UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM MANUTENÇÃO INDUSTRIAL
CAMPUS MEDIANEIRA

DAIANE BECKER
KARINA ROSANA DOTTO

DESENVOLVIMENTO DE UM CONTROLE DE ACESSO MICROCONTROLADO COM CÓDIGO DE BARRAS

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA – PR
2013

DAIANE BECKER
KARINA ROSANA DOTTO

DESENVOLVIMENTO DE UM CONTROLE DE ACESSO MICROCONTROLADO COM CÓDIGO DE BARRAS

Trabalho de Conclusão de Curso apresentado como requisito à obtenção do grau de Tecnólogo, no Curso Superior de Tecnologia em Manutenção Eletromecânica promovido pela Universidade Tecnológica Federal do Paraná – UTFPR, Campus Medianeira.

Orientador: Prof. Adriano de Andrade Bresolin

MEDIANEIRA-PR

2013

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação profissional
Coordenação do curso superior de
tecnologia em manutenção industrial

TERMO DE APROVAÇÃO

Implantação de um controle de acesso microcontrolado com código de barras
Por

Daiane Becker

Karina Rosana Dotto

Este Trabalho de Conclusão de Curso (TCC) foi apresentado às 16:00 hs do dia 11 de abril de 2013 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Manutenção Eletromecânica, da Universidade Tecnológica Federal do Paraná, Campus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinado. Após deliberação, a Banca Examinadora considerou o trabalho Aprovado.

Prof. Yuri Ferruzzi
Responsável pelas atividades
de TCC(UTFPR)

Prof. Adriano de Andrade
Bresolin
Orientador
(UTFPR)

Prof. Alberto Noboru Miyadaira
UTFPR – Campus Medianeira
(UTFPR)

Prof. Yuri Ferruzzi
UTFPR – Campus Medianeira
(UTFPR)

AGRADECIMENTOS

Aos nossos pais, pelo apoio e compreensão.

Ao professor Adriano, pela orientação.

A Universidade, pelo espaço e oportunidade.

RESUMO

O controle de acesso por microcontrolador, já vem sendo muito utilizado para maior segurança de um local, onde apenas as pessoas autorizadas para a entrada do ambiente terão acesso ao mesmo. Alguns locais exemplos do uso do controle de acesso são: escritórios, bancos, prefeituras. Este trabalho de conclusão de curso explana sobre a confecção de um sistema de Acesso Microcontrolado por Código de Barras que controle o acesso do laboratório J38 da UTFPR – Campus de medianeira.

ABSTRACT

Access control by microcontroller is already being widely used for security of places where only authorized persons have access to it. Some local that use it for access control are: offices, banks, prefecture. This work completion of course explains about the making of a System Access Barcode Microcontrolado by controlling access lab J38 UTFPR - Campus mediator.

LISTA DE QUADROS

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

O avanço da tecnologia revolucionou de forma drástica e definitiva a forma como o ser humano encarava a sua vida e a sua relação com o mundo em que vivia.

Qualquer instituição em busca de qualidade e segurança procura cada vez mais aperfeiçoar qualquer sistema diretamente ligado as atividades, onde o tempo se torna o fator predominante para o sucesso organizacional.

A fim de facilitar as atividades diárias, a informatização e automação são exploradas de tal maneira que aumentem a eficiência das operações.

No Campus Medianeira da UTFPR circulam em torno de 160 professores e 95 servidores, deste modo diariamente são acessados centenas de ambientes com chave e sem chave. Visando à maior comodidade destes funcionários nota-se a necessidade de implantação de um sistema de controle de acesso controlado nos ambientes.

Hoje em dia o campus conta com o acesso manual das salas através de chaves o que dificulta muitas vezes o acesso dos funcionários.

O objetivo do controle de acesso visa à segurança dos ambientes, das informações e dos bens moveis contra acesso não autorizado e visa também eliminar a quantidade de chaves necessárias como mostra a figura 1 para que o funcionário tenha acesso a todas as áreas desejadas na universidade.

Figura 1: Molho de Chaves utilizado pelo professor Orientador

O objetivo principal deste projeto é eliminar o excesso das mesmas. Para tanto, notou-se a necessidade de otimizar este processo, onde todas as salas pudessem ser acessadas por um único mecanismo.

Diante desta necessidade verificou-se que a implantação de um sistema que pudesse ler o código de barras do crachá dos servidores poderia ser um modo de identificar a pessoa que pretende ter acesso a uma determinada sala ou laboratório.

O objetivo principal deste trabalho de conclusão de curso – TCC é o desenvolvimento de um sistema que controle acesso ao laboratório J31 mais conhecido como LabSis através da leitura de um cartão de código de barras com oito dígitos (padrão da UTFPR) e digitação de uma senha com quatro dígitos.

2. METODOLOGIA E FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas de Controle de Acesso

Os sistemas de controle de acessos físicos são os que utilizam barreiras físicas e métodos de identificação e contagem para as pessoas ou veículos que queiram ou necessitem transpor essas barreiras. Serve para restringir o movimento de entrada e saída em ambientes os mais diversos.

Estes sistemas são utilizados para o cadastro e controle de fluxo de visitantes, prestadores de serviço, funcionários e veículos e pode ser efetuado em qualquer ambiente, ou seja, qualquer local que se queira controlar o acesso, como salas, departamentos ou mesmo estacionamentos.

Para os sistemas de controle de acesso físico existem muitas e variadas opções disponíveis no mercado. Esses sistemas envolvem quase sempre uma barreira física e alguns dispositivos de entrada de dados que permitem receber as informações e liberam ou não a barreira física, permitindo a entrada do individuo ou veiculo.

O circuito utilizado neste trabalho se assemelha a muitos outros equipamentos comerciais distribuídos no mercado especializado. A inserção da leitura de um código de barras (cartão) aliada à digitação de uma senha para confirmar o usuário garante um nível muito bom de segurança no controle de acesso a recinto e outros. O projeto por ser aplicado em diversas situações como:

- controle de acesso de pessoas em áreas restritas (escritórios, fábricas, residências);

- controle liga/desliga de equipamentos (alarme residenciais, maquinas importantes no chão de fabrica);

- controle automotivo (controle de partida; inibição de alarmes).

