestre e o outro o escravo. A figura 3.3 ilustra a como dispositivos se conectam em piconets e scatternet.

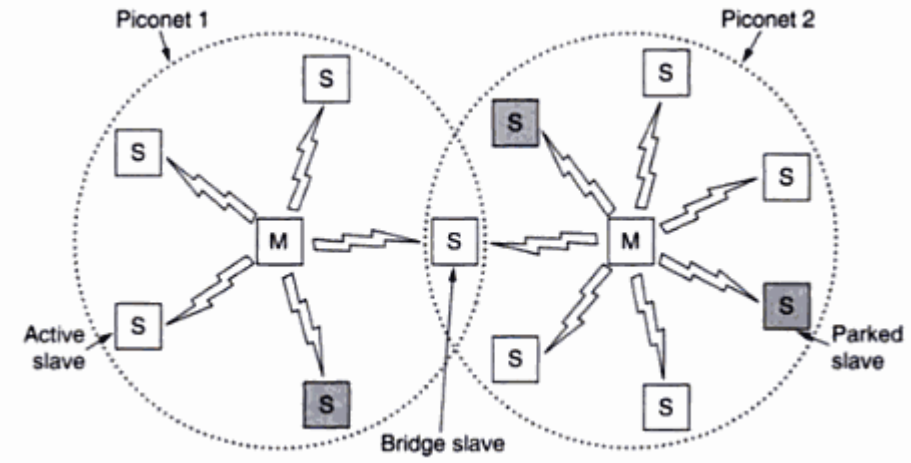

Figura 3. 3 – Piconets e Scatternet. Fonte: TANENBAUM, Andrew S., 2003

O canal físico é dividido, pelo mestre do piconet, em slots de 625 μs cada um. Tanenbaum afirma que “os quadros podem ter 1, 3 ou 5 slots de duração”. Tanenbaum assevera que “a sincronização por saltos de freqüência permite um tempo de ajuste de 250 a 260 μs por salto, para permitir que os circuitos de rádio se estabilizem. Os dados (quadros) são transmitidos por um link que possui dois tipos. O ACL (Asynchronous Connectio-Less) e o SCO (Synchronous Connection Oriented). O ACL não é orientado a conexão e, portanto, sem garantia de entrega. Um escravo ter apenas um canal ACL com o mestre do piconet. O SCO é orientado a conexão. Aloca-se um slot fixo em cada sentido da comunicação. Pode haver até três canais SCO com o mestre.

O protocolo de adaptação de controle de enlace lógico ou a camada L2CAP trabalha com a multiplexação e demultiplexação dos pacotes (os quadros são divididos em pacotes pelo L2CAP). O L2CAP entrega o quadro ao protocolo apto a ler o pacote no dispositivo receptor. Tanenbaum afirma que L2CAP também “lida com os requisitos de qualidade de serviço, tanto quando os enlaces são estabelecidos quando durante a operação normal. Também é negociado em tempo de configuração de tamanho máximo de carga útil permitido”.

A figura 3.4 mostra a estrutura de um quadro Bluetooth.

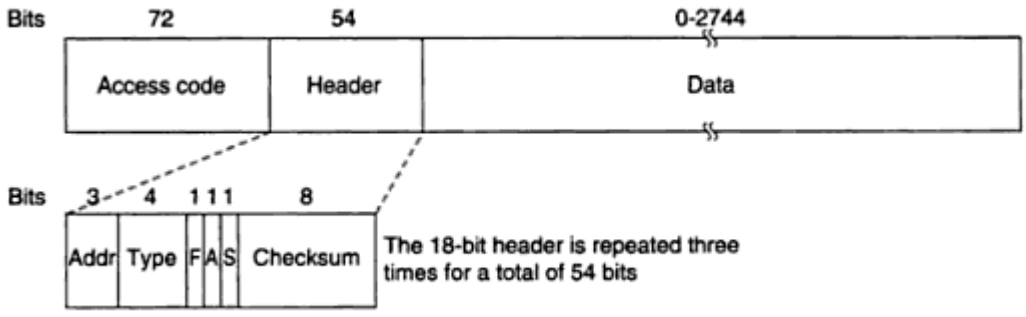

Figura 3. 4 – Estrutura de um quadro Bluetooth. Fonte: TANENBAUM, Andrew S., 2003

O dispositivo usado é o “Bluetooth serial converter UART interface 19200 Bps” da Sure Eletreonic’s. Este conversor é usado para converter um UART (Universal Asynchronous Receiver Transmitter) com 19200 Bps, um star bit, 8 bits de dados, um stop bit, nenhum formato de bit de paridade para o protocolo Bluetooth UART. A figura 3.5 ilustra um exemplo rede Bluetooth:

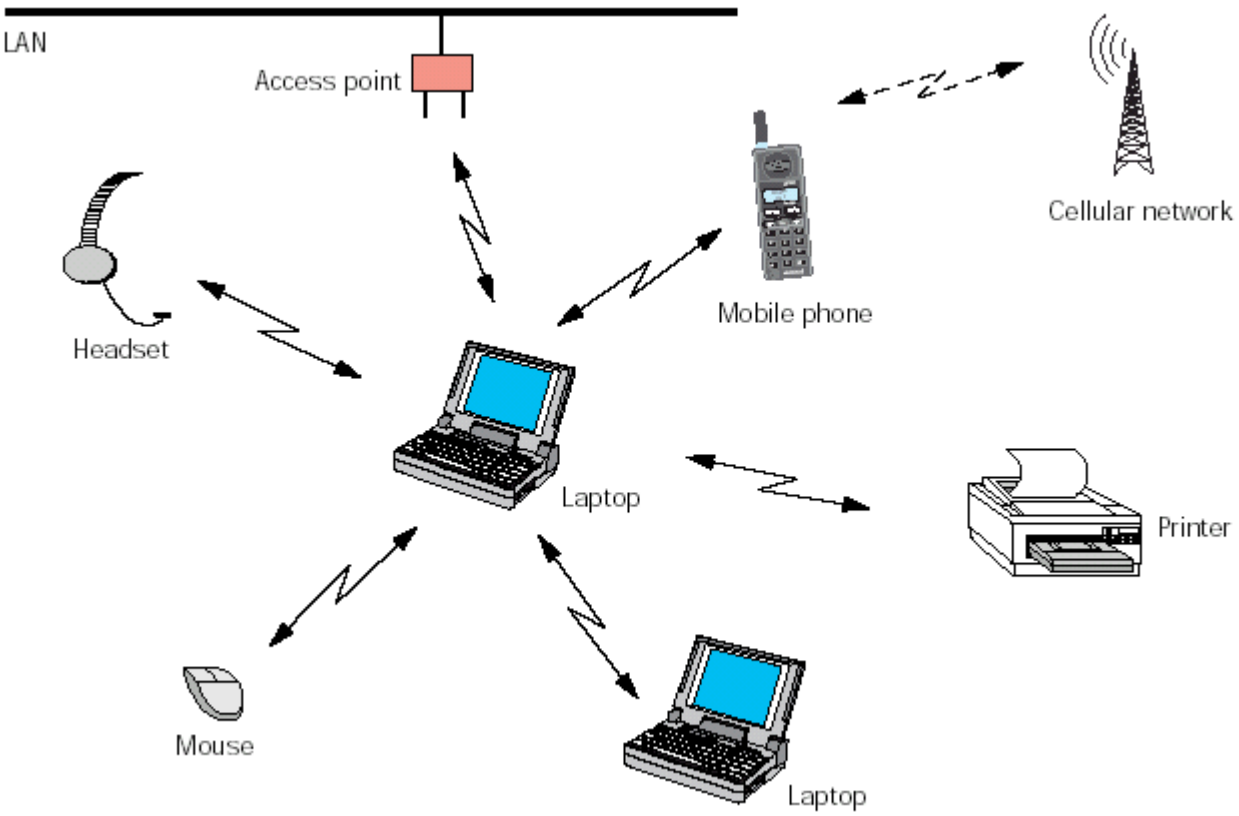

Figura 3. 5 – Exemplo de uma rede Bluetooth. Fonte: http://www.gta.ufrj.br/grad/02_1/bluetooh/introducao.htm

3.2.2 Microcontroladores

Microcontroladores são chips usados em muitas implementações de equipamentos eletrônicos. Há muitos usos para este tipo de tecnologia. São usados em geral na automação e controle de periféricos. Por seu baixo consumo, são usados em máquinas de lavar roupas, micro-ondas, controle remoto e muitos outros equipamentos. Podem-se controlar as saídas, tendo como referencia a entrada ou um software interno. Um microcontrolador é composto por um processador, memória e pinos com funções de entrada/saída. Existem vários tipos de microcontroladores. As principais diferenças entre esses tantos tipos são a memória interna (quantidade), o tempo de processamento (velocidade), quantidade de pinos entrada/saída (I/O). Os microcontroladores possuem a habilidade do modo de espera, ou seja, podem aguardar por uma interrupção, como por exemplo, o acionamento de uma tecla. O microcontrolador usado neste projeto é o microcontrolador PIC16F876A de 28 pinos composto por 22 portas de entrada/saída (I/O), memória de programa Flash com 8kW, memória EEPROM com 256 bytes, dois timers de 8 bits, um timer de 16 bits, Ustart, AD de 10 bits. A figura 3.6 mostra o microcontrolador usado neste projeto:

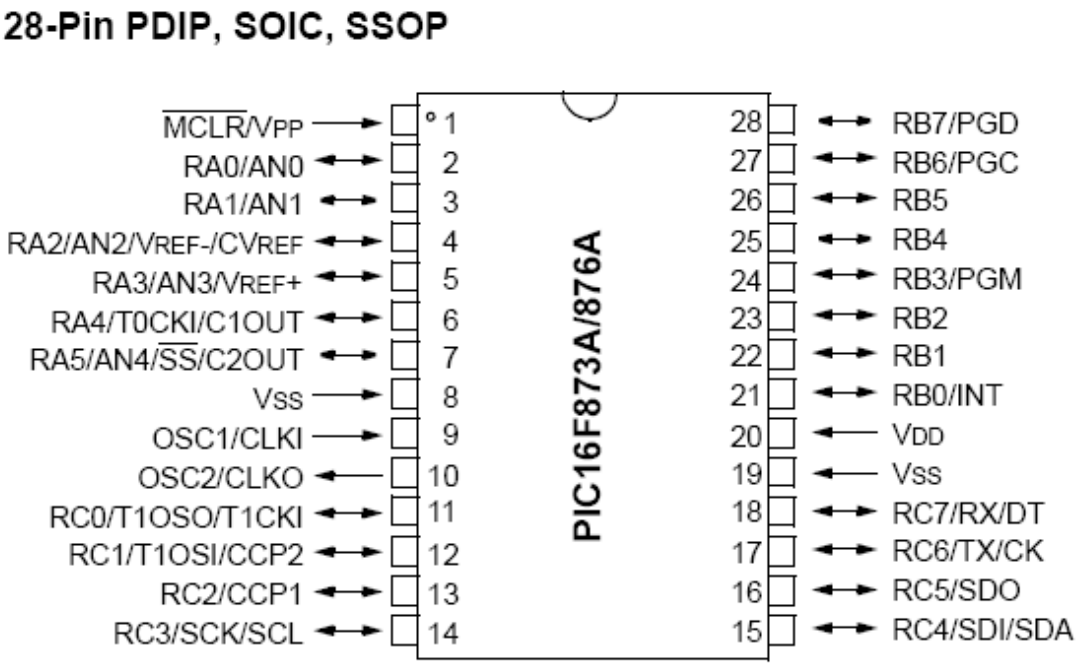

Figura 3. 6 – Microcontrolador PIC16F8763A. Fonte: http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf

3.2.3 Leitor de Códigos de Barras (Barcode Scanner)

Uma maneira de se controlar estoques é utilizar o controle por meio de códigos de barras. Códigos de barras impressos no produto identificam aquele produto. O equipamento utilizado para fazer a leitura desse código é o scanner de código de barras. Este equipamento funciona da seguinte maneira: o equipamento faz a leitura na superfície do produto onde se localizam as barras impressas por meio de um feixe de laser e transforma as informações impressas de barras em códigos binários. A figura 3.7 ilustra um exemplo desse equipamento.

Figura 3. 7 – Leitora ótica

3.2.4 Banco de dados

Nenhuma empresa, atualmente, sobrevive sem o histórico de informações geradas ao longo de sua existência. Os dados são armazenados em vários meios como, por exemplo: papéis, fitas magnéticas, fotografias etc. O estudo da computação permitiu a criação de sistemas que armazenam e gerenciam dados de forma ágil, eficiente e segura. São os sistemas de banco de dados.

Os sistemas de gerenciamento de bando de dados (DBMS – *Database Management System*) muitas informações. Esses sistemas, além de guardar informações de forma estruturada, também são projetados para garantir a segurança dos dados armazenados. Os dados armazenados podem ser compartilhado por vários usuários do sistema. Podem ser acessados por aplicativos de diferentes plataformas e, dependendo da interligação em redes, são distribuídos geograficamente pelo planeta.

São exemplos de aplicação do uso de um DBMS os bancos, linhas aéreas, indústria, departamento de recursos humanos e neste projeto, será usado para armazenar os dados coletados pela caneta ótica.

Visão dos dados

Um sistema de banco de dados é uma coleção de dados inter-relacionados e um conjunto de programas que permite aos usuários acessar e modificar esses dados. (SILBERSCHATZ, KORTH, SUDARSHAN, 2006)

Composição de um DBMS

Um DBMS é composto por instâncias e esquemas, modelos de dados, linguagens de banco de dados, banco de dados relacionais, tabelas. SILBERSCHATZ, KORTH, SUDARSHAN as definem assim:

Instâncias e esquemas – as instâncias são informações armazenadas em um dado momento no banco de dados (os dados são inseridos e excluídos a todo tempo). O projeto do banco de dados é chamado de esquema e dificilmente é alterado.

Modelos de dados – o modelo de dados é uma coleção de ferramentas conceituais. Descrevem dados, relação de dados, semântica de dados tanto no nível físico quando no lógico. Os modelos podem ser classificados em quatro diferentes categorias:

- Modelo relacional – é um modelo que se baseia em registros. Cada tabela possui registros específicos. Cada registro define um número fixo de campos. As colunas da tabela ficam os atributos dos registros.
- Modelo de entidade/relacionamento – é baseado em uma percepção de um mundo real que consistem em uma coleção de objetos básicos, chamados entidades, e as relações entre esses objetos. Uma entidade é uma “coisa” no mundo real que é distinguível dos outros objetos.
- Modelo de dados baseado em objeto – ele pode ser visto como uma extensão ao mundo E/R com noções de encapsulamento, métodos (funções) e identidade do objeto. O modelo de dados relacional de objeto combina recursos do modelo de dados orientado a objetos e do modelo de dados relacional.
- Modelo de dados semi-estruturado – permite a especificação dos dados em que itens de dados individuais do mesmo tipo possam ter diferentes conjuntos de atributos. Isso é o oposto dos modelos de dados mencionados anteriormente, em que todos os itens de dados de um determinado tipo precisam ter o mesmo conjunto de atributos.

Linguagem de banco de dados – na teoria trabalha-se com uma “linguagem de manipulação de dados” e outra de “definição de dados”. Na prática, essas linguagens são parte de um todo chamado linguagem SQL.

Linguagem de definição de dados (LDD*)*: a linguagem de definição de dados (LDD), assim como qualquer linguagem de programação, tem como entrada algumas instruções. Transforma essas entradas em saídas. A especificação de um esquema de banco de dados é feita pela chamada LDD.

Linguagem de manipulação de dados (LMD): a LMD permite o acesso e manipulação dos dados. Temos como exemplo de acessos:

- Recuperação de informações;
- Inserções de informações
- Exclusão de informações
- Modificações.

Existem dois tipos de LMD:

- LMDs procedurais - necessita que um usuário do sistema especifique quais dados são necessários e como recuperá-los.
- LMDs declarativas – necessita que um usuário especifique quais dados são necessários sem especificar como recuperá-los.

Banco de dados relacionais

Baseia-se no modelo relacional. As tabelas são usadas para representar os dados e a relação entre elas. A LDD e a LMD também são usadas.