Todo o controle é feito através de microcontroladores que controla um display de cristal liquido (LCD), um teclado, um relé, dois leds, um buzzer e dois pontos de comunicação: um com o leitor de cartões de código de barras e outro com o PC.

2.2 O Problema Encontrado

O acesso ao laboratório de sistema inteligente (sala J-38) é muito simples, pois conta apenas com um controle mecânico via fechadura com chave. Esta chave pode ser encontrada no DEADS (Departamento de Administração da Sede) e assim é possível ter acesso a sala.

Devido existência de equipamentos e maquinas de valores consideráveis teve-se a ideia de reforçar a segurança do lugar. Certificando-se assim a apenas a entrada das pessoas autorizadas a este local como: professores da área e alunos estagiários e envolvidos em trabalhos de eletrônica.

2.3 A Solução Apresentada

O novo esquema proposto se utilizará um sistema eletromecânico onde a porta só vai abrir após uma combinação assertiva de código de barra e senha numérica de 4 dígitos correta. Após a combinação correta à tranca eletromecânica é desativada liberando assim a entrada da sala.

2.4 Controle de Acesso Microcontrolado com Código de Barras e Senha

Na figura 2 apresenta-se o circuito elétrico do sistema que foi desenvolvido inicialmente, este sistema foi baseado em um editorial da Revista Saber Eletrônica[1].

Todo o controle é feito pelo microcontrolador PIC16F876-I/P. O PIC controla um display de cristal liquido (LCD), um teclado matricial com 16 teclas (4x4), um relé, dois *leds*, um *buzzer* (sirene) e dois pontos de comunicação: um com o leitor de cartões de código de barras e outro com o PC. A comunicação com o PC não é um objetivo deste trabalho de TCC.

O *buzzer* foi ligado ao canal CCP (Compare, Capture e PWM) do microcontrolador. Desta maneira, não é necessário programar via software a oscilação

[1] Revista Saber Eletrônica – www.sabereletronica.com.br

do mesmo. O LCD e o teclado são controlados por meio da mesma porta I/O do microcontrolador (porta B). Esse controle é feito com a ajuda do CI4, um 75HC573. Este CI é um *lacth* de oito bits. Assim os dados do display são isolados do teclado e vice-versa com o auxilio deste CI.

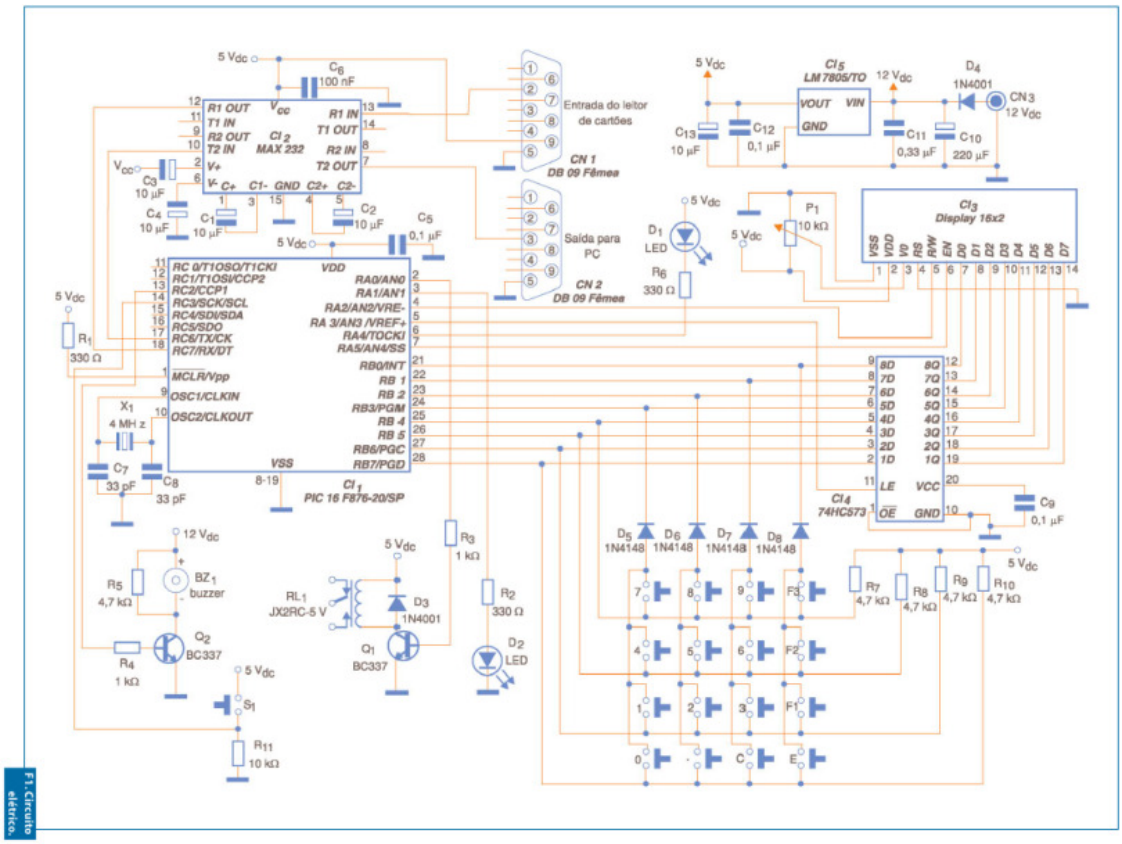


Figura 2: Circuito elétrico do sistema proposto
fonte: saber eletrônica

Um detalhe importante sobre o circuito é sua comunicação serial (RS-232). Aproveitamos o pino RX do microcontrolador para receber os dados do leitor de cartão e o pino TX para enviar dados para o PC (não implementado neste trabalho). Dessa maneira, com uma única porta de comunicação (TX/RX) pode-se implementar a comunicação com dois dispositivos distintos. CN1 é a entrada para o leitor de cartões de código de barras e CN2 a saída para comunicação com o PC. CI2 é um *driver* para comunicação RS-232 (max232). A alimentação (5 VDC) para o leitor de códigos de

barras é retirada do circuito através do pino 9 de CN1. S1 é uma chave de tipo magnética “NA”. Esta chave serve para informar ao sistema o estado da porta controlada, se aberta ou fechada.

O sistema ainda informará o fato a um PC, se o mesmo estiver conectado. O *buzzer* BZ1 deve ser do tipo sem oscilador interno. Sua tensão de trabalho deve estar entre 3 e 30 volts. X1, C7 e C8 formam o “circuito de oscilação”. C12 é um regulador de tensão para 5 VDC e os capacitor C10, C11, C12 e C13 formam os filtros necessários para a mesma. O diodo D4 protege o circuito contra uma possível inversão de polaridade da fonte externa, quando a mesma for ligada ao circuito. Esta fonte de alimentação deve fornecer uma tensão, previamente regulada e filtrada, entre 9 VDC e 12 VDC. P1 regula o contraste do LCD. Os *leds* D1 e D2 devem ter cores diferentes, conforme indicado na lista de materiais vermelho e verde (Soares, 2011).

2.5 O Microcontrolador

Os microcontroladores (também denominados MCU ou µC) são dispositivos eletrônicos utilizados no controle de processos lógicos.

Além dos componentes lógicos e aritméticos usuais de um microprocessador de uso geral, os microcontroladores integram elementos adicionais, tais como, memória RAM (*Random Access Memory* ou Memória de Acesso Aleatório), EEPROM (*Electrically Erasable Programmable Read Only Memory*) ou Memória flash para armazenamento de dados ou programas, dispositivos periféricos e interfaces de I/O (Input/Output) que podem ir de um simples pino digital do componente a uma interface USB ou Ethernet nos mais avançados (Muniz, 2009).

2.5.1 O PIC 16F876

O PIC (*Peripherals Integrated Controller* – Controlador Integrado de Periféricos) é um circuito integrado produzido pelo *Microchip Technology* Inc., que pertence à categoria dos microcontroladores, ou seja, um componente integrado que em um único dispositivo contém todos os circuitos necessários para realizar um completo sistema digital programável.

O PIC dispõe de todos os dispositivos típicos de um sistema microprocessado, ou seja, uma CPU (*Central Processor Unit* ou Unidade de Processamento Central) cuja finalidade é interpretar as instruções de programa; Uma memória PROM (*Programmable Read Only Memor*y ou Memória Programável Somente para Leitura) na qual serão armazenadas de maneira permanente as instruções do programa; uma memória RAM utilizada para memorizar as variáveis utilizadas pelo programa; uma série de Portas de I/O para controlar dispositivos externos ou receber pulsos de sensores, chaves, e outros; uma série de dispositivos auxiliares ao funcionamento, como gerador de clock, bus, contador, etc.

A presença de todos estes dispositivos em um espaço extremamente pequeno, dá ao projetista ampla gama de trabalho e enorme vantagem em usar um sistema microcontrolado, onde em pouco tempo e com poucos componentes externos pode-se fazer o que seria oneroso fazer com circuitos tradicionais.

O PIC 16F876, é um CI (Circuito Integrado) que possui 28 pinos cada um com uma ou mais funções bem definidas, e a cada pino é associado um nome que lembra a sua função. A definição dos terminais do PIC16F876 é apresentada na figura 3.

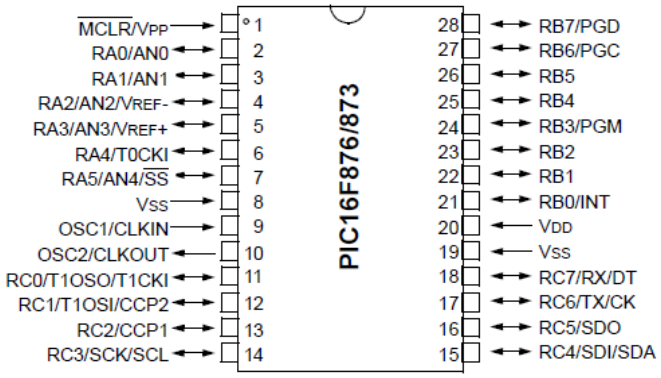


Figura 3: Pinagem do PIC 16F876

Fonte: *Datasheet* do microchip PIC 16F876

A estrutura interna do PIC 16F876 é detalhada através do diagrama de blocos apresentado na Figura 4, que mostra todos os periféricos e comunicações que compõem o mesmo.

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F873	4K	192 Bytes	128 Bytes
PIC16F876	8K	368 Bytes	256 Bytes

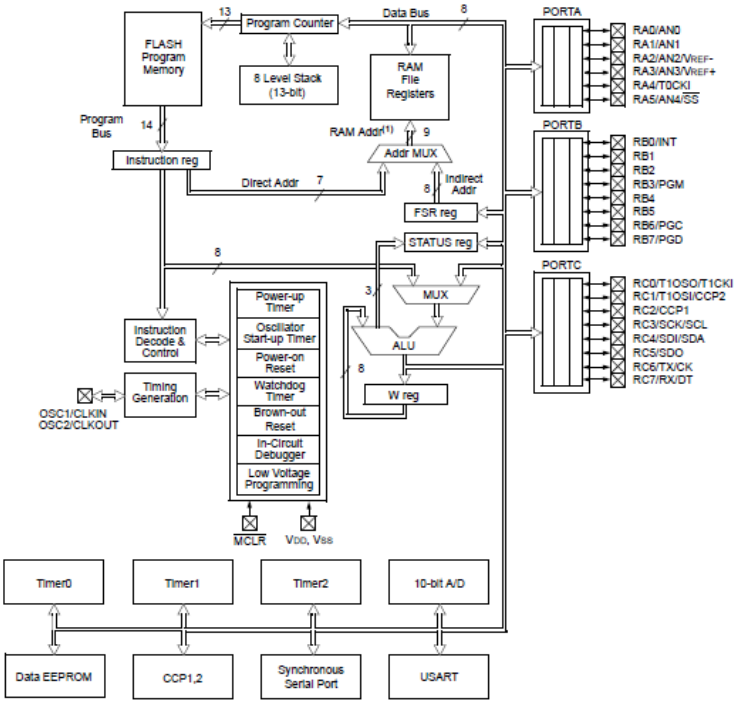


Figura 4: Diagrama de blocos

Fonte: *Datasheet* PIC 16F876

Através do diagrama de blocos podem-se visualizar as diversas partes que compõem o microcontrolador, tais como:

- ULA (Unidade Lógica e Aritmética): é responsável pela execução das operações lógicas e aritméticas;
- Memória de Programa FLASH: é responsável pelo armazenamento do programa que será executado pelo microcontrolador;
- Memória de Dados RAM: utilizada para armazenar as variáveis de programa durante a sua execução;
- PORTs: são as portas de I/O que vão de PORTA a PORTE;
- Timers: são contadores que podem ser acessados diretamente na memória o PIC possui TIMER0, TIMER1 e TIMER2;
- A/Ds: conversores A/D (Analógico/Digital) responsáveis pela conversão de sinais analógicos em sinais digitais;

- CCP: é possível acessar três recursos diferentes através dos módulos CCP (Compare, Capture e PWM);

Portas de comunicação serial: SSP (Synchronous Serial Port) e USART. Através delas que é possível a comunicação do PIC com um PC via padrão RS-232.

2.6 O Display LCD

O LCD (*Liquid Crystal Display* – Display de Cristal Líquido) é um elemento fundamental para que os equipamentos eletrônicos tornem-se mais compactos, interativos e de fácil operação, pois pode facilitar a forma como o usuário vai interagir com o equipamento e, consequentemente, aumentar o valor agregado a ele. Existe uma variedade de displays LCD no mercado, desde os LCDs capazes de exibir uns poucos caracteres até LCDs gráficos, coloridos, e nos mais variados tamanhos.

Os LCDs caracteres normalmente são compatíveis com o código ASCII, e podem gerar letras, números e caracteres especiais, além de caracteres europeus e gregos.

Os LCDs caracteres podem possuir ainda uma memória RAM interna que permite criar caracteres especiais ou símbolos que podem ser imprescindíveis numa determinada aplicação.

O LCD caractere 16x2, ou seja, dezesseis caracteres por duas linhas é um dos LCDs mais utilizados em equipamentos eletrônicos.

A Figura 5 mostra a função de cada um dos 16 pinos do LCD caractere 16 x 2. Os dois primeiros pinos (1 e 2) são relativos a alimentação do componente e devem ser ligados a uma tensão de alimentação, que fica compreendida entre e , sendo 5 o valor ideal.

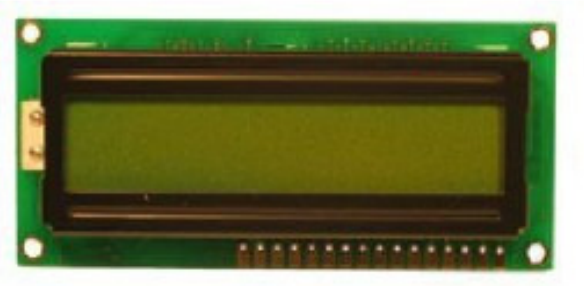


Figura 5: Display LCD 16x2
Fonte: (Muniz, 1999)

Os pinos 15 e 16 acionam um conjunto de LEDs (*Light Emitting Diode* – Diodo Emissor de Luz) responsáveis pela iluminação do painel (*backlight).* Sua alimentação é feita aplicando-se 5 VDC nesses pinos (Figura 6).

Pino	Símbolo	Função	Pino	Símbolo	Função
1	V_{SS}	Comum	9	DB2	Dado
2	V_{DD}	5V	10	DB3	Dado
3	V_0	Ajuste do contraste	11	DB4	Dado
4	RS	Seleção de registro	12	DB5	Dado
5	R/W	Leitura/Escrita	13	DB6	Dado
6	E	Inicia ciclo R/W	14	DB7	Dado
7	DB0	Dado	15	A	Anodo
8	BD1	Dado	16	K	Catodo

Quadro 1: Função Pinos
Fonte: (Muniz, 1999)

O pino três (V_0) é utilizado para que seja possível fazer ajuste no contraste da imagem exibida no Display. Para isso, liga-se esse pino ao centro de um potenciômetro de 10kΩ, para que se possa obter uma tensão variável entre e V_{SS} e $V_{DD.}$

O pino RS (*Register Select*) é utilizado para informar ao LCD o tipo de informação que se encontra no barramento de dados (DB7:DB0).

Para dar início a um ciclo de leitura ou de escrita, é necessário aplicar um pulso no pino 6 (E – *Enable*). No pino E, que normalmente é mantido em nível lógico 0, deve

ser aplicado nível lógico 1 e depois aplicado nível lógico 0. Quando for efetuada uma escrita no LCD, é necessário que o dado se mantenha por alguns nano segundos no barramento depois do pulso aplicado no pino E. Isso é necessário porque o LCD efetua a leitura do dado na borda de descida do pulso aplicado ao pino E. No caso de leitura no LCD, alguns nano segundos após ser aplicado nível lógico um no pino E, o dado estará disponível no barramento para ser lido pelo microcontrolador. Depois do dado lido, no pino E pode ser aplicado novamente nível lógico 0.

Os pinos de DB0 a DB7 equivalem ao barramento de dados paralelo. Este é um barramento bidirecional, pois ele pode ser efetuado tanto para a escrita quanto para leitura dos dados armazenados na memória RAM do LCD. Apesar de existirem oito vias de dados, esses displays também podem operar com quatro vias (DB4 a DB7), ficando assim as demais vias sem função. Neste caso, as informações são enviadas em dois pacotes de quatro bits cada um (Muniz, 1999).

2.7 Relé

Os relés são dispositivos comutadores eletromecânicos. O que determina a utilização de um relé numa aplicação prática são suas características. O entendimento dessas características é fundamental para a escolha do tipo ideal. A bobina de um relé é enrolada com um fio esmaltado cuja espessura e número de voltas é determinado pelas condições em que se deseja fazer sua energização. A intensidade do campo magnético produzido e, portanto, a força com que a armadura é atraída depende tanto da intensidade da corrente que circula pela bobina como do número de voltas que ela contém. O número de contatos e sua disposição vão depender das aplicações a que se destinam os relés. Têm-se então diversas possibilidades:

- Contatos NA ou Normalmente Aberto;
- Contatos NF ou Normalmente Fechado;
- Contatos NA, NF ou Reversíveis;
- Reles aberto, fechados ou selados.