Tabelas

As tabelas são usadas para armazenar os dados de forma organizada. Elas permitem interação com outras tabelas. Essa interação é chamada de relacionamento de tabelas ou banco de dados relacional. Geralmente possuem um identificador

invariável que identifica a tabela, chamado de chave primária (*Primary Key - PK*). Cada tabela possui linhas e colinas que conterão as informações. A figura 3.8 mostra um exemplo de tabela criada no software de banco de dados do Microsoft Access 2003 usado neste trabalho. No Microsoft Access 2003, a chave primária é identificado por um símbolo de chave a esquerda do capo considerado identificador da tabela:

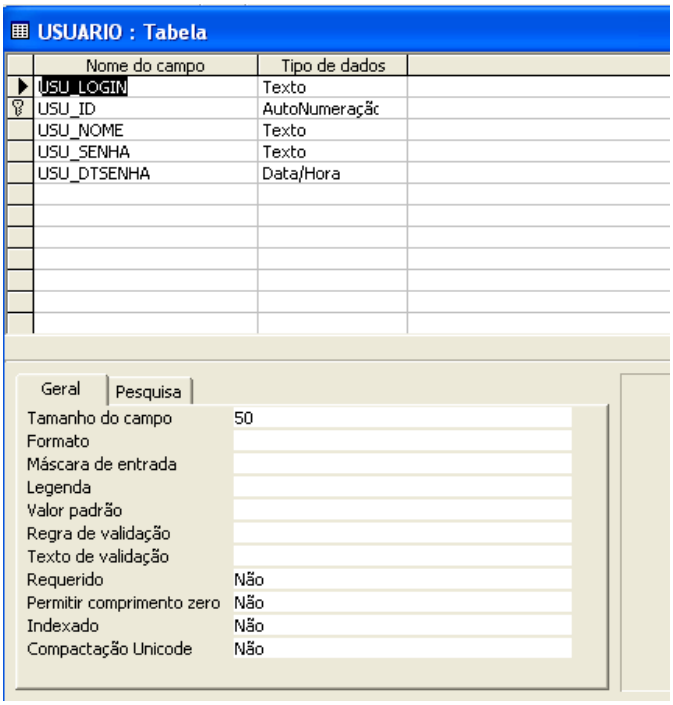

Figura 3. 8 – Figura que ilustra a chave primária da tabela

CAPÍTULO 4. PROPOSTA DE SOLUÇÃO DE MODELO

O projeto está dividido em duas partes:

- A etapa 1 envolve a construção do dispositivo que terá a caneta eletrônica acoplada. Essa etapa é a de confecção do hardware que o cliente do supermercado irá utilizar no carrinho de compras.
- A etapa 2 que envolve o software usado pelo caixa e o banco de dados no servidor. Também inclui nesta etapa, o conector Bluetooth no servidor para a recepção dos dados.

4.1 Etapa 1 – Construção do Hardware

Para criar o hardware ao qual a caneta será acoplada, foi utilizado os seguintes componentes:

- Kit Cerne PICLAB16F876A com gravador e microcontrolador PIC16f876A;
- LCD de quatro linhas
- Software MikroBasic Pro versão 1.45 Trial
- Scanner de código de barras comum

A figura 4.1 ilustra a ligação elétrica do microcontrolador:

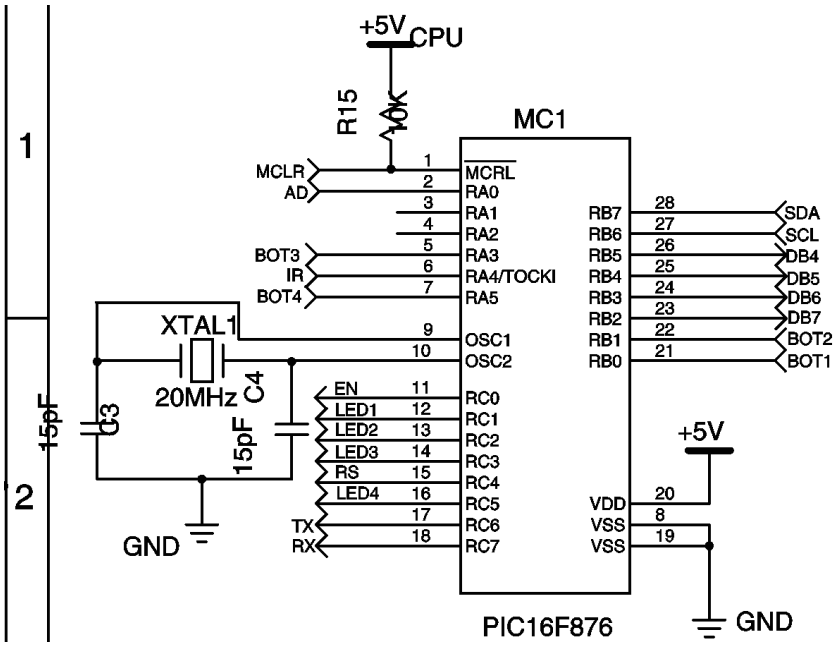

Figura 4. 1 – Esquema elétrico do microcontrolador PIC16F786A. Fonte: Tutorial que acompanha o kit Cerne Bluetooth.

A figura 4.2 ilustra a ligação elétrica dos botões:

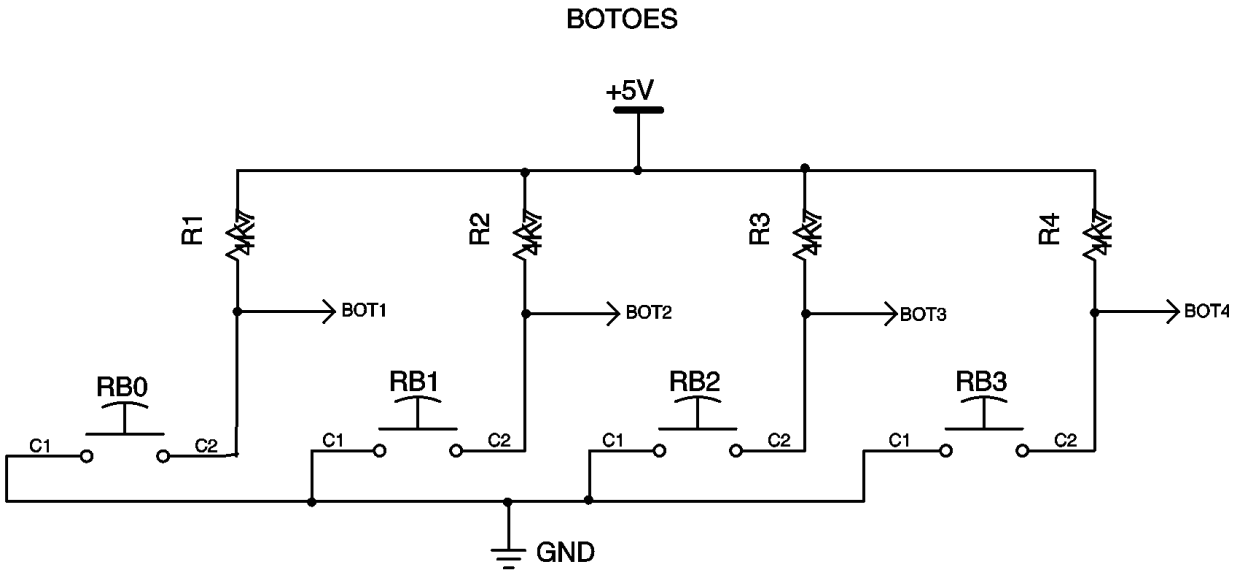

Figura 4. 2 – Esquema elétrico dos botões do kit. Fonte: Tutorial que acompanha o kit Cerne Bluetooth.

A figura 4.3 ilustra a ligação elétrica dos LEDs

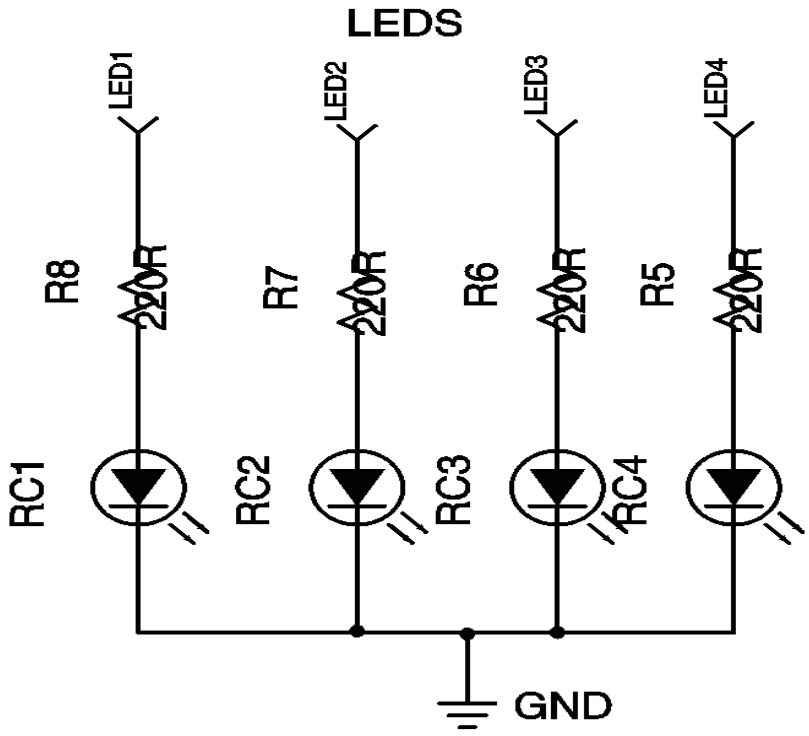

Figura 4. 3 – Esquema elétrico dos LEDs do kit. Fonte: Tutorial que acompanha o kit Cerne Bluetooth.

A figura 4.4 ilustra a ligação elétrica do Display LCD:

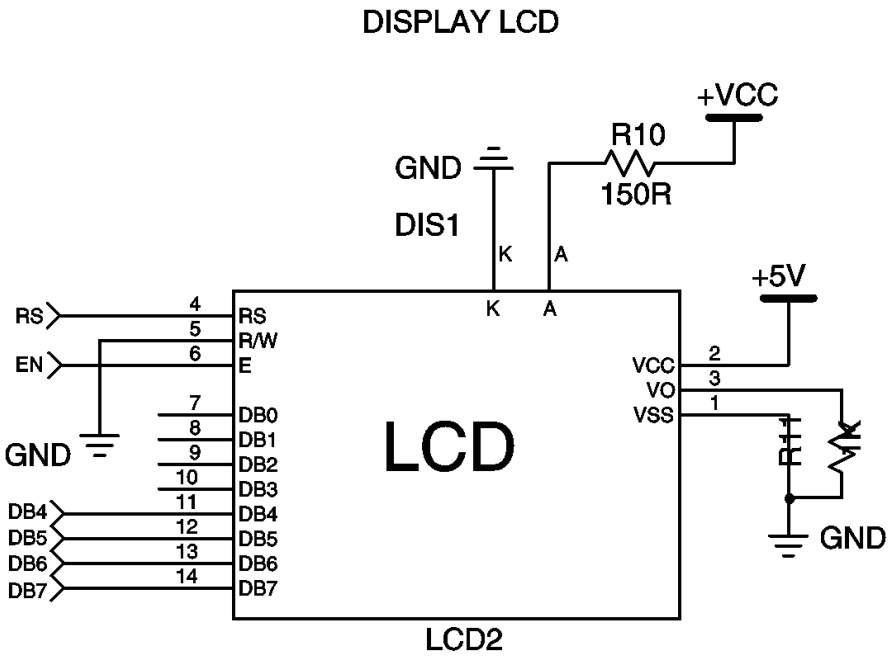

Figura 4. 4 - Esquema elétrico do display LCD do kit. Fonte: Tutorial que acompanha o kit Cerne Bluetooth.

A figura 4.5 ilustra a ligação elétrica do Módulo Bluetooth:

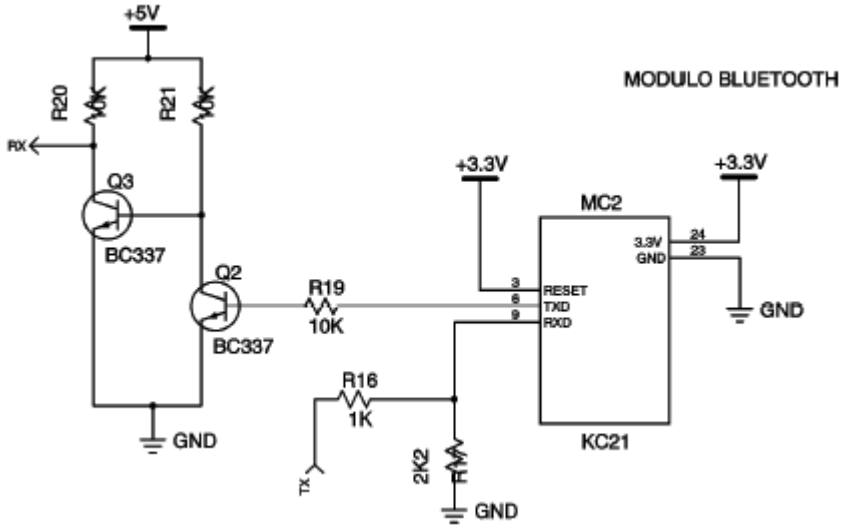

Figura 4. 5 - Esquema elétrico do módulo Bluetooth. Fonte: Tutorial que acompanha o kit Cerne Bluetooth.

A figura 4.6 ilustra a ligação resumida da placa:

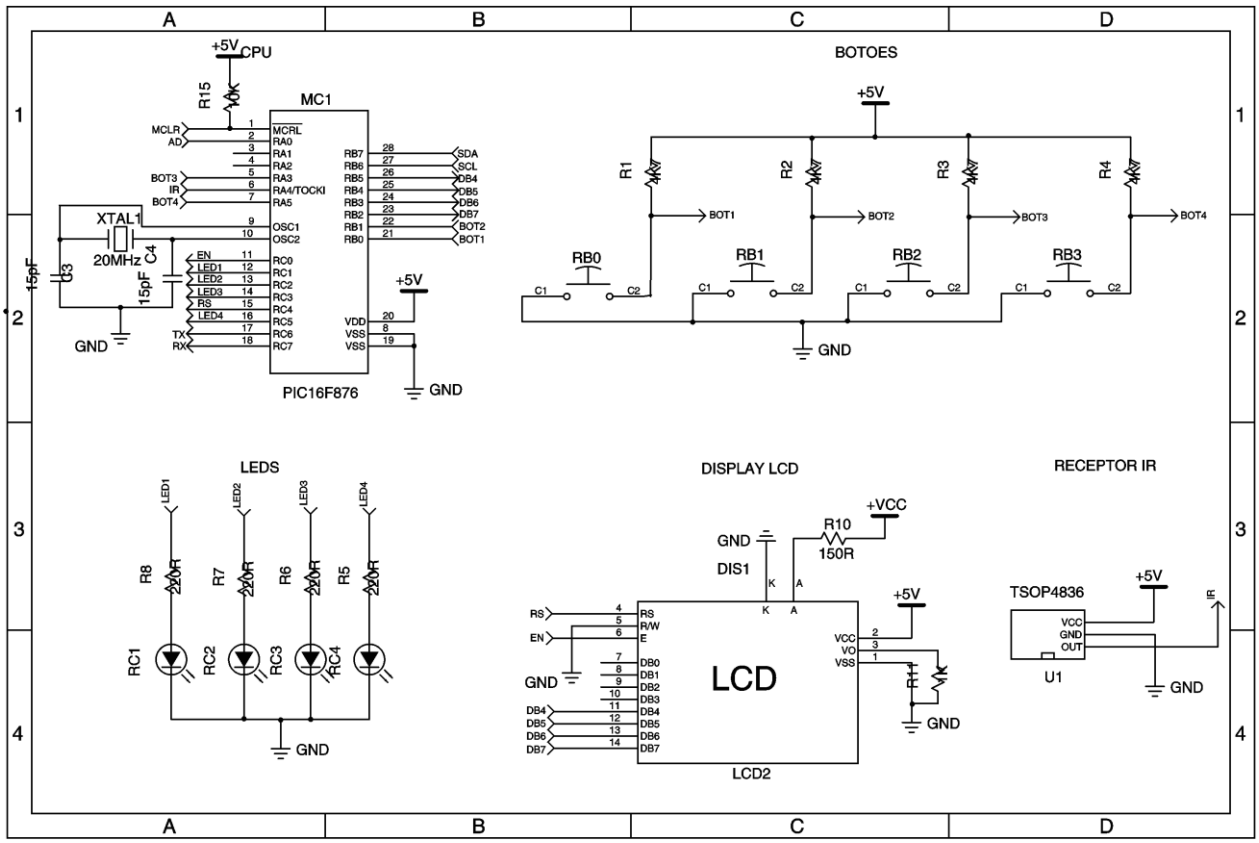

Figura 4. 6 - Esquema elétrico da placa. Fonte: Tutorial que acompanha o kit Cerne Bluetooth.

Foram feitas algumas modificações no circuito apresentado.

O receptor IR (Infrared – infravermelho) foi retirado do circuito. Não será usado no projeto.

A porta do microcontrolador RA0 (AD – analógico digital) foi retirada e substituída pelo conector PS/2. Pela proximidade, a outra porta do conector PS/2 foi conectada na porta RA1 do microcontrolador. O PS/2 necessita de duas portas.

Como mostram as figuras, o circuito apresenta-se da seguinte forma:

- O conector PS/2, que conecta a caneta ótica, usa as portas RA0 e RA1 do microcontrolador.
- O display LCD está conectado nas portas RB2(DB7), RB3(DB6), RB4(DB5) e RB5(DB4) do microcontrolador.
- O Bluetooth está conectado nas portas RC6(TX) e RC7(RX) do microcontrolador.
- Os botões do kit estão ligados nas portas RB0(botão 1), RB1(botão 2), RA3(botão 3) e RA4(botão 4).

O código criado no compilador mickrobasic para controlar as portas do microcontrolador é listado no apêndice I.

4.1.1 Detalhamento do código fonte do hardware:

Nesta parte do código a seguir, são definidas as portas usadas pelo LCD. As variáveis que identificam os pinos foram escolhidas de forma a serem iguais aos nomes identificados no circuito elétrico.

```
dim LCD_RS as sbit at RC4_bit
LCD_EN as sbit at RC0_bit
LCD_WR as sbit at RC3_bit

LCD_D4 as sbit at RB5_bit
LCD_D5 as sbit at RB4_bit
```

```
LCD_D6 as sbit at RB3_bit

LCD_D7 as sbit at RB2_bit
```

No código abaixo, as portas do LCD são definidas para ter a transmissão bidirecional.

```
LCD_RS_Direction as sbit at TRISC4_bit

LCD_EN_Direction as sbit at TRISC0_bit

LCD_WR_Direction as sbit at TRISC3_bit

LCD_D4_Direction as sbit at TRISB5_bit

LCD_D5_Direction as sbit at TRISB4_bit

LCD_D6_Direction as sbit at TRISB3_bit

LCD_D7_Direction as sbit at TRISB2_bit
```

Nesta parte do código, são definidas as portas do PS/2 e configuradas para ter a transmissão bidirecional.

```
dim PS2_Data as sbit at RA1_bit

PS2_Clock as sbit at RA0_bit


PS2_Data_Direction  as sbit at TRISA1_bit

PS2_Clock_Direction as sbit at TRISA0_bit
```

Neste código, declaração da variável que receberá os dados na Bluetooth.

```
dim byte_rx as byte
```

Neste código, declaração da variável que receberá os dados da caneta ótica.

```
dim keydata  as byte
```

Este código define uma variável que é usada para comparar os dados lidos pela caneta. Essa comparação visa descartar os códigos de barras lidos duas vezes em seguida.

```
dim keydataAnt  as byte
```

Nesta linha, é declarado uma variável caso o sistema tenha algum teclado com teclas especiais como F2, F10 etc.

```
dim special  as byte
```

Este código declara uma variável que será usada para verificar se o botão do kit está pressionado.

```
dim down  as byte
```

Definição de variável que guarda o texto a ser listado no display do kit ao usuário quando esse texto estiver com 16 caracteres.

```
dim txt  as string[17]
```

Variável que verifica o comprimento atual do texto.

```
dim conttxt as byte
```

Variável que define a quantidade máxima de linhas mostradas no display.

```
dim MAX_LINHA_LCD as byte
```

Variável que guarda a linha corrente do display.

```
dim LINHA_LCD as byte
```

Esta função define a rotina de interrupção. Os dados da porta serial são armazenados na variável *byte_rx*. Se o texto for que 17 caracteres, é armazenado na variável *txt* o caractere que está na variável *byte_rx*. A variável *conttxt* é acrescido de 1. Este procedimento é repetido até atingir que a variável *conttxt* tenha 17 de valor e a variável *txt* tenha 16 caracteres. Posteriormente, essa variável será enviada ao display por meio da função *MostrarNoDisplay*.

```
sub procedure interrupt
byte_rx = UART1_Read()
if conttxt < 17 then 'se menor que 17 vou colocando o byte em txt
strAppendSuf(txt, byte_rx)
conttxt = conttxt + 1
end if
end sub
```

Essa função faz o LED do botão ser ligado ao pressionar o botão. O teste é feito da seguinte forma. Se o botão não estiver pressionado (**if** PORTB.0=1) o LED não é aceso. Caso contrário, o LED é aceso.

```
sub procedure PiscaLED

if PORTB.0=1 then
        PORTC.1 = 0
else
        PORTC.1 = 1  'Acende o LED
end if

if PORTB.1=1 then
        PORTC.2 = 0
else
        PORTC.2 = 1
end if

if PORTA.3=1 then
        PORTC.3 = 0
else
        PORTC.3 = 1
end if

if PORTA.5=1 then
        PORTC.5 = 0
else
        PORTC.5 = 1
end if

end sub
```

Esta parte do código faz com que um comando seja enviado ao servidor e interpretado por ele, devolvendo a resposta ao display da caneta de acordo com o botão pressionado. Temos os seguintes comandos:

- NVG 0 – desce uma linha no display.
- NVG 1 – sobe uma linha no display.
- PSQ – confirma.
- CNC – cancela

A procedure chama a função VerificarBotoes para ver o botão pressionado. O código *UART1_Write_Text* envia pela porta serial(Bluetooth) o comando como argumento. O comando delay_ms(100) faz o sistema ter um atraso de 1s após o envio. A função PiscaLED é chamada para ligar o LED do botão.

```
sub procedure VerificarBotoes
        if (PORTB.0=0) then
        if down=1 then
        UART1_Write_Text ("NVG 0")
        delay_ms(100)
        down = 0
        end if
        else
        if PORTB.1=0 then
        if down=1 then
        UART1_Write_Text("NVG 1")
        delay_ms(100)
        down = 0
        end if
        else
        if PORTA.3=0 then
        if down=1 then
```

```
                UART1_Write_Text("PSQ")
                delay_ms(100)
                down = 0
        end if
        else
        if PORTA.5=0 then
                if down=1 then
                UART1_Write_Text("CNC")
                delay_ms(100)
                down = 0
                end if
        else
                down = 1
        end if
        end if
        end if
        end if
        PiscaLED
end sub
```

Nesta parte, é apresentada a rotina para mostrar os textos no display. É verificado em qual linha é possível imprimir no display. A variável que informa em qual linha é possível imprimir é a LINHA_LCD. Caso seja a linha 3, o comando é para imprimir na linha 1 e posição 17, que é na terceira linha. Se é linha 4, é impresso na linha 2 e posição 17, que é a linha 4. Ao final, a variável LINHA_LCD é acrescida de 1. É feito um teste para ver já mostrou as quatro linhas. Se sim, a variável LINHA_LCD é novamente iniciada a posição 1.

```
sub procedure MostrarNoDisplay()
if LINHA_LCD= 1 then
Lcd_Out(1,1,txt)
end if
if LINHA_LCD= 2 then
```

```
        Lcd_Out(2,1,txt)
end if
if LINHA_LCD= 3 then
        Lcd_Out(1,17,txt)
end if
if LINHA_LCD= 4 then
        Lcd_Out(2,17,txt)
end if
LINHA_LCD = LINHA_LCD + 1
if LINHA_LCD > MAX_LINHA_LCD then
        LINHA_LCD = 1
end if
end sub
```

Aqui segue a rotina principal do código. Nesta parte do código é feito a inicialização das interrupções. Estes comandos são definidos na documentação (tutorial) do microcontrolador.

```
INTCON.PEIE = 1
INTCON.GIE = 1
PIE1.RCIE = 1
PIE2.TXIE = 0
```

Este comando inicializa a interrupção da serial (Bluetooth).

```
UART1_Init(115200)
```

Esta parte do código inicializa o módulo Bluetooth. Estes comandos são enviados ao servidor que os interpreta.

```
delay_ms(2000)

UART1_Write_Text("AT+ZV DefaultLocal")
```

```
UART1_Write_Text("CCI - Carrinho de Compras Interativo")
UART1_Write(13)
UART1_Write(10)
delay_ms(2000)
UART1_Write_Text("AT+ZV EnableBond")
UART1_Write(13)
UART1_Write(10)
delay_ms(2000)
```

Estes códigos inicializam o display. O comando Lcd_Cmd(_LCD_CURSOR_OFF) é usado para desligar o cursor do display.

```
Lcd_Init()
Lcd_Out(1,1,"   CCI - v3.0 ")
Lcd_Out(2,1," Trabalho Final ")
Lcd_Out(1,17,"  Regis Levino  ")
Lcd_Out(2,17,"               ")
Lcd_Cmd(_LCD_CURSOR_OFF)
```

Estes códigos inicializam a comunicação da caneta ótica. A variável *txt* é inicializada com nenhuma informação.

```
Ps2_Config()
Delay_ms(100)
txt = ""
```

Esta parte do código é a parte principal. A caneta lê o código de barras. O comando **if** Ps2_Key_Read(keydata, special, down) **then** testa se o dado foi lido corretamente. Se sim, é verificado se o dado é um dado comum e não especial por meio do comando **if** (down <> 0) **and** (special = 0) **and** (keydata <> 0) **then**. Se sim, verifica-se se o dado não foi lido repetidamente por meio do comando **if** keydata <> keydataAnt **then**. Se não está repetido, o dado é impresso no display e enviado ao servidor por meio do comando UART1_Write(keydata). A variável keydataAnt recebe o dado lido. A rotina principal é mostrada abaixo:

```
while TRUE
if Ps2_Key_Read(keydata, special, down) then      ' Se o dado foi lido do PS/2
if (down <> 0) and (special = 0) and (keydata <> 0) then   ' Testa se é uma tecla comum
        if keydata <> keydataAnt then
        Lcd_Out(2,17, "Codigo:       ")
        Lcd_Out(2,25, keydata)
        keydataAnt=keydata
        UART1_Write(keydata)
        else
        keydataAnt = 0
        end if
end if
else
if conttxt = 17 then  'Só coloco no display string com 16 caracteres recebidos pela serial
MostrarNoDisplay()
conttxt = 1
txt = ""
else
VerificarBotoes()
end if
end if
Wend
```

A figura 4.7 mostra o programa usado no projeto para a construção do código que gerencia o microcontrolador.

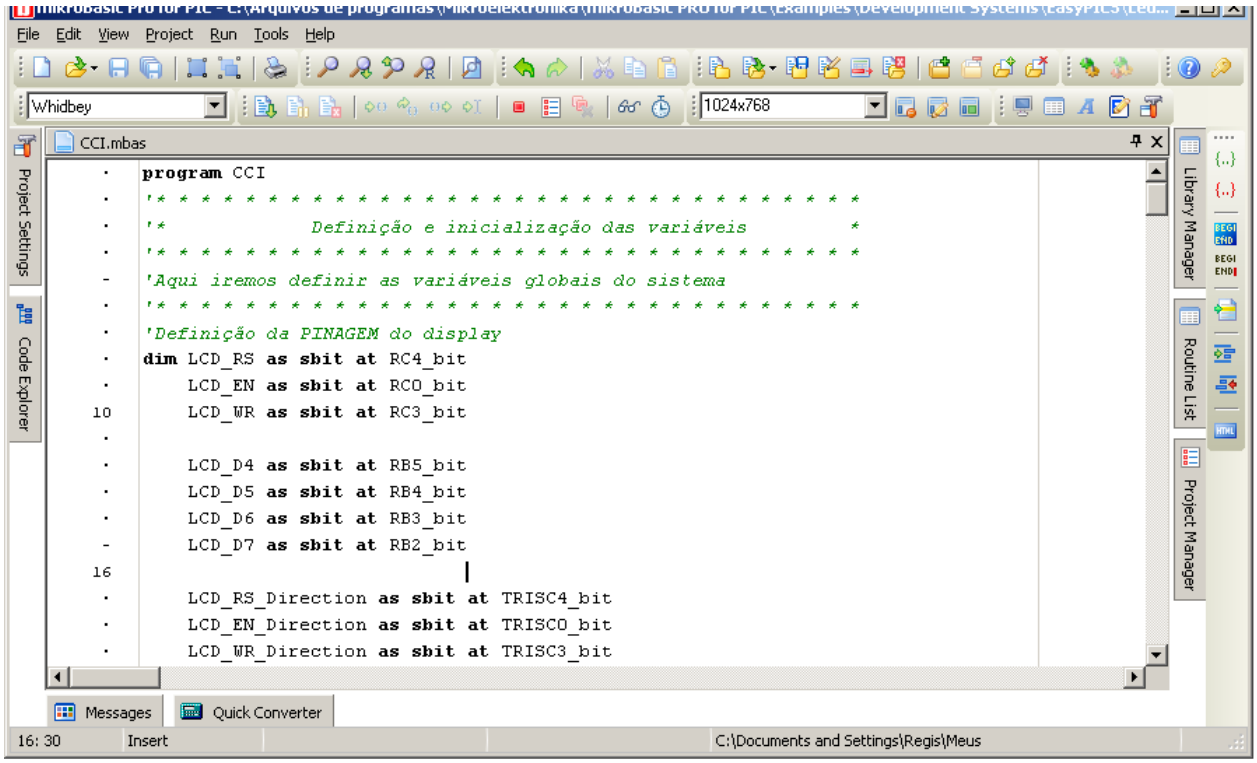

Figura 4. 7 – Código no software MickoBasic.

A figura 4.8 mostra o kit usado no projeto.

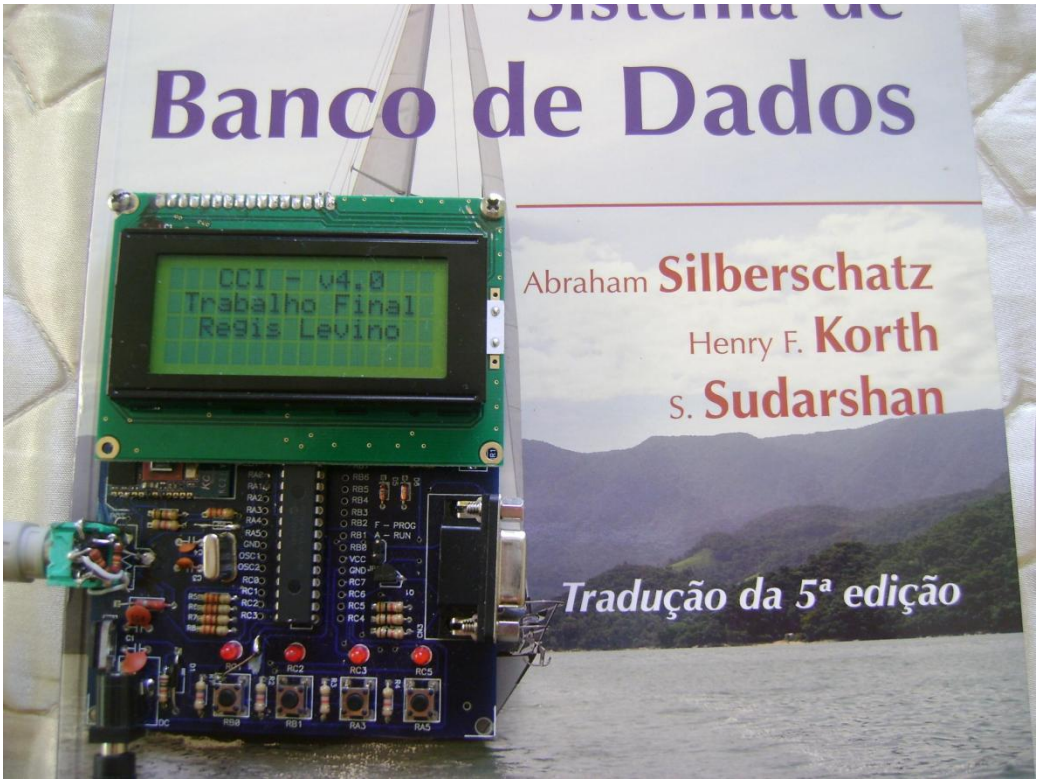

Figura 4. 8 – Kit Cerne Bluetooth

A figura 4.9 mostra a caneta ótica

Figura 4. 9 – Leitora ótica.