2.8 Buzzer (Sirene)

Dentro do buzzer existem duas camadas de metal e uma camada interna de cristal piezoelétrico, quando é enviada corrente através do buzzer a camada interna vibra na mesma frequência reproduzindo assim um som, por exemplo, se enviarmos corrente em uma frequência de 440Hz, ouviremos o som da nota Lá. Quanto maior a corrente maior o “volume” do som.

O Buzzer tem apenas dois conectores, um positivo (vermelho) e um negativo (preto).

2.9 Leitor de Código de Barras - MinyScan

O O MinyScan é um leitor manual de documentos com código de barras. É utilizado, com grande vantagem, em aplicações que exigem a captura rápida e eficiente de informações dos boletos de pagamentos e das contas de concessionárias públicas (contas de telefone, luz, água etc.).

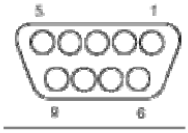
A utilização de sensor especial de altíssima resolução permite a leitura de código de barras de alta densidade, sem ser afetado por pequenas falhas. Possui as seguintes especificações:

- Peso do leitor – 165 g sem o cabo;
- Tensão de alimentação – 5 VDC +- 5%;
- Consumo – 100 mA;
- Temperatura de Operação – 5° a 40°;
- Dimensões A x C x L (mm) 50 x 120 x 65;
- Interface* Serial RS-232-C, Teclado ou USB;
- Temperatura de Armazenagem - 10o a 50o C;
- Umidade de Operação 20% a 80% UR sem condensação;
- Umidade de Armazenagem 20% a 90% UR sem condensação;
- Velocidade de Transporte 30 cm/s a 150 cm/s;
- Tempo de processamento Código Barras 80 ms;

- Número máximo de caracteres 70 caracteres;
- Especificação do Código de Barras 2 de 5 Intercalados, CODE 39 e 128.

A Figura 8 apresenta a interface de comunicação Serial tipo RS-232 com a pinagem do plugue de 9 pinos.

A Figura 9 apresenta as configurações originais do leitor de Código de Barras. Para alterar estas configurações é necessário passar um cartão padrão fornecido pelo fabricante e em seguida passar os cartões referentes a cada característica, tais como: Taxa de transmissão, número de bits, paridade etc.



Conector DB-9 Fêmea

PINO	SINAL
2	Transmissão
3	Recepção
4	Curto com pino 6
5	Terra
6	Curto com pino 4
7	Curto com pino 8
8	Curto com pino 7

Figura 6: Interface da comunicação.

Fonte: Manual do Leitor de Código de Barras - MinyScan

Figura 7: Leitor de Código de Barras

INTERFACES	PARÂMETROS	VALORES DA FÁBRICA
RS-232-C	Baud Rate	2400 bps
	Tamanho do Byte	7 bits
	Paridade	Impar
	Stop Bits	2 Stop Bits
	Protocolo	STX-ETX-BCC com ACK
Teclado e USB	Velocidade	140 toques por segundo
	Com ou sem “enter” no fim	Com “enter”
	Temporização	Padrão
RS-232-C, Teclado e USB	Com ou sem Tab no fim	Sem Tab
	Identificador do código	Sem
	Check digit do cheque	Não
	Caracter do campo com check digit errado	“?”
	Mensagem de erro de leitura	Sem
	Formato de transmissão de dados	Leitura

Figura 8: Configurações

Fonte: Manual do Leitor de Código de Barras – MinyScan

2.10 Fluxograma do Sistema

A Figura 10 apresenta o fluxograma simplificado do sistema.