4.2 Etapa 2 - Criação do servidor

4.2.1 Criação do banco de dados

O banco de dados usado no projeto é um banco de dados em Access 20003. Será explicado como foi criado o banco e sua integração com o Delphi. Para criar o banco de dados do servidor, realizar os seguintes passos. Abrir o ACCESS e criar um banco de dados vazio, como mostra a figura 4.10:

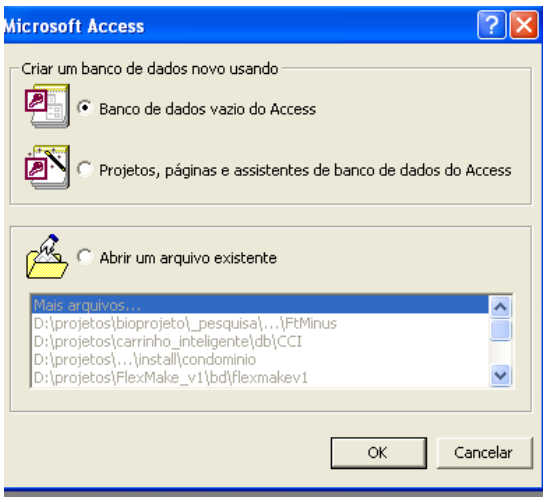

Figura 4. 10 – Criar banco de dados

Após salvar o novo projeto de banco de dados, deve-se escolher a opção “Criar tabela no modo estrutura”. A figura 4.11 mostra esse passo.

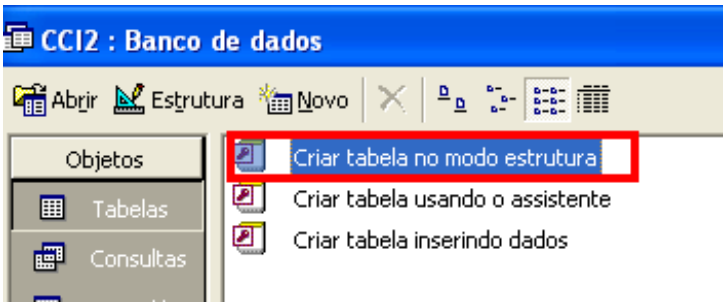

Figura 4. 11 – Tabela no modo estrutura

A primeira tabela chama-se USUÁRIOS e terá os campos como mostra a figura 4.12.

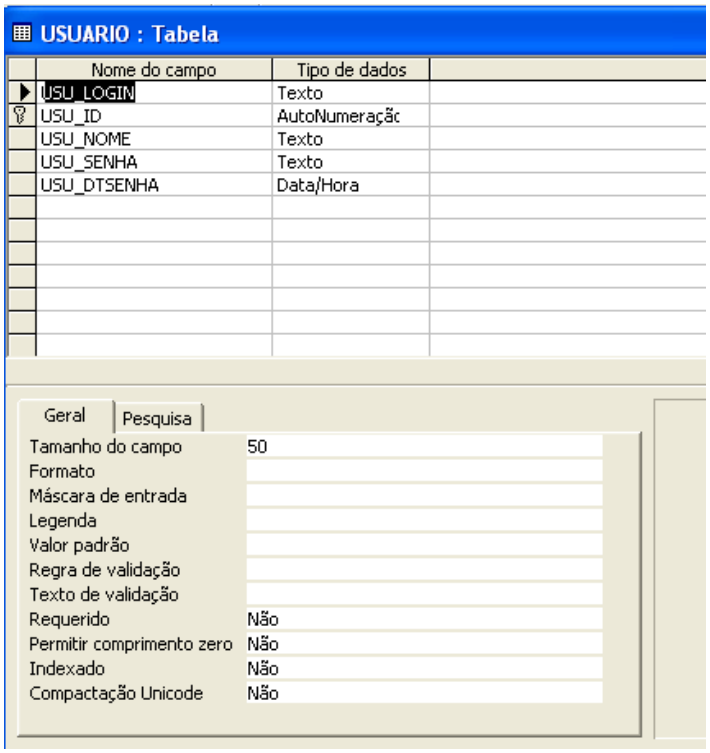

Figura 4. 12 – Tabela Usuários

Foi definido um índice para a tabela. É recomendável que toda tabela tenha um índice. O índice permite agilizar a pesquisa. Ele torna o campo único e garante que não terá duplicidade. A figura 4.13 ilustra o índice da tabela USUÁRIO.

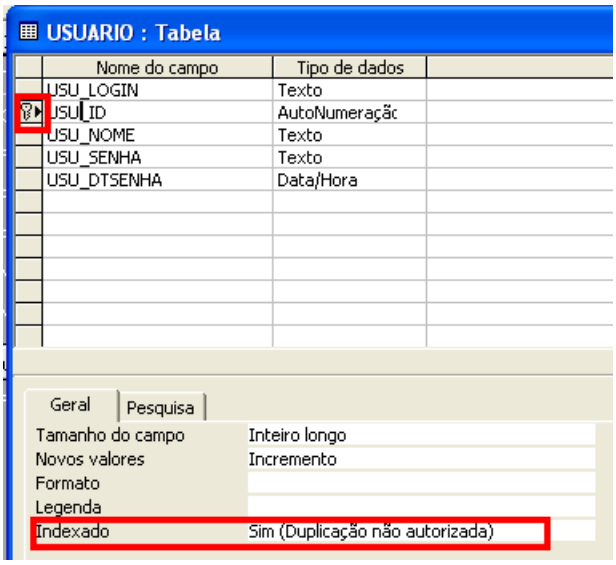

Figura 4. 13 – Definição da chave primária

Os passos anteriores foram seguidos para criar as demais tabelas. As tabelas do banco são listadas na figura 4.14.

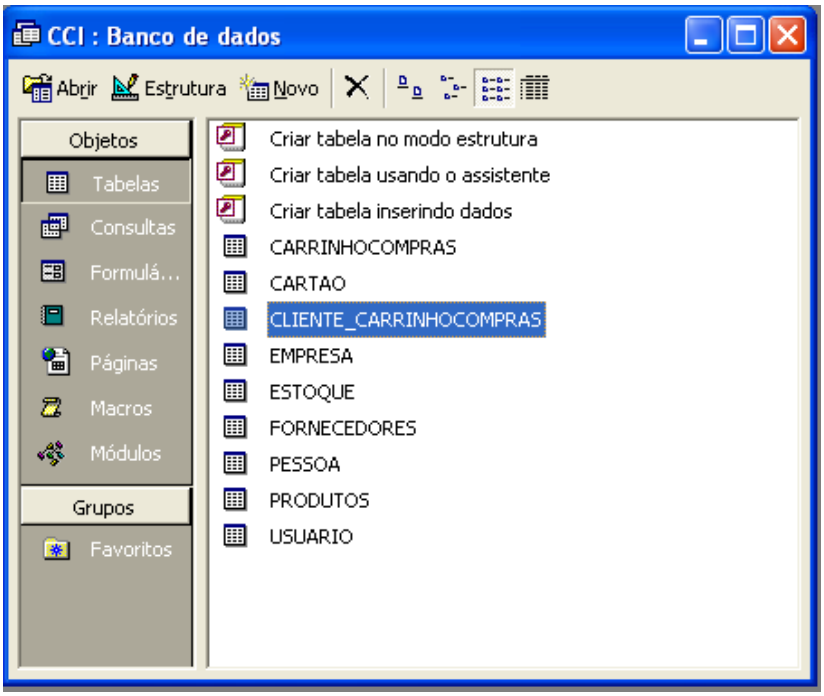

Figura 4. 14 – Tabelas do banco de dados

4.2.2 Conexão do banco com o Delphi

Para conectar o Delphi ao Acces é necessário criar um DataModule no Delphi. O DataModule é o repositório do Delphi responsável pelas rotinas de interação com o banco de dados assim como os componentes não visuais que interage com o banco de dados. Existem vários componentes no Delphi para interação com banco de dados. Alguns interagem com qualquer banco de dados e outros específicos para um tipo de banco de dados apenas.

Criando o DataModule. Abrir o Delphi e seguir os procedimentos como mostra a figura 4.15.

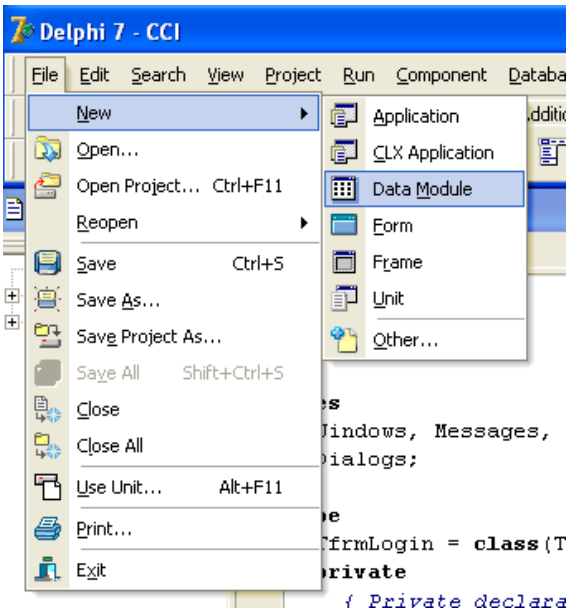

Figura 4. 15 – Criação do DataModule

Clicar no DataModule e depois pressionar a tecla F11 para que o ObjectInspector possa aparecer. No object inspector, mude o nome para Data. Clicar em SALVAR. Salve-o com o nome untData. Componente que interage com o banco de dados

Access. Na aba ADO, clicar no componente ADOConnection. A figura 4.16 ilustra o ícone.

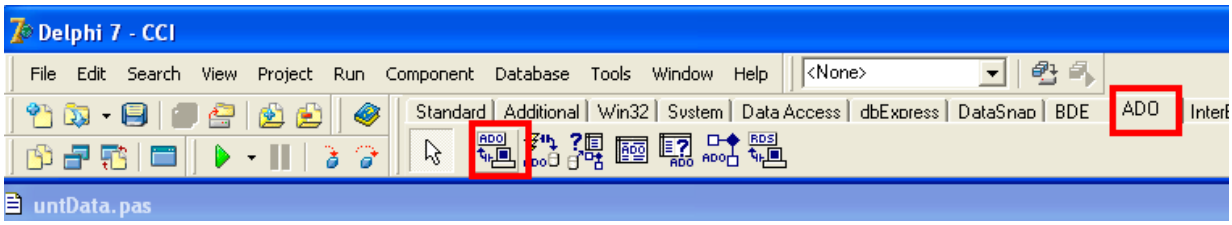

Figura 4. 16 – ADOConnection

Dar duplo-clique para aparecer dentro do DataModule ou clique e arraste-o. Mudar o nome do componente para DBADO. Para conectar o componente de conexão ao banco de dados CCI.mdb criado no ACCESS, seguir o procedimento mostrado na figura abaixo. clicar no botão a direita da propriedade ConnectionString. A figura 4.17 ilustra o campo.

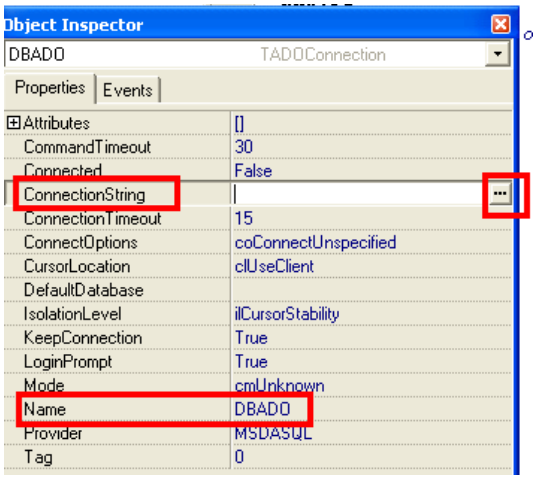

Figura 4. 17 – Conexão com o banco de dados

Clicar no botão BUILD. Selecionar o provedor para os bancos de dados do MICROSOFT (ACCESS). O provedor do Access 2003 é o provedor ilustrado na figura 4.18.

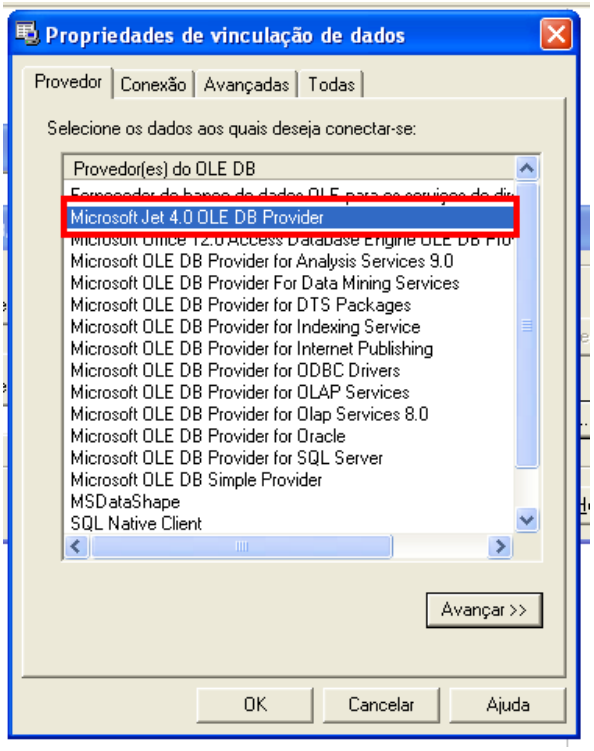

Figura 4. 18 – Conexão com o banco de dados

Adicionar o caminho do banco de dados criado no campo ilustrado na figura 4.19.

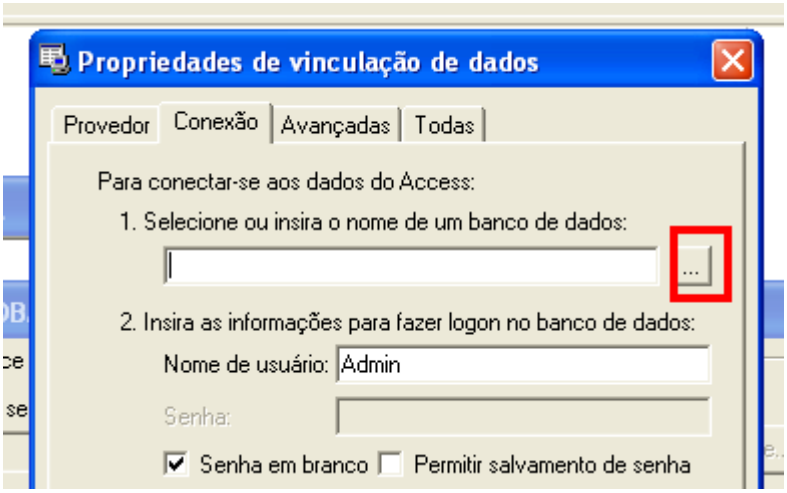

Figura 4. 19 – Conexão com o banco de dados

Clicar na aba ADO e escolher a opção mostrada na figura 4.20:

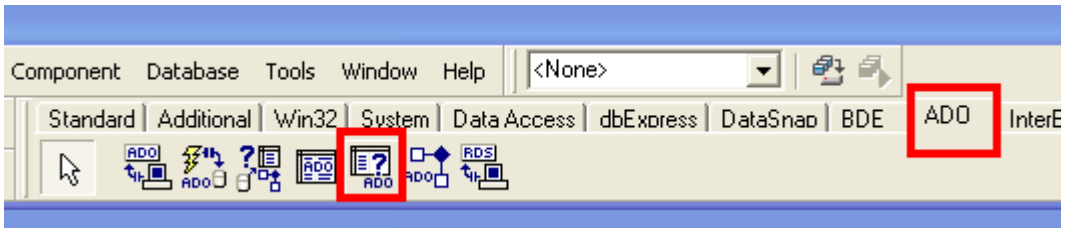

Figura 4. 20 – Estrutura de pesquisa

Este é o componente ADOQuery. É ele que permite fazer CONSULTAS no banco de dados. Banco de dados do tipo ACCESS ou SQL Server. Para outros tipos de bancos de dados como o FIREBIRD, INTERBASE, ORACLE, existem outros tipos de CONEXAO e CONSULTAS.

Mudar a nome para qryAux. Este componente não vai ter um nome especifico porque ele vai interagir com varias tabelas. Para que o componente possa trazer dados, é necessário que ele se conecte com o ADOConnection, que por sua vez se conecta ao banco de dados. Para conectar o CONSULTA ao PROVIDER, deve-se clicar na

propriedade CONNECTION e selecionar o ADOConnection DBADO (como temos apenas um, somente ele vai aparecer).

Povoando o DataModule. Mudar a propriedade LOGINPROMPT para FALSO. O objetivo desta mudança é para o componente não solicitar o prompt de LOGIN/SENHA todo momento que for executado uma transação no banco.

Clique na propriedade SQL. Aparecerá a tela mostrada na figura 4.21:

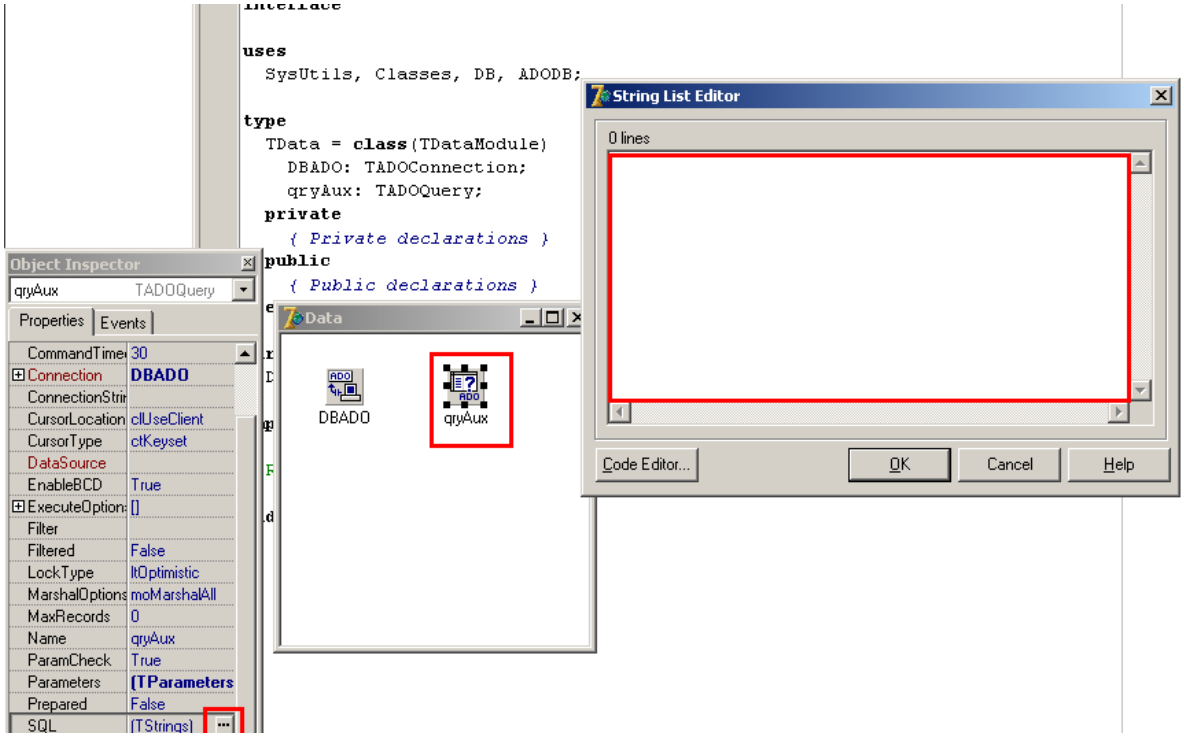

Figura 4. 21 – Criar consulta

Escreve-se a CONSULTA. Aproveitar para testar o componente. Digitar, dentro da área, o seguinte comando “SELECT * FROM PRODUTOS”. Este comando diz a tabela PRODUTOS do banco de dados CCI.mdb para retornar todos os produtos. O “*” diz para trazer todos os CAMPOS. Uma vez feito isso, vamos à propriedade ACTIVE da

qryAux e mudar para TRUE. Para que o processo se complete, é necessário de mais 3 componentes. Neste momento vamos trabalhar com mais 2. O primeiro é um PROVIDER. Este componente só permite que os dados trazidos pela nossa CONSULTA – qualquer que seja o banco de dados – seja enxergado pelo componente CLIENTDATASET.

Na aba Data Access do Delphi, colocar este componente no DataModule e renomear para dtsAux. A figura mostra o componente DataSetProvider renomeado para dtsAux e a sua propriedade DATASET direcionada para a QRYAUX. A figura 4.22 ilustra o campo DataSet.

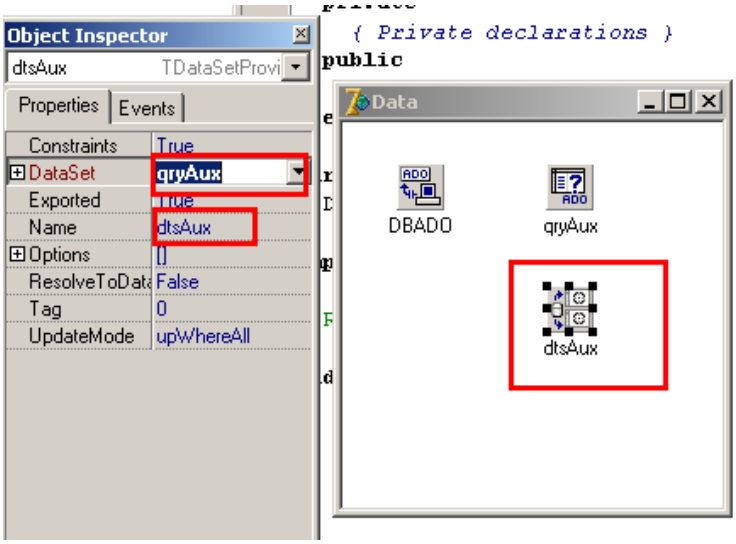

Figura 4. 22 – Criar consulta

Serve para ligar o componente QRYAUX ao próximo componente que será usado, que é um ClientDataSet -> (fornecedor de um conjunto de dados ao cliente).

Na aba Data Access (Acesso a dados), clicar no componente ClientDataAccess e colocar no DataModule.

Como mostra a figura, colocar o componente tipo TclientDataAccess. Mudar o nome para cdsAux e na propriedade PROVIDERNAME direcionar para o componente PROVIDER, que no caso deste projeto é o dtsAux. A figura 4.23 ilustra o ProviderName.

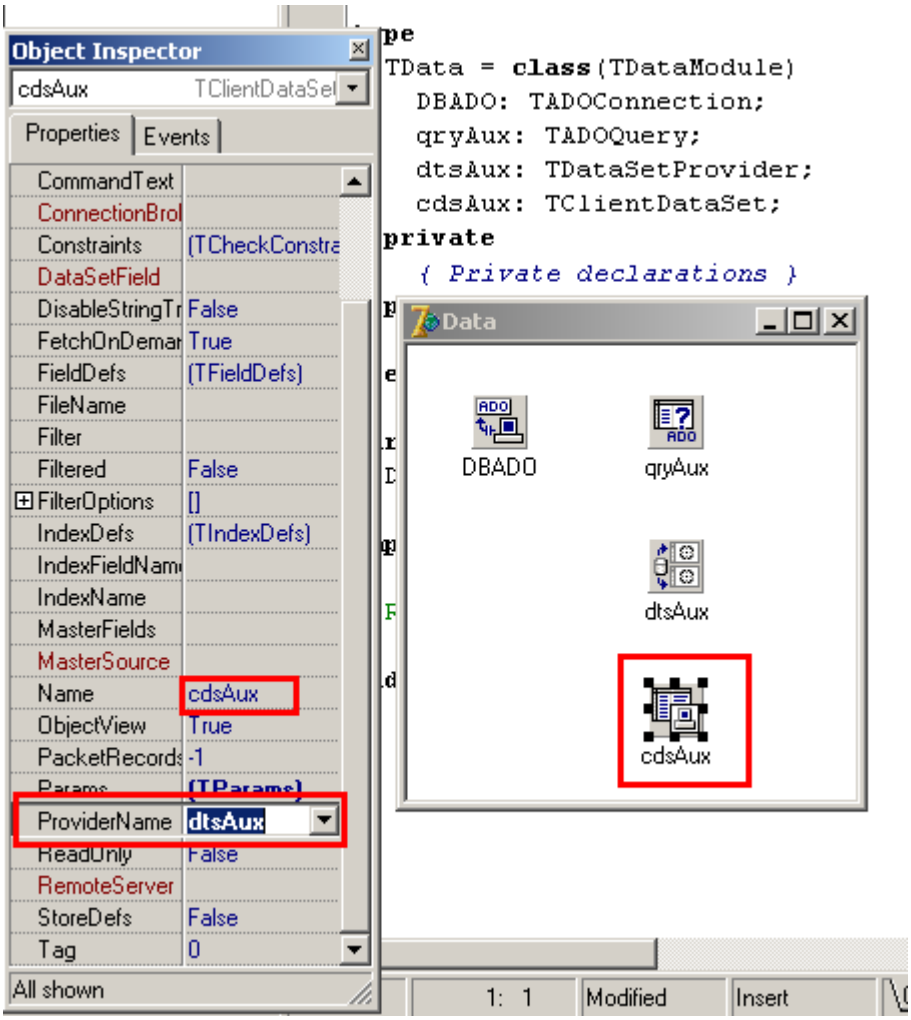

Figura 4. 23 – Provider Name

Isso finaliza a parte estrutural do DataModule. Funções principais do DataModule:

- **Função CartaoExiste**

1. Objetivo: Verificar se o cartão existe.
2. Parâmetros: MatriculaCartao – que é o número do cartão cadastrado.
3. Código-fonte: No apêndice IV

Explicação do código da Função CartaoExiste:

Fecha a consulta e o ClientDataSet para não “encavalar” dados de consultas anteriores.

```
qryAux.Close;
cdsAux.Close;
```

Na variável *aux* colocar o conteúdo do SQL a interagir no banco de dados.

```
aux := 'SELECT * FROM CARTAO WHERE CRT_MATRICULA=' + QuotedStr(MatriculaCartao);
```

Esse comando verifica se o cartão, cujo número está em *MatriculaCartao*, existe no banco de dados. Limpar a CONSULTA:

```
qryAux.SQL.Clear;
```

Colocar os COMANDOS SQL na consulta:

```
qryAux.SQL.Text := aux;
```

Executar a consulta (abrir):

cdsAux.Open;

Verificar se os dados foram retornados:

if cdsAux.IsEmpty then

Se o cartão existir no banco retorna VERDADEIRO:

result := true;

Se não existir retorno FALSO:

result := false

- **Função getDescricao**

1. Objetivo: Retorna a descrição do produto na tabela PRODUTOS.
2. Parâmetros de entrada: codigoBarras – código de barras do PRODUTO CADASTRADO.
3. Código-fonte: No apêndice IV

Explicação do código da função getDescricao:

O código tem quase os mesmo comandos da função CartaoExiste com algumas diferenças:

result := cdsAux.FieldByName('PDT_DESC').AsString;

Caso exista na tabela PRODUTOS o produto cujo código de barras está na variável codigoBarras, então a descrição deste produto é retornada 'PDT_DESC'.

Poderíamos colocar qualquer campo da tabela PRODUTOS, mas, neste caso, o que interessa é a descrição do produto. Poderíamos pegar qualquer um destes campos, como mostra a figura 4.24:

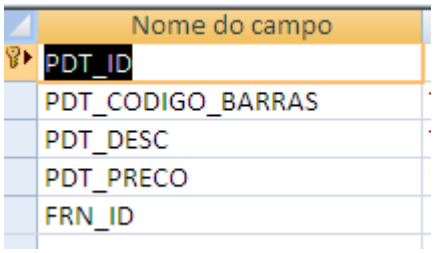

Figura 4. 24 – Tabela PRODUTOS

- **Função ExcluirProdutoDoCarrinho**

1. Objetivo: Excluir compra da tabela CARTAO_COMPRAS (Carrinho por assim dizer).
2. Parâmetros de entrada: CodigoDaCompra – Número do item que vai ser excluído.
3. Código-fonte: No apêndice IV

Explicação do código da função ExcluirProdutoDoCarrinho

Este código tem os procedimentos para excluir um item das compras. Todo o código é semelhante às funções anteriores. O cdsAux não está presente porque não existe dados a serem retornados. A única diferença é esta:

if qryAux.ExecSQL >0 then result := true;

Ou seja, quando o comando SQL é executado (neste caso para excluir o produto), o ExeSQL retorna maior que ZERO, caso alguma coluna da tabela

tenha sido AFETADA, e a função retorna TRUE. Caso contrário, retorna FALSE (FALSO).

- **Função AdicionarRegistroDeCompraNoBanco**

1. Objetivo: Adiciona na tabela COMPRAS_CARTAO o produto.
2. Parâmetros de entrada : CodigoCartao, codigoBarras , descricao, preco , Item: string; PagamentoEfetuado: boolean)
3. Código-fonte: No apêndice IV

Tratamento dos dados recebidos pelo servidor

Há cinco formas do kit da caneta ótica enviar informação ao servidor. O servidor recebe informações por meio de um dos quatro botões do kit ou por meio da porta serial (Bluetooth). A procedure TfrmBlueTooth.wclClientData(Sender: TObject; Buffer: Pointer; Size: Cardinal); faz o tratamento dos dados recebidos que foram enviados pelo kit. A procedure está no apêndice. Esta procedure verifica se foi algum botão ou dado vindo da PS/2 (caneta ou teclado). O botão RB0, ao ser pressionado, envia ao servidor o comando “NVG 1”. Este comando faz o display do kit navegar uma linha para baixo. O botão RB1, ao ser pressionado, envia o comando “NVG 0”. Este comando faz o display do kit navegar uma linha para cima. O botão RA3, ao ser pressionado, envia o comando “PSC”. Este comando ativa a função MontarPacoteTotalDasCompras que trás o valor total das compras e a quantidade de itens comprados. O botão RA5, ao ser pressionado, envia o comando “CNC”. Este comando exclui o item que está na linha 4

do display do kit. Se o comando veio da PS/2 (caneta ou teclado), a função TData.AdicionarProdutoNoCarrinho é chamada para inserir o dado no banco de dados.

CAPÍTULO 5. APLICAÇÃO DA SOLUÇÃO COM RESULTADOS

A solução deste projeto pode ser aplicada ao projeto anterior para economizar o tempo dos clientes nos caixas. O Objetivo é um dinamismo na comunicação entre cliente/servidor para que as compras sejam feitas no menor tempo possível agregando qualidade ao estabelecimento comercial.

Validação do Projeto

Para os testes, foram usados 10 produtos de diferentes características. Os dados são lidos, enviados e armazenados no servidor e apresentado no display do usuário como explicado no capítulo anterior.

Resultados obtidos

Com a implementação do servidor, o operador de caixa pode verificar diretamente do servidor as compras feitas pelo cliente. Os dados ficam armazenados no servidor para futuras consultas. O operador de caixa visualiza as compras efetuadas pelo cliente de forma ordenada e organizada.

Variáveis de impacto indireto no projeto

Academicamente falando, o projeto atende todos os requisitos de funcionamento. Para um melhor atendimento comercial, é necessário continuar aprimorando o software do operador de caixa e implementar melhorias no hardware de leitura dos produtos para que se tornem o mais fácil possível para os usuários.

CAPÍTULO 6. CONCLUSÃO

O projeto anterior visava, como seu principal objetivo, reduzir o tempo de espera nas filas dos caixas de supermercados. O dispositivo criado no projeto anterior, academicamente, atendia o objetivo. Alguns itens precisavam ser melhorados para melhor atender os objetivos. Algumas dessas melhorias foram concebidas neste projeto.

Com a construção deste projeto, os clientes agora passam a contar com uma tecnologia sem fio, que transmite os dados lidos dos produtos diretamente a um servidor. Ao chegar ao caixa, o operador de caixa pode recuperar os dados armazenados no servidor através da rede de computadores do estabelecimento comercial. Para o cliente, a operação é transparente. O cliente entende que, ao acionar a caneta e identificar um produto, os dados são enviados diretamente ao computador do operador de caixa. O cliente fica muito satisfeito. Ao chegar ao caixa, seus produtos já estão somados. Eles efetuam apenas o pagamento, o que agiliza o processo de compras.