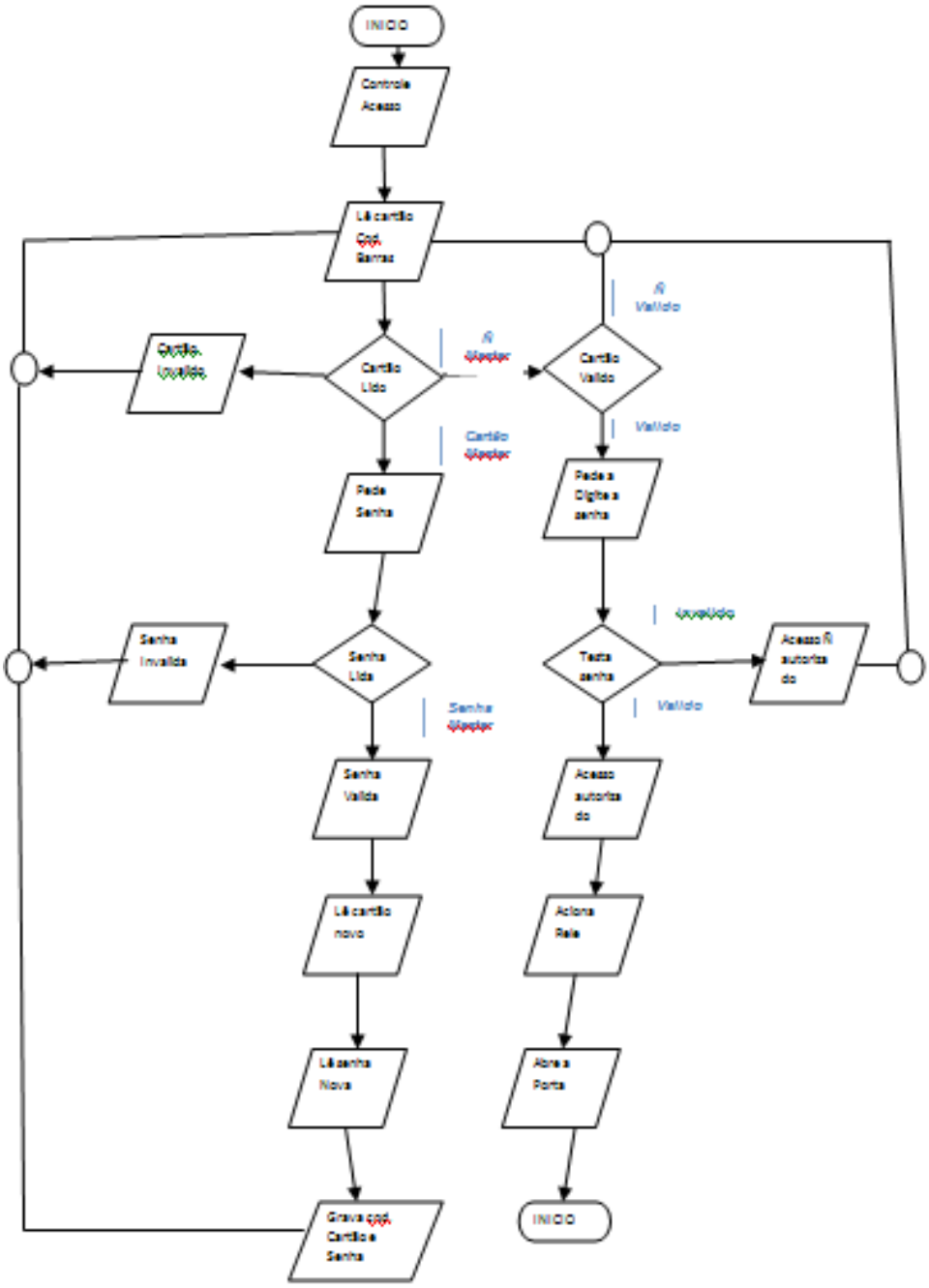


Figura 9: Fluxograma do Sistema

3. DESENVOLVIMENTO DO SISTEMA

3.1 Projeto da Placa de Circuito Impresso

O software PROTEUS (versão Trial) (*Ares, Isis*) permite criar circuitos eletrônicos e simular os mesmos. Através deste software podem-se identificar possíveis erros de esquemática. A figura 11 nos mostra o layout da placa de circuito impresso gerado pelo ISIS.

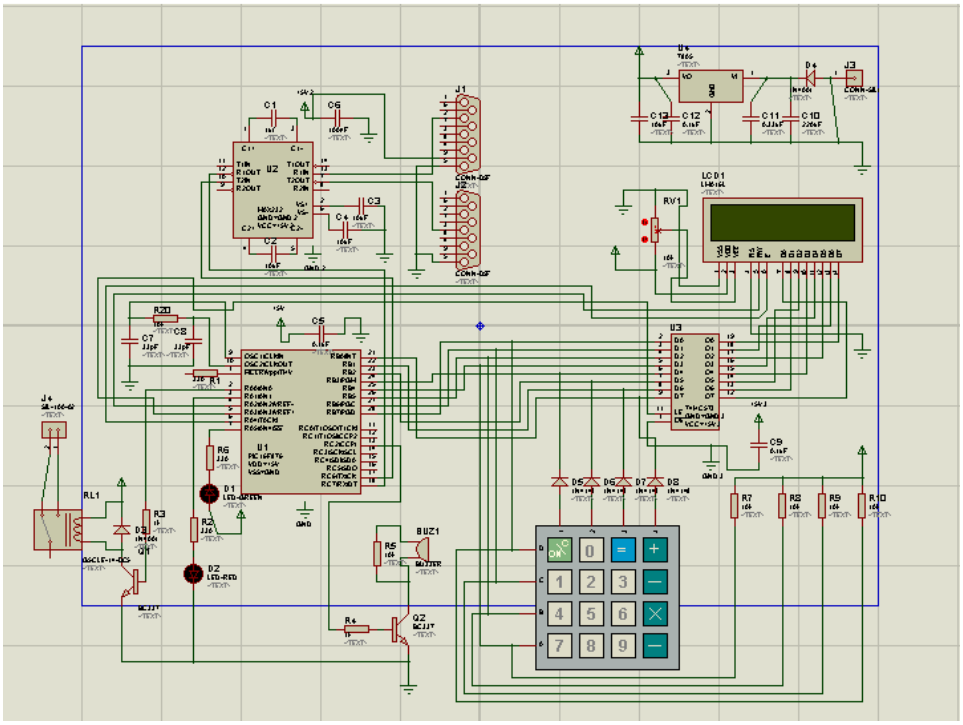


Figura 10: Esquemática gerado no Isis

Após feito o esquemática no ISIS fez-se a transferência do mesmo para o ARES onde organizou-se os componentes e trilhas da melhor forma possível, conforme figura 12 abaixo.Depois de os componentes organizados, realizou-se a impressão da mesma em PDF como mostra a figura 13.

Percebeu-se então que as trilhas estavam finas o que ocasionaria problemas na passagem de corrente. Assim sendo decidiu-se utilizar a esquemática sugerida pela revista saber eletrônica conforme figura 14.