Este projeto consistiu no desenvolvimento de um hardware que facilitasse a compra, acoplando um equipamento eficiente na leitura do código de barras e desenvolvimento do programa controlador do hardware com códigos simples e com poucas linhas. Poucas linhas garante um melhor desempenho do hardware.

Após os testes realizados, fica comprovada a melhoria do projeto anterior. As limitações ora persistentes podem ser sanadas no processo de estudo, melhoria e aperfeiçoamento deste projeto.

O próximo tópico deste capítulo inclui sugestões para a melhoria e aperfeiçoamento deste projeto. São algumas sugestões de trabalhos futuros para colocar mais qualidade a este trabalho que agora não puderam ser tratadas porque requer um maior tempo com pesquisas, testes e orçamento.

Sugestões para trabalhos futuros

Implementar um dispositivo de celular, que lê os códigos de barras através de uma fotografia e os envia por meio da tecnologia Bluetooth do telefone celular. Com esse dispositivo o próprio cliente trás seu equipamento.

Desenvolver um sistema para os computadores dos operadores de caixa de forma mais amigável que possa integrar outros serviços.

Desenvolver um equipamento em que os produtos possam ser detectados automaticamente e contabilizados no momento em que são postos no carrinho de compras, por meio de um sensor que detecta o produto.

Implementar a solução por meio de redes Wi-Fi para que o alcance da transmissão de dados sejam maiores.

Criação de outros dispositivos para a coleta dos dados, de forma a facilitar o manuseio por parte do cliente.

REFERÊNCIAS BIBLIOGRÁFICAS

MICROSOFT. ***Microsoft SQL Server 2005 Implementation and Maintenance***. Editora Microsoft Press, 2006

Online Training Solutions, Inc. ***Microsoft Office Access 2003 - Passo a Passo***. Inc, 2003

PEREIRA, Fábio. ***Microcontroladores PIC***. Segunda Edição. Editora Érica, 2003

SILBERSCHATZ, Abraham; KORTH, Henry F e SUDARSHAN, S.. ***Sistema de Banco de Dados***. Tradução da Quinta Edição. Editora Campus, 2006

Site Bar Codes Symbology, disponível em http://www.gs1tw.org/twct/gs1w, último acesso em 10/06/2009.

Site da Microchip disponível em http://www.microchip.com, último acesso em 10/06/2009.

Site do SIG disponível em http://www.bluetooth.com , último acesso em 16/06/2009

TANENBAUM, Andrews S. ***Redes de Computadores***. Tradução da Quarta Edição. Editora Campus, 2003

TOLEDO, Luciano Cortez. ***Caneta Ótica para registro e contabilização automática de produtos***. Brasília, n.1, 2008. CD-ROM.

Zanco, Wagner da Silva. ***Microcontroladores PIC***. Primeira Edição. Editora Érica, 2005.

APÊNDICE I

Código-fonte do hardware do projeto Carrinho de compras inteligente.

```
program CCI

'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

'*          Definição e inicialização das variáveis          *

'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

'Aqui iremos definir as variáveis globais do sistema

'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

'Definição da PINAGEM do display

dim LCD_RS as sbit at RC4_bit

    LCD_EN as sbit at RC0_bit

    LCD_WR as sbit at RC3_bit


    LCD_D4 as sbit at RB5_bit

    LCD_D5 as sbit at RB4_bit

    LCD_D6 as sbit at RB3_bit

    LCD_D7 as sbit at RB2_bit


    LCD_RS_Direction as sbit at TRISC4_bit

    LCD_EN_Direction as sbit at TRISC0_bit

    LCD_WR_Direction as sbit at TRISC3_bit
```

```
    LCD_D4_Direction as sbit at TRISB5_bit
    LCD_D5_Direction as sbit at TRISB4_bit
    LCD_D6_Direction as sbit at TRISB3_bit
    LCD_D7_Direction as sbit at TRISB2_bit

'Definição da pinagem da conexão PS2
dim PS2_Data          as sbit at RA1_bit
    PS2_Clock         as sbit at RA0_bit

    PS2_Data_Direction  as sbit at TRISA1_bit
    PS2_Clock_Direction as sbit at TRISA0_bit

dim byte_rx as byte 'byte recebido pela serial

dim keydata  as byte 'byte recebido pela conexão PS2
dim keydataAnt  as byte 'keydata anterior, usado para evitar a repetição de teclas
dim special  as byte 'tecla especial
dim down  as byte    'verifica se a tecla está pressionada

dim bt_apertado as byte
dim CONT_AUX as byte

dim txt  as string[17] ' texto recebido pela serial e mostrado no display
```

```
dim conttxt as byte 'comprimento atual de txt


dim MAX_LINHA_LCD as byte 'quantidade maxima de linhas do display
dim LINHA_LCD as byte 'linha corrente do display
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
'*              Declaração da Rotina de Interrupção          *
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
sub procedure interrupt
    byte_rx = UART1_Read()
    if conttxt < 17 then 'se menor que 17 vou colocando o byte em txt
      strAppendSuf(txt, byte_rx)
      conttxt = conttxt + 1
    end if
end sub
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
'* Rotinas comuns
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
sub procedure PiscaLED
   if PORTB.0=1 then
      PORTC.1 = 0
   else
       PORTC.1 = 1  'Acende o LED
    end if
```

```
    if PORTB.1=1 then
       PORTC.2 = 0
    else
        PORTC.2 = 1
     end if

    if PORTA.3=1 then
       PORTC.3 = 0
    else
        PORTC.3 = 1
     end if

    if PORTA.5=1 then
       PORTC.5 = 0
    else
        PORTC.5 = 1
     end if
end sub
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
'* Envia comando ao servidor via serial a depender da tecla que foi
'* pressionada
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
sub procedure VerificarBotoes

    if (PORTB.0=0) then

     if down=1 then

        UART1_Write_Text ("NVG 0")

        delay_ms(100)

        down = 0

     end if

    else

     if PORTB.1=0 then

      if down=1 then

        UART1_Write_Text("NVG 1")

        delay_ms(100)

        down = 0

      end if

     else

       if PORTA.3=0 then

         if down=1 then

           UART1_Write_Text("PSQ")

           delay_ms(100)

           down = 0

         end if

       else

         if PORTA.5=0 then
```

```
            if down=1 then
              UART1_Write_Text("CNC")
              delay_ms(100)
              down = 0
            end if
          else
            down = 1
          end if
        end if
      end if
    end if
    PiscaLED
end sub
'**********************************
'*                Rotina de exibição do display
'**********************************
sub procedure MostrarNoDisplay()

  if LINHA_LCD= 1 then
    Lcd_Out(1,1,txt)
  end if

  if LINHA_LCD= 2 then
```

```
    Lcd_Out(2,1,txt)
  end if

  if LINHA_LCD= 3 then
    Lcd_Out(1,17,txt)
  end if

  if LINHA_LCD= 4 then
    Lcd_Out(2,17,txt)
  end if

  LINHA_LCD = LINHA_LCD + 1

  if LINHA_LCD > MAX_LINHA_LCD then
    LINHA_LCD = 1
  end if

end sub
'*********************************
'*          Rotina Principal     *
'*********************************
Main:
```

```
INTCON.PEIE = 1              ' enable peripheral interrupts

INTCON.GIE = 1               ' enable all interrupts


'INTCON.INTE = 1

'INTCON.INTF = 0


PIE1.RCIE = 1                ' enable interrupt on UART1 receive

PIE2.TXIE = 0                ' disable interrupt on UART1 transmit


adcon1=0x07                  'Configura todas as portas com função analógica como digital

trisc.6=0



UART1_Init(115200)   'Inicializa o módulo USART


trisc.6=0

conttxt = 1


trisc.5 = 0

trisc.3 = 0

trisc.2 = 0

trisc.1 = 0
```

```
portc.5 = 1

portc.3 = 1

portc.2 = 1

portc.1 = 1

'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

'Inicializa o módulo Bluetooth

'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

delay_ms(2000)

UART1_Write_Text("AT+ZV DefaultLocal")

UART1_Write_Text("CCI - Carrinho de Compras Interativo")

UART1_Write(13)

UART1_Write(10)


delay_ms(2000)


UART1_Write_Text("AT+ZV EnableBond")

UART1_Write(13)

UART1_Write(10)


delay_ms(2000)


'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
'Inicializa o Display 16x4
'**********************************
Lcd_Init()
Lcd_Out(1,1,"   CCI - v3.0 ")
Lcd_Out(2,1," Trabalho Final ")
Lcd_Out(1,17,"  Regis Levino  ")
Lcd_Out(2,17,"  Aguardando... ")
Lcd_Cmd(_LCD_CURSOR_OFF)
'**********************************
byte_rx=0
'**********************************
'Inicializa a comunicação PS2
'**********************************
Ps2_Config()
Delay_ms(100)
txt = ""
'**********************************
conttxt = 1
LINHA_LCD = 1
MAX_LINHA_LCD = 4

bt_apertado = 0
keydata = 0
```

```
special = 0
down    = 0
CONT_AUX = 0

portc.5 = 0
portc.3 = 0
portc.2 = 0
portc.1 = 0

while TRUE
  if Ps2_Key_Read(keydata, special, down) then      ' Se o dado foi lido do PS/2
     if (down <> 0) and (special = 0) and (keydata <> 0) then   ' Testa se é uma tecla comum
       if keydata <> keydataAnt then
         Lcd_Out(2,17, "Codigo:        ")
         Lcd_Out(2,25, keydata)
         keydataAnt=keydata
         UART1_Write(keydata)
       else
         keydataAnt = 0
       end if
     end if
  else
   if conttxt = 17 then  'Só coloco no display string com 16 caracteres recebidos pela serial
```

```
            MostrarNoDisplay()
            conttxt = 1
            txt = ""
        else
            VerificarBotoes()
        end if
      end if
    Wend
'*********************************
'*********************************

end.
```

APÊNDICE II

Código-fonte do servidor.

untServerCCI (Tela de apresentação do servidor)

```
procedure TfrmServer.SpeedButton1Click(Sender: TObject);
begin
  frmBlueTooth.ShowModal
end;

procedure TfrmServer.btnEnviarMensagemClick(Sender: TObject);
begin
  frmClientes.ShowModal;
end;

procedure TfrmServer.btnFecharClick(Sender: TObject);
begin
    close
end;

end.
```

APÊNDICE III

untBluetooth (Códigos que controlam a comunicação kit/servidor)

```
unit untBlueTooth;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls, wclConnections, wclSerialMonitor,
  wclAPI, wclErrors, Buttons, btOdeum;

type
  TfrmBlueTooth = class(TForm)
    Panel1: TPanel;
    lvPorts: TListView;
    btEnum: TButton;
    btConnect: TButton;
    tDisconnect: TButton;
    btnFecharPainel: TButton;
    meAns: TMemo;
    wclAPI: TwclAPI;
    wclSerialDiscovery: TwclSerialDiscovery;
    wclClient: TwclClient;
    btBeeper1: TbtBeeper;
    btnSincronizar: TButton;
    procedure FormCreate(Sender: TObject);
    procedure wclClientDisconnect(Sender: TObject);
    procedure wclClientConnect(Sender: TObject; Error: Integer);
    procedure btnFecharPainelClick(Sender: TObject);
```

```
    procedure btConnectClick(Sender: TObject);
    procedure btEnumClick(Sender: TObject);
    procedure tDisconnectClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure wclClientData(Sender: TObject; Buffer: Pointer;
      Size: Cardinal);
    procedure btnSincronizarClick(Sender: TObject);

  private

    // procedure EnviarFrase(Frase: AnsiString);
//    function MontarPacoteDoProduto(CodigoDeBarrasDoProduto: string): string;
    procedure EnviarFrasePara4LinhasDoLCD(frase: string);
    function FormatarPacotePara64Caracteres(dados: string): string;
    function FormatarPara12Caracteres(dados: string): string;
    function FormatarPara16Caracteres(dados: string): string;
    function FormatarPara3Caracteres(dados: string): string;
    function MontarPacoteTotalDasCompras: string;
    function VerificarSeCartaoExiste(s: AnsiString): boolean;
    function RetornarCodigoProdutoSemLetras(s: string): string;

    { Private declarations }
  public
    function  MontarPacote(): string;
    procedure EnviarDadosPelaSerialBlueTooth(s: AnsiString);
    { Public declarations }
  end;

var
  MATRICULA_CARTAO_CORRENTE : String='';
  CANCELAR_PRODUTO : boolean=false;
```

```
  PESQUISAR_PRODUTO : boolean=false;

  frmBlueTooth: TfrmBlueTooth;

implementation

uses untDB, untClientesAtivos;

{$R *.dfm}
//Ao carregar a API do Bluetooth é carregada
procedure TfrmBlueTooth.FormCreate(Sender: TObject);
begin
    wclAPI.Load;
end;
//Quando o bluetooth é desconectado aparece este aviso
procedure TfrmBlueTooth.wclClientDisconnect(Sender: TObject);
begin
   MessageDlg('Desconectado', mtInformation, [mbOK], 0);
end;
//Avisa quando o bluetooth é conectado
procedure TfrmBlueTooth.wclClientConnect(Sender: TObject; Error: Integer);
begin
  if Error = WCL_E_SUCCESS then
    MessageDlg('Conectado', mtInformation, [mbOK], 0)
  else
    MessageDlg('Incapaz de conectar. Error: ' + IntToStr(Error), mtError,
      [mbOK], 0);
end;
//Envio caracter pela serial do bluetooth
procedure TfrmBlueTooth.EnviarDadosPelaSerialBlueTooth(s: AnsiString);
begin
```

```
  if s = '' then
    MessageDlg('Enter AT command', mtError, [mbOK], 0)
  else begin
    s := s;
    wclShowError(wclClient.Write(Pointer(s), Length(s)));
  end;
end;
//Fecho o formulário bluetooth mais ele permanece na memoria
procedure TfrmBlueTooth.btnFecharPainelClick(Sender: TObject);
begin
  ModalResult := mrCancel;
end;
//Ao clicar no botão conectar este evento é chamado e caso a serial do bluetooth
//não esteja conectado uma tentativa de conectar é realizada
procedure TfrmBlueTooth.btConnectClick(Sender: TObject);
begin
 if wclClient.State <> csDisconnected then     //verifica se o status é diferente de
Desconectado
    MessageDlg('Client is active', mtWarning, [mbOK], 0)
  else
    if not Assigned(lvPorts.Selected) then   //porta serial selecionada
      MessageDlg('Select port', mtError, [mbOK], 0)
    else begin
      wclClient.Transport := ctSerial;      //se Desconectado entao conecto
      wclClient.SerialParams.Port := StrToInt(lvPorts.Selected.SubItems[0]);
      wclShowError(wclClient.Connect);
    end;
end;
//Quando o botão busca é acionado este evento é chamado
//Este procura pelas seriais bluetooth e lista em lvPorts
procedure TfrmBlueTooth.btEnumClick(Sender: TObject);
```

```
var
  Devices: TwclSerialDevices;
  I: Integer;
  Item: TListItem;
begin
  lvPorts.Items.Clear;
  Devices := TwclSerialDevices.Create;
  try
    if not wclShowError(wclSerialDiscovery.EnumDevices(Devices)) then
      if Devices.Count = 0 then
        MessageDlg('Nothing found', mtInformation, [mbOK], 0)
      else
        for I := 0 to Devices.Count - 1 do begin
          Item := lvPorts.Items.Add;
          Item.Caption := Devices[I].FriendlyName;
          Item.SubItems.Add(IntToStr(Devices[I].Port));
        end;
  finally
    Devices.Free;
  end;
end;
//Botao disconectar é acionado
procedure TfrmBlueTooth.tDisconnectClick(Sender: TObject);
begin
  wclClient.Disconnect;
end;
//Ao carregar o formulário o evento um click do botão buscar é chamado para não
precisar ficar apertando o botão
procedure TfrmBlueTooth.FormShow(Sender: TObject);
begin
  btEnumClick(nil);
```

```
end;
//***************************************************************************
procedure TfrmBlueTooth.EnviarFrasePara4LinhasDoLCD(frase : string);
var
  pacote : string;
begin
  pacote := frase;
  EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres('@'));//sincronismo
  EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres(pacote));
  sleep(200);
  pacote := '';
  EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres(pacote));
  sleep(200);
  EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres(pacote));
  sleep(200);
  EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres(pacote));
  sleep(200);
end;
//***************************************************************************
// Verifica se tem compras abertas
//***************************************************************************
function TfrmBlueTooth.VerificarSeCartaoExiste(s: AnsiString) : boolean;
var
  c : integer;
  MatriculaDoCartao,CodigoDeBarrasProduto,CCMItem,pacote: string;
begin
  result := False;
  s := UpperCase(s);

  MatriculaDoCartao := '';
```

```
c := pos('CRT=',s); //Codigo do cartão para abrir as compras
if c <> 0 then
begin
  c := pos('=',s);
  MatriculaDoCartao := copy(s,c+1, length(s)-(c));
end;

if MatriculaDoCartao <> '' then
begin
     if strtointdef(MatriculaDoCartao,-1) > 0 then
     begin
          if not data.cdsCompras.IsEmpty then data.cdsCompras.Close;
          if Data.CartaoExiste(MatriculaDoCartao) then
          begin
              Data.getCompras(MatriculaDoCartao);
              MATRICULA_CARTAO_CORRENTE := MatriculaDoCartao;
              EnviarFrasePara4LinhasDoLCD('Compras abertas.');
              Sleep(2000);
              pacote := MontarPacoteTotalDasCompras;
              EnviarDadosPelaSerialBlueTooth(pacote);
              frmClientes.edtCodigoCartao.Text :=  MATRICULA_CARTAO_CORRENTE;
              result := True;
          end
           else
             EnviarFrasePara4LinhasDoLCD('Matrícula não existe...');
      end
       else
          EnviarFrasePara4LinhasDoLCD('Matrícula inválida...');
  end
   else
     if DATA.cdsCompras.Active then
```

```
      begin
        if data.cdsCompras.IsEmpty then
          EnviarFrasePara4LinhasDoLCD('Nenhuma compra...') ;
        result := true;
      end
       else
          EnviarFrasePara4LinhasDoLCD('Compras fechada...')
end;
//*****************************************************************************
//
//*****************************************************************************
function TfrmBlueTooth.RetornarCodigoProdutoSemLetras(s: string) : string;
var
  aux,ch : string;
  i   : integer;
begin
  aux := '';
  if (s <> 'NVG 1') and (s <> 'NVG 0') and (s <> 'CNC') and (s <> 'PSQ') then
  begin
    for i:=1 to length(s) do
    begin
      ch := s[i];
      if strtointdef(ch,-1) > 0 then
        aux := aux + ch;
    end ;
    result := aux;
  end
   else
     result := s;
end;
//*****************************************************************************
```

```
//Todas as vezes que chega um dado na porta bluetooth do computador este evento é chamado
//Este analisa o que chegou e providencia uma resposta.
//Os dados podem ser algum dos botões do kit precionado ou a chegada de um CÓDIGO DE BARRAS DE PRODUTO
//Se for um numero é um codigo de barras caso contrario é um comando.
//*****************************************************************************
procedure TfrmBlueTooth.wclClientData(Sender: TObject; Buffer: Pointer;
  Size: Cardinal);
var
  c : integer;
  MatriculaDoCartao,CodigoDeBarrasProduto,CCMItem,pacote, s: AnsiString;
begin

  SetLength(s, Size);
  CopyMemory(Pointer(s), Buffer, Size);
  s := trim(s);
  meAns.Lines.Add(s); //mostra os comandos que vêm do kit
  pacote := '';

  if VerificarSeCartaoExiste(s)=false then exit;

  if pos('CRT=',uppercase(s)) <> 0 then exit;

  s := RetornarCodigoProdutoSemLetras(s);

  meAns.Lines.Add(s);

  if StrToIntDef(s,-1) = -1 then // verifica se é um número
  begin
```

```
    //recebeu um comando e vai tratá-lo, um dos quatro botões foi pressionado
    if s='NVG 1' then
    begin
      data.cdsCompras.Prior;
      pacote := MontarPacote();
      exit;
    end ;

    if s='NVG 0' then
    begin
      data.cdsCompras.Next;
      pacote := MontarPacote();
      exit;
    end;

    c := Pos('CNC',s);
    if c> 0 then
    begin
       EnviarFrasePara4LinhasDoLCD(''); //LIMPO O LCD
       if data.ExcluirProdutoDoCarrinho(data.cdsComprasCCM_ITEM.AsString) = True
then
       begin
         EnviarFrasePara4LinhasDoLCD('Excluido...');
         data.getCompras(MATRICULA_CARTAO_CORRENTE);//RECARREGO O GRID
       end
        else
          EnviarFrasePara4LinhasDoLCD('Falha ao excluir...');
       Sleep(1000);
       pacote := MontarPacote();
    end;
```

```
  c := Pos('PSQ',s);
  if c > 0 then
  begin
     pacote := MontarPacoteTotalDasCompras;
     EnviarDadosPelaSerialBlueTooth(pacote);
    {PESQUISAR_PRODUTO := Not PESQUISAR_PRODUTO;
    if PESQUISAR_PRODUTO then
    begin
      EnviarFrasePara4LinhasDoLCD('Pesquisa inicio...');
    end
     else
      begin
        EnviarFrasePara4LinhasDoLCD('Pesquisa fim...');
      end; {}
      exit;
  end;

end
 else
  {begin
   { if PESQUISAR_PRODUTO then
    begin
      CodigoDeBarrasProduto := Trim(s);
      MontarPacoteDoProduto(CodigoDeBarrasProduto);
      exit;
    end
     else  {}
       begin
         s := data.AdicionarProdutoNoCarrinho(MATRICULA_CARTAO_CORRENTE,s);
         if s <> '' then
         begin
```

```
            btBeeper1.PlayPresetSound(psError);
            EnviarFrasePara4LinhasDoLCD(s);
            exit;
          end
           else
            begin
              btBeeper1.PlayPresetSound(psOk);
              pacote := MontarPacote() ;
              exit;
            end;
        end;
  // end;

  if length(pacote) > 0 then
    EnviarDadosPelaSerialBlueTooth(pacote);
    {}
end;

{function TfrmBlueTooth.MontarPacoteDoProduto(CodigoDeBarrasDoProduto : string):
string;
begin
 try
    EnviarFrasePara4LinhasDoLCD(''); //LIMPO O LCD
    result := data.getDescricao(CodigoDeBarrasDoProduto);
    if result='' then
      EnviarFrasePara4LinhasDoLCD('Produto não existe.')
    else
      begin
        result := FormatarPara16Caracteres(result);
        EnviarDadosPelaSerialBlueTooth(result);
        sleep(200);
```

```
        result := 'Preço: ' + data.getPreco(CodigoDeBarrasDoProduto);
        EnviarDadosPelaSerialBlueTooth(FormatarPara16Caracteres(result));
        sleep(200);
      end  ;
  except
    result := '';
  end;
end;{}

function TfrmBlueTooth.FormatarPara3Caracteres(dados: string): string;
var
  c: integer;
begin
  c := length(dados);
  if c <3 then
    result := dados + '     '
  else
    result := dados  ;
  result := copy(result,1,3)  ;
end;

function TfrmBlueTooth.FormatarPara16Caracteres(dados: string): string;
var
  c: integer;
begin
  c := length(dados);
  if c <16 then
    result := dados + '                '
  else
    result := dados  ;
  result := copy(result,1,16)  ;
```

```
end;

function TfrmBlueTooth.FormatarPara12Caracteres(dados: string): string;
var
  c: integer;
begin
  c := length(dados);
  if c <12 then
    result := dados + '                          '
  else
    result := dados  ;
  result := copy(result,1,12)  ;
end;

function TfrmBlueTooth.FormatarPacotePara64Caracteres(dados : string): string;
begin
   result := '                                                                ';
   result := copy(dados + result,1,64);
end;{}

function TfrmBlueTooth.MontarPacoteTotalDasCompras(): string;
begin
   EnviarFrasePara4LinhasDoLCD(''); //limpo LCD
   Sleep(100);
   result := FormatarPara16Caracteres( 'Quantidade = ' +
inttostr(data.cdsCompras.RecordCount));
   EnviarDadosPelaSerialBlueTooth(result);
   Sleep(100);
   result := FormatarPara16Caracteres('Valor = ' + data.getTotalGeral())   ;
   EnviarDadosPelaSerialBlueTooth(result);
end;
```

```
procedure TfrmBlueTooth.btnSincronizarClick(Sender: TObject);
begin
    //EnviarFrasePara4LinhasDoLCD( '12345678901234567890');
     EnviarFrasePara4LinhasDoLCD(''); //LIMPO O LCD
     EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres('@'));
end;


function TfrmBlueTooth.MontarPacote(): string;
var
  lcdLinha1,
  lcdLinha2,
  lcdLinha3,
  lcdLinha4 : string;
begin
  lcdLinha1 := '';
  lcdLinha2 := '';
  lcdLinha3 := '';
  lcdLinha4 := '';

  //EnviarFrasePara4LinhasDoLCD('');
  EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres('@'));//sincronismo

  Sleep(250);

  if data.cdsCompras.RecordCount > 0 then data.cdsCompras.Prior;
  if data.cdsCompras.RecordCount > 1 then data.cdsCompras.Prior;
  if data.cdsCompras.RecordCount > 2 then data.cdsCompras.Prior;

  if data.cdsCompras.RecordCount > 0 then
```

```
  begin
     lcdLinha1 :=
FormatarPara12Caracteres(inttostr(data.cdsComprasCCM_ITEM.AsInteger) + '-' +
data.cdsComprasCCM_DESCRICAO_COMPRA.AsString);
     lcdLinha1 := lcdLinha1 + '-' +
FormatarPara3Caracteres(data.cdsComprasCCM_PRECO_EPOCA.AsString);
     EnviarDadosPelaSerialBlueTooth(FormatarPara16Caracteres( lcdLinha1));
     Application.ProcessMessages;
     Sleep(250);
  end;

  if data.cdsCompras.RecordCount > 1 then
  begin
     data.cdsCompras.Next;
     lcdLinha2 :=
FormatarPara12Caracteres(inttostr(data.cdsComprasCCM_ITEM.AsInteger) + '-' +
data.cdsComprasCCM_DESCRICAO_COMPRA.AsString);
     lcdLinha2 := lcdLinha2 + '-' +
FormatarPara3Caracteres(data.cdsComprasCCM_PRECO_EPOCA.AsString);
     EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres(lcdLinha2));
     Application.ProcessMessages;
     Sleep(250);
  end;

  if data.cdsCompras.RecordCount > 2 then
  begin
     data.cdsCompras.Next;
     lcdLinha3 :=
FormatarPara12Caracteres(inttostr(data.cdsComprasCCM_ITEM.AsInteger) + '-' +
data.cdsComprasCCM_DESCRICAO_COMPRA.AsString);
```

```
    lcdLinha3 := lcdLinha3 + '-' +
FormatarPara3Caracteres(data.cdsComprasCCM_PRECO_EPOCA.AsString);
    EnviarDadosPelaSerialBlueTooth(FormatarPara16Caracteres( lcdLinha3));
    Application.ProcessMessages;
    Sleep(250);
  end;

  if data.cdsCompras.RecordCount > 3 then
  begin
    data.cdsCompras.Next;
    lcdLinha4 :=
FormatarPara12Caracteres(inttostr(data.cdsComprasCCM_ITEM.AsInteger) + '-' +
data.cdsComprasCCM_DESCRICAO_COMPRA.AsString);
    lcdLinha4 := lcdLinha4 + '-' +
FormatarPara3Caracteres(data.cdsComprasCCM_PRECO_EPOCA.AsString);
    EnviarDadosPelaSerialBlueTooth( FormatarPara16Caracteres(lcdLinha4));
    Application.ProcessMessages;
    Sleep(250);
  end;

  result := FormatarPacotePara64Caracteres(lcdLinha1 + lcdLinha2 + lcdLinha3 +
lcdLinha4); {}
end;

end.
```

APÊNDICE IV

unit untDB; (Funções de consultas no banco de dados)

```
unit untDB;

interface

uses
  SysUtils, Classes, DB, ADODB, DBClient, Provider;

type
  TData = class(TDataModule)
    DBADO: TADOConnection;
    qryAux: TADOQuery;
    dspAux: TDataSetProvider;
    cdsAux: TClientDataSet;
    qryCompras: TADOQuery;
    qryComprasCCM_DESCRICAO_COMPRA: TWideStringField;
    qryComprasCCM_PRECO_EPOCA: TBCDField;
    qryComprasCCM_ID: TAutoIncField;
    qryComprasCCM_ITEM: TIntegerField;
    qryComprasCRT_MATRICULA: TWideStringField;
    qryComprasPDT_CODIGO_BARRAS: TWideStringField;
    qryComprasPAGAMENTO_EFETUADO: TSmallintField;
    qryComprasDATA_HORA: TDateTimeField;
    qryComprasCCM_QUANTIDADE: TIntegerField;
    dspCompras: TDataSetProvider;
    cdsCompras: TClientDataSet;
    cdsComprasCCM_ITEM: TIntegerField;
    cdsComprasCCM_DESCRICAO_COMPRA: TWideStringField;
```

```
    cdsComprasCCM_PRECO_EPOCA: TBCDField;
    cdsComprasCCM_QUANTIDADE: TIntegerField;
    cdsComprasCCM_ID: TAutoIncField;
  private
    { Private declarations }
  public
    function CartaoExiste(MatriculaCartao: string) : boolean;
    function getDescricao(codigoBarras: string): string;
    function getPreco(codigoBarras: string): string;
    function getContagem(MatriculaCartao : string): string;
    function AdicionarRegistroDeCompraNoBanco(CodigoCartao, codigoBarras ,
descricao, preco , Item: string; PagamentoEfetuado: boolean): boolean;
    function ExcluirProdutoDoCarrinho(ITEM: string): boolean;
    function  ExcluirProdutoDoCarrinho2(MatriculaCartao,  CCMItemDaCompra  :  string):
boolean;
    function getCompras(MatriculaCartao: string) : boolean;
    function getTotalGeral(): string;
    function AdicionarProdutoNoCarrinho(codigoCartao, codigoProduto: string): string;
    { Public declarations }
  end;

var
  Data: TData;

implementation

{$R *.dfm}

function TData.getDescricao(codigoBarras : string): string;
var
  aux : string;
```

```
begin
  qryAux.Close;
  cdsAux.Close;
  aux := 'SELECT * FROM PRODUTOS WHERE PDT_CODIGO_BARRAS=' +
QuotedStr(codigoBarras);
  qryAux.SQL.Clear;
  qryAux.SQL.Text := aux;
  cdsAux.Open;
  if cdsAux.IsEmpty then
     result := ''
  else
      result := cdsAux.FieldByName('PDT_DESC').AsString;
end;

function TData.ExcluirProdutoDoCarrinho(ITEM : string): boolean;
var
  aux : string;
begin
  try
      result := false;
      qryAux.Close;
      aux := 'DELETE FROM CARTAO_COMPRAS WHERE CCM_ITEM=' + ITEM ;
      qryAux.SQL.Clear;
      qryAux.SQL.Text := aux;
      if qryAux.ExecSQL >0  then result := true;
  except
      result := false;
  end;
end;
```

```
function TData.ExcluirProdutoDoCarrinho2(MatriculaCartao, CCMItemDaCompra : string): boolean;
var
  aux : string;
begin
  try
     result := false;
     qryAux.Close;
     aux := 'DELETE FROM CARTAO_COMPRAS WHERE CCM_ITEM=' + CCMItemDaCompra + ' AND CRT_MATRICULA='+ QuotedStr( MatriculaCartao );
     qryAux.SQL.Clear;
     qryAux.SQL.Text := aux;
     if qryAux.ExecSQL >0  then result := true;
  except
     result := false;
  end;
end;

function TData.getContagem(MatriculaCartao : string): string;
var
  aux : string;
  teste : integer;
begin
  qryAux.Close;
  cdsAux.Close;
  aux := 'SELECT Max(CCM_ITEM) as ITEM FROM CARTAO_COMPRAS WHERE CRT_MATRICULA=' + QuotedStr(MatriculaCartao);
  qryAux.SQL.Clear;
  qryAux.SQL.Text := aux;
  cdsAux.Open;
  if cdsAux.IsEmpty then
```

```
    result := ''
  else
    begin
      teste := strtointdef(cdsAux.FieldByName('ITEM').AsString,0);
      result := inttostr(teste+1);
    end;
end;

function TData.getPreco(CodigoBarras : string): string;
var
  aux : string;
begin
  qryAux.Close;
  cdsAux.Close;
  aux := 'SELECT * FROM PRODUTOS WHERE PDT_CODIGO_BARRAS=' +
QuotedStr(CodigoBarras);
  qryAux.SQL.Clear;
  qryAux.SQL.Text := aux;
  cdsAux.Open;
  if cdsAux.IsEmpty then
     result := ''
  else
      result := cdsAux.FieldByName('PDT_PRECO').AsString;
end;

function TData.AdicionarRegistroDeCompraNoBanco(CodigoCartao, codigoBarras,
descricao, preco,Item : string; PagamentoEfetuado: boolean): boolean;
var
  aux : string;
begin
  result := true;
```

```
  qryAux.Close;
  aux := 'INSERT INTO CARTAO_COMPRAS
(CRT_MATRICULA,PDT_CODIGO_BARRAS,CCM_DESCRICAO_COMPRA,CCM_PR
ECO_EPOCA,CCM_ITEM,DATA_HORA) VALUES (' +
                          QuotedStr(CodigoCartao) + ',' +
                          QuotedStr(codigoBarras) + ','  +
                          QuotedStr(descricao) + ','  +
                          QuotedStr(preco) + ','  +
                          item + ','  +
                          QuotedStr(FormatDateTime('dd/mm/yyyy hh:mm:ss',now)) + ')';
  qryAux.SQL.Clear;
  qryAux.SQL.Text := aux;
  try
    qryAux.ExecSQL;
  except
    result := false;
  end;
end;

function TData.CartaoExiste(MatriculaCartao: string) : boolean;
var
  aux : string;
begin
  qryAux.Close;
  cdsAux.Close;
  aux := 'SELECT * FROM CARTAO WHERE CRT_MATRICULA=' +
QuotedStr(MatriculaCartao);
  qryAux.SQL.Clear;
  qryAux.SQL.Text := aux;
  cdsAux.Open;
  if cdsAux.IsEmpty then
```

```
    result := false
  else
     result := true;
end;

function TData.getCompras(MatriculaCartao: string) : boolean;
var
  aux : string;
begin
  cdsCompras.Close;
  qryCompras.Close;
  aux := 'SELECT * FROM CARTAO_COMPRAS WHERE CRT_MATRICULA=' +
QuotedStr(MatriculaCartao) + ' ORDER BY CCM_ITEM ASC';
  qryCompras.SQL.Clear;
  qryCompras.SQL.Text := aux;
  cdsCompras.Open;

  if cdsCompras.IsEmpty then
    result := false
  else
     result := true;
end;

function  TData.getTotalGeral(): string;
var
  total : real;
  aux : string;
begin
  total := 0;
  cdsCompras.First;
  while not cdsCompras.Eof do
```

```
  begin
     aux :=cdsComprasCCM_PRECO_EPOCA.AsString;
     total := total + strtofloatdef(aux,0);
     cdsCompras.Next;
  end;          {}
  result := floattostr(total);
end;

function TData.AdicionarProdutoNoCarrinho(codigoCartao, codigoProduto: string):
string;
var
  Item, Descricao, Preco : string;
begin
   result := '';
   if CartaoExiste(codigoCartao)=False then
   begin
     result := 'CARTÃO NÃO EXISTE...';
     exit;
   end
    else
      begin
        Descricao :=  getDescricao(codigoProduto);
        Preco     :=  getPreco    (codigoProduto);
        if (Descricao='') or (Preco='') then
        begin
          result := 'NÃO CADASTRADO...';
        end
         else
          begin
            Item      :=  getContagem(codigoCartao);
```

```
            if
AdicionarRegistroDeCompraNoBanco(codigoCartao,codigoProduto,Descricao,
Preco,Item, true)=False then
            begin
              result := 'NÃO CADASTRADO...';
            end
             else
              begin
                result :='';

getCompras(codigoCartao);//btnCarregarComprasPeloNumeroDoCartaoClick(nil);
              end;
          end;
      end;
end;

end.
```