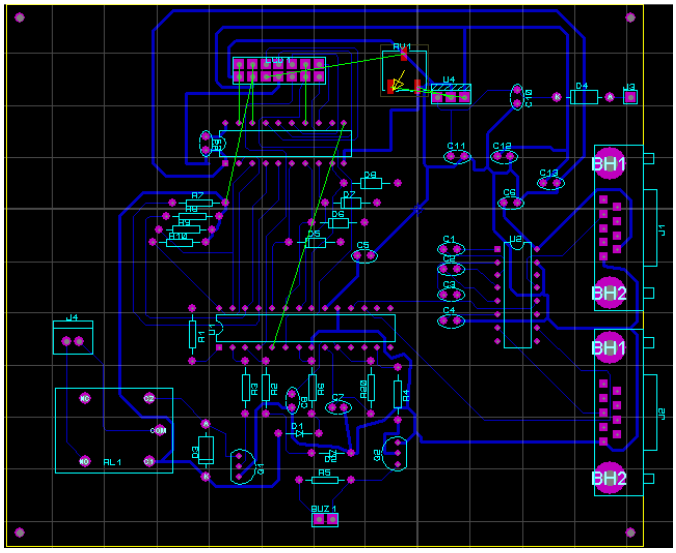


Figura 11: Layout feito no Ares

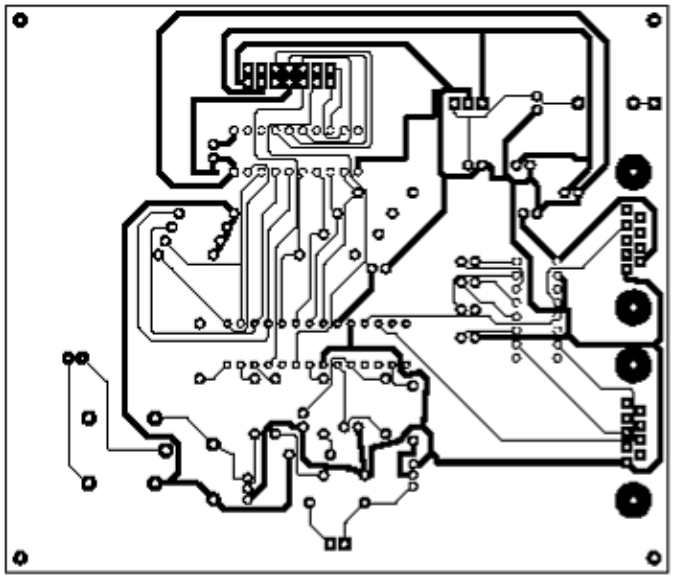


Figura 12: Layout Placa

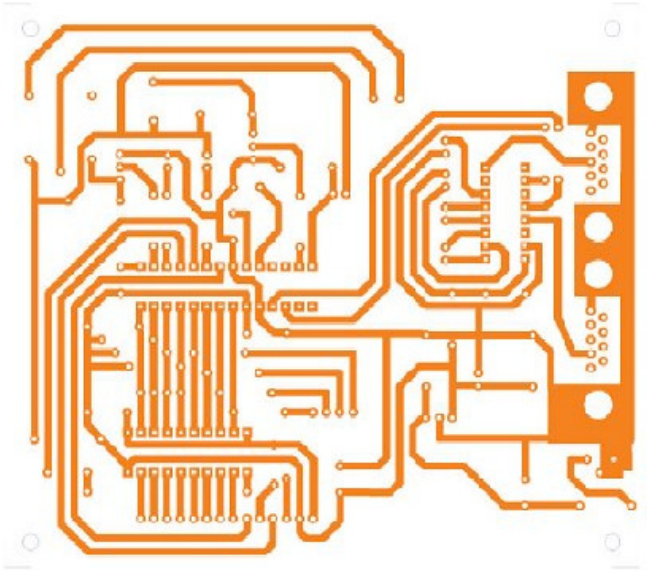


Figura 13: Layout placa sugerida pela saber eletrônica
Fonte: Soares, 2011

3.2 Confecção da Placa de Circuito Impresso

Com o auxilio de uma palha de aço, fez-se a limpeza da placa de fenolite para retirada de qualquer substancia aderente na placa, após a mesma foi lavada com água corrente e passou-se álcool, conforme figura 15.

Figura 14: Limpeza da placa

Para transferência térmica, utilizou-se o método de fixação da tinta por aquecimento sobre contato entre a placa de fenolite e o papel *Glossy Paper.*

Normalmente utiliza-se ferro de passar roupa para a transferência térmica, porém utilizou-se o equipamento confeccionado pelo Professor da UTFPR Alberto Nocoru Miyadaira, onde o aquecimento se da por uma lâmpada halógena, e um cilindro revestido de uma borracha de alta resistência térmica, auxilia no deslizamento da placa facilitando a transferência de calor em toda superfície da mesma, fazendo com que a figura do circuito tenha uma ótima aderência com o fenolite. A figura 16 apresenta o sistema de aquecimento.

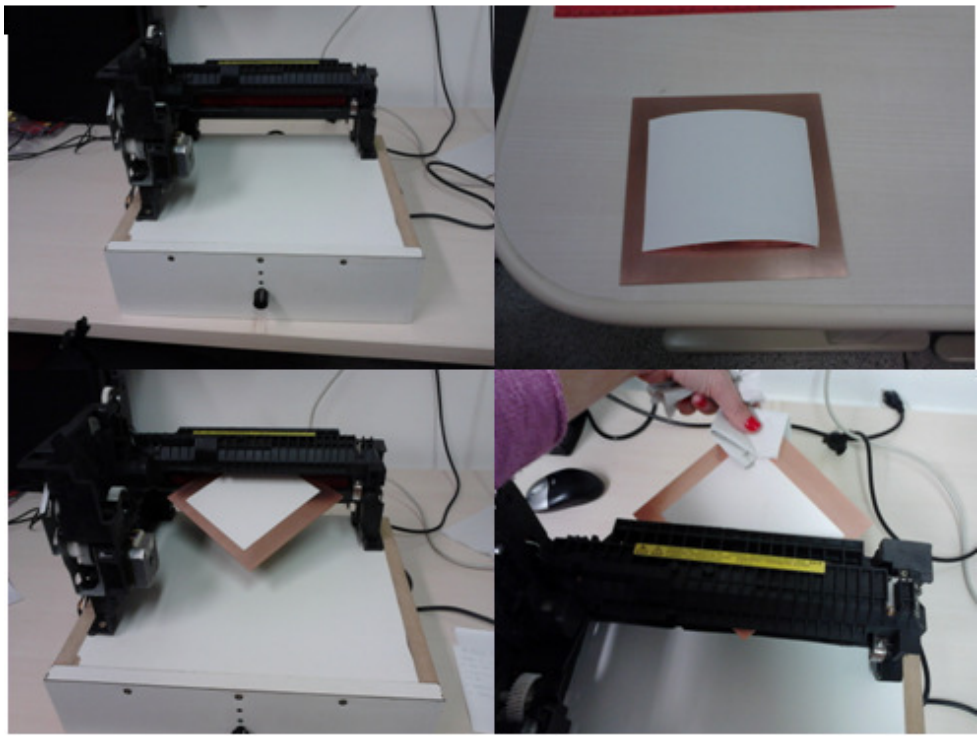


Figura 15: Transferência Térmica

Para retirada do papel Glossy Paper da placa de fenolite utilizou-se uma bacia com água quente e um pouco de detergente liquido, este processo é bastante demorado, pois o papel deve ser retirado com muito cuidado para que não ocorra rompimento de trilhas durante o processo, conforme mostra a figura 17.

Figura 16: Processo de retirada do Papel Glossy Paper

Para a corrosão da placa de fenolite, utilizou-se uma bacia onde foi misturada uma solução de percloreto de ferro já preparada diluído em água, deixou-se a placa mergulhada na mistura, até que toda a parte de fenolite que não haviam tinta fossem corroídas, sobrando somente as trilhas (Figura 18).