APÊNDICE V

unit untPrincipal; (Código de cadastros do servidor)

```
unit untPrincipal;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ExtCtrls, Buttons;

type
  TfrmPrincipal = class(TForm)
    Menu: TMainMenu;
    mnuAdmin: TMenuItem;
    mnuEmpresa1: TMenuItem;
    N16: TMenuItem;
    mnuUsuarios1: TMenuItem;
    mnuCadastro: TMenuItem;
    mnuEmpresasdeTercearizaodeServios1: TMenuItem;
    mnuRelatorios: TMenuItem;
    mnuManuteno1: TMenuItem;
    Utilitarios1: TMenuItem;
    mnuCalculadora1: TMenuItem;
    mnuEditorTexto: TMenuItem;
    mnuImpressoras1: TMenuItem;
    N1: TMenuItem;
    mnuPreferencias1: TMenuItem;
    mnuAjuda: TMenuItem;
    Sobre1: TMenuItem;
```

```
  Outros1: TMenuItem;
  Sair1: TMenuItem;
  Sair2: TMenuItem;
  Produtos1: TMenuItem;
  AtendentesCaixas1: TMenuItem;
  N2: TMenuItem;
  CarrinhosInteligentes1: TMenuItem;
  Panel1: TPanel;
  btnClientes: TSpeedButton;
  btnUsuarios: TSpeedButton;
  btnFornecedores: TSpeedButton;
  btnFechar: TSpeedButton;
  btnCaixa: TSpeedButton;
  btnProdutos: TSpeedButton;
  btnCarrinhos: TSpeedButton;
  btnCartao: TSpeedButton;
  btnEstoque: TSpeedButton;
  Bevel1: TBevel;
  procedure FormShow(Sender: TObject);
  procedure btnFecharClick(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure btnUsuariosClick(Sender: TObject);
  procedure btnClientesClick(Sender: TObject);
  procedure btnFornecedoresClick(Sender: TObject);
  procedure btnProdutosClick(Sender: TObject);
  procedure btnCarrinhosClick(Sender: TObject);
  procedure btnCaixaClick(Sender: TObject);
  procedure btnCartaoClick(Sender: TObject);
private
  procedure StatusBotosPrincipais(sts: boolean);
  { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  LOGIN_USUARIO : String='NÃO DEFINIDO';
  frmPrincipal: TfrmPrincipal;

implementation

{$R *.dfm}
uses
  untLogin, untCadUsuarios, untCadClientes, untCadFornecedores,
  untCadProdutos, untCarrinhosDeCompras, untCaixa, untCadCartao;

procedure TfrmPrincipal.StatusBotosPrincipais(sts : boolean);
begin
  btnUsuarios.Enabled :=  sts;
  btnCartao.Enabled :=  sts;
  btnClientes.Enabled :=  sts;
  btnFornecedores.Enabled :=  sts;
  btnProdutos.Enabled :=  sts;
  btnCarrinhos.Enabled :=  sts;
  btnCaixa .Enabled :=  sts;
  btnFechar.Enabled :=  sts;
end;

procedure TfrmPrincipal.FormShow(Sender: TObject);
begin
   self.Top  := 10;
   self.Left := 10;
```

```
end;

procedure TfrmPrincipal.btnFecharClick(Sender: TObject);
begin
  Close
end;

procedure TfrmPrincipal.FormCreate(Sender: TObject);
begin
  frmLogin := TfrmLogin.Create(frmPrincipal);
  frmLogin.ShowModal ;
  frmLogin.Free;
end;

procedure TfrmPrincipal.btnUsuariosClick(Sender: TObject);
begin
  StatusBotosPrincipais(false);
  frmCadUsuarios := TfrmCadUsuarios.Create(Application);
  frmCadUsuarios.ShowModal;
  frmCadUsuarios.Free;
  frmCadUsuarios := nil;
  StatusBotosPrincipais(true);
end;

procedure TfrmPrincipal.btnClientesClick(Sender: TObject);
begin
  StatusBotosPrincipais(false);
  frmCadClientes := TfrmCadClientes.Create(Application);
  frmCadClientes.ShowModal;
  frmCadClientes.Free;
  frmCadClientes := nil;
```

```
  StatusBotosPrincipais(true);
end;

procedure TfrmPrincipal.btnFornecedoresClick(Sender: TObject);
begin
  StatusBotosPrincipais(false);
  frmCadFornecedores := TfrmCadFornecedores.Create(Application);
  frmCadFornecedores.ShowModal;
  frmCadFornecedores.Free;
  frmCadFornecedores := nil;
  StatusBotosPrincipais(true);
end;

procedure TfrmPrincipal.btnProdutosClick(Sender: TObject);
begin
  StatusBotosPrincipais(false);
  frmCadProdutos := TfrmCadProdutos.Create(Application);
  frmCadProdutos.ShowModal;
  frmCadProdutos.Free;
  frmCadProdutos := nil;
  StatusBotosPrincipais(true);
end;

procedure TfrmPrincipal.btnCarrinhosClick(Sender: TObject);
begin
  StatusBotosPrincipais(false);
  frmCadCarrinhoDeCompra := TfrmCadCarrinhoDeCompra.Create(Application);
  frmCadCarrinhoDeCompra.ShowModal;
  frmCadCarrinhoDeCompra.Free;
  frmCadCarrinhoDeCompra := nil;
  StatusBotosPrincipais(true);
```

```
end;

procedure TfrmPrincipal.btnCartaoClick(Sender: TObject);
begin
  StatusBotosPrincipais(false);
  frmCadCartao := TfrmCadCartao.Create(Application);
  frmCadCartao.ShowModal;
  frmCadCartao.Free;
  frmCadCartao := nil;
  StatusBotosPrincipais(true);
end;

end.
```

unit untCadCartao;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, untFrmBase, DB, ADODB, Grids, DBGrids, ComCtrls, StdCtrls,
  Buttons, ExtCtrls, Mask, DBCtrls;

type
  TfrmCadCartao = class(TfrmBase)
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit2: TDBEdit;
    cdsGeralCRT_ID: TAutoIncField;
    cdsGeralCRT_MATRICULA: TWideStringField;
```

```
    cdsGeralCRT_DESC: TWideStringField;
    procedure btnIncluirClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmCadCartao: TfrmCadCartao;

implementation

{$R *.dfm}

procedure TfrmCadCartao.btnIncluirClick(Sender: TObject);
begin
  inherited;

end;

end.
```

unit untCadProdutos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, untFrmBase, DBCtrls, StdCtrls, Mask, DB, ADODB, Grids, DBGrids,
```

```
  ComCtrls, Buttons, ExtCtrls;

type
  TfrmCadProdutos = class(TfrmBase)
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit2: TDBEdit;
    Label3: TLabel;
    DBLookupComboBox1: TDBLookupComboBox;
    DBEdit3: TDBEdit;
    Label4: TLabel;
    cdsGeralPDT_ID: TAutoIncField;
    cdsGeralPDT_CODIGO_BARRAS: TWideStringField;
    cdsGeralPDT_DESC: TWideStringField;
    cdsGeralPDT_PRECO: TBCDField;
    cdsGeralFRN_ID: TIntegerField;
    ADODataSet1: TADODataSet;
    DataSource1: TDataSource;
    ADODataSet1FRN_ID: TAutoIncField;
    ADODataSet1FRN_DESC: TWideStringField;
    procedure btnIncluirClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmCadProdutos: TfrmCadProdutos;
```

```
implementation

{$R *.dfm}

procedure TfrmCadProdutos.btnIncluirClick(Sender: TObject);
begin
  inherited;
  ADODataSet1.Open;
end;

end.
```

```
unit untCadFornecedores;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, untFrmBase, DB, ADODB, Grids, DBGrids, ComCtrls, StdCtrls,
  Buttons, ExtCtrls, Mask, DBCtrls;

type
  TfrmCadFornecedores = class(TfrmBase)
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label3: TLabel;
    DBEdit3: TDBEdit;
    Label4: TLabel;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
```

```
    Label5: TLabel;
    procedure btnIncluirClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmCadFornecedores: TfrmCadFornecedores;

implementation

{$R *.dfm}

procedure TfrmCadFornecedores.btnIncluirClick(Sender: TObject);
begin
  inherited;

end;

end.
```

APÊNDICE VI

unit untCaixa; (Código do operador do caixa)

```
unit untCaixa;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Grids,Buttons, wclConnections,
  wclSerialMonitor, wclAPI, ComCtrls,  btOdeum, DBGrids, DB, DBClient,
  Provider, ADODB;

type
  TfrmCaixa = class(TForm)
    Label1: TLabel;
    edtCodigoBarras: TEdit;
    Label2: TLabel;
    edtPreco: TEdit;
    Label3: TLabel;
    edtTotalParcial: TEdit;
    Label4: TLabel;
    edtQuantidade: TEdit;
    Label5: TLabel;
    edtTotalGeral: TEdit;
    Shape4: TShape;
    Bevel1: TBevel;
    Bevel2: TBevel;
    btnAdicionar: TSpeedButton;
    btnRemover: TSpeedButton;
```

```
    btnFechar: TSpeedButton;
    Label6: TLabel;
    btnCarregarComprasPeloNumeroDoCartao: TSpeedButton;
    btBeeper1: TbtBeeper;
    btnCarregarCartao: TSpeedButton;
    DBGrid1: TDBGrid;
    dtsCompras: TDataSource;
    Bevel3: TBevel;
    edtAviso: TEdit;
    edtCodigoCartao: TEdit;
    procedure btnFecharClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btnCarregarComprasPeloNumeroDoCartaoClick(Sender: TObject);
    procedure btnCarregarCartaoClick(Sender: TObject);
    procedure btnAdicionarClick(Sender: TObject);
    procedure btnRemoverClick(Sender: TObject);
  private
    procedure ExibirDadosDaCompra;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  LINHA_CORRENTE : integer=0;
  frmCaixa: TfrmCaixa;

implementation

uses untDB, untPrincipal, Math;
```

```
{$R *.dfm}

procedure TfrmCaixa.btnFecharClick(Sender: TObject);
begin
  Close
end;

procedure TfrmCaixa.FormActivate(Sender: TObject);
begin
  edtCodigoCartao.SetFocus
end;

procedure TfrmCaixa.btnCarregarComprasPeloNumeroDoCartaoClick(
  Sender: TObject);
begin
   edtAviso.Text := '';
   if Data.CartaoExiste(edtCodigoCartao.Text)=False then
   begin
     edtAviso.Text := 'Cartão não cadastrado, favor informar um código válido para o
cartão';
     edtCodigoCartao.SetFocus;
     exit;
   end;
   data.getCompras(edtCodigoCartao.Text);

   ExibirDadosDaCompra();
end;

procedure TfrmCaixa.ExibirDadosDaCompra();
begin
  edtTotalGeral.Text := data.getTotalGeral();
```

```
  edtPreco.Text := data.cdsComprasCCM_PRECO_EPOCA.AsString;
  edtQuantidade.Text := inttostr(data.cdsCompras.RecordCount);
  edtTotalParcial.Text :=  edtTotalGeral.Text;
  edtAviso.Text := 'Item -> ' + data.cdsComprasCCM_ITEM.AsString
end;{}

procedure TfrmCaixa.btnCarregarCartaoClick(Sender: TObject);
begin
  frmPrincipal.btnCartaoClick(NIL)
end;

procedure TfrmCaixa.btnAdicionarClick(Sender: TObject);
begin
  edtAviso.Text := data.AdicionarProdutoNoCarrinho (edtCodigoCartao.Text,
edtCodigoBarras.Text);
  edtCodigoCartao.SetFocus;
  if edtAviso.Text <> '' then
    btBeeper1.PlayPresetSound(psError)
  else
    begin
      btBeeper1.PlayPresetSound(psOk);
      edtCodigoBarras.Text := '';
    end;
  ExibirDadosDaCompra();
end;

procedure TfrmCaixa.btnRemoverClick(Sender: TObject);
begin
  if MessageDlg('Tem certeza que deseja excluir este produto
?',mtConfirmation,mbYesNoCancel,0)=IDYES then
  begin
```

```
    if data.ExcluirProdutoDoCarrinho(data.cdsComprasCCM_ITEM.AsString) = True then
    begin
      data.getCompras(edtCodigoCartao.Text);
      ShowMessage('Produto excluído com sucesso !');
      ExibirDadosDaCompra();
    end
     else
       ShowMessage('Não foi possivel excluir o produto !');
  end;
end;


end.
```