Figura 17: Corrosão da placa de fenolite

Conforme mostra a figura 19, depois de corroída a placa fez-se a lavagem da mesma em água corrente e a secagem ao ar livre.

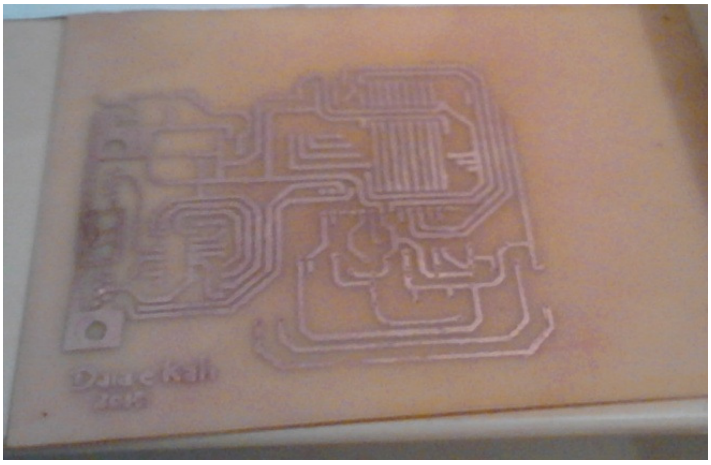


Figura 18: Placa pronta para furação

Para a soldagem da placa foi necessário fazer a furação da mesma, com o auxilio de um perfurador manual de placa para circuito impresso. Conforme mostra a figura 20, fez-se a pré-colocação dos componentes na placa cuidando sempre o posicionamento do mesmo conforme sua pinagem.

Depois de conferido o posicionamento da pinagem de cada componente, fez-se a soldagem, conforme a figura 21.

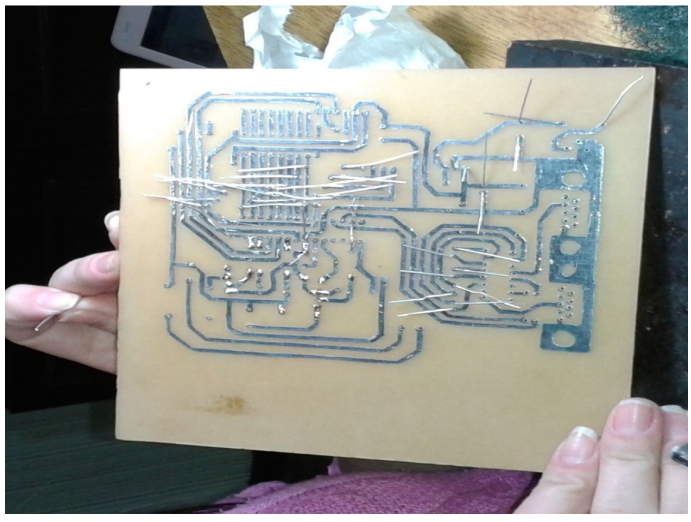


Figura 19: Encaixe de componentes

Figura 20: Soldagem dos componentes

3.3 Teste da Placa de Circuito Impresso

Na Na conferencia das trilhas, percebeu-se a falta de uma ligação que interrompia a passagem de corrente, assim sendo, corrigiu-se o projeto conforme mostra a figura 22.

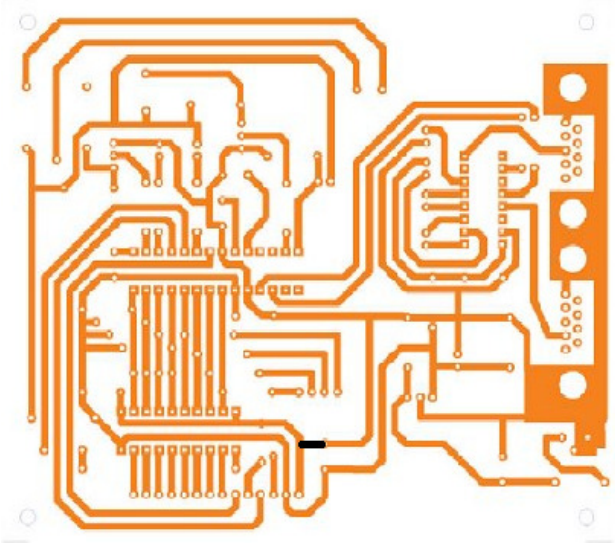


Figura 21: Layout placa 2 corrigida

Após a correção, refez-se todo o processo de transferência térmica, corrosão, furação e soldagem dos componentes na placa.

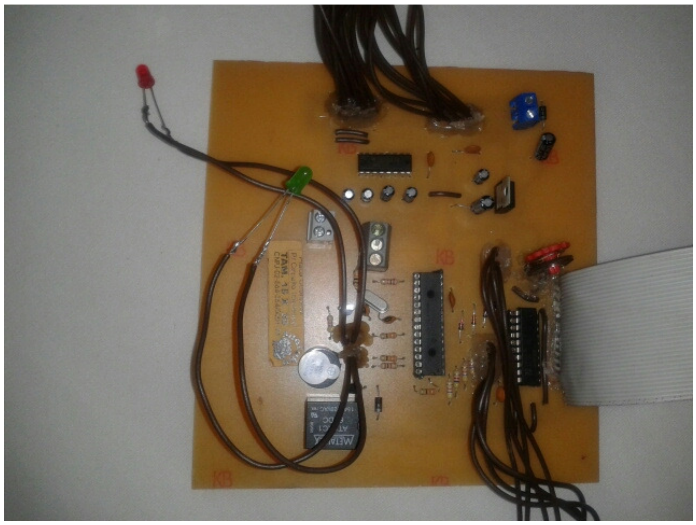


Figura 22: Placa soldada pronta pra teste

Com o auxilio do multímetro, fez-se o teste de continuidade no teclado soldado na placa, verificou-se que o mesmo estava em curto circuito, pois os cabos estavam mal posicionados.

Para resolver o problema, perfurou-se a placa o mais próximo possível de cada trilha, e refez-se a solda, cuidando para que as mesmas não entrassem em contato com nenhuma parte da placa, a não ser sua respectiva trilha (Figura 24).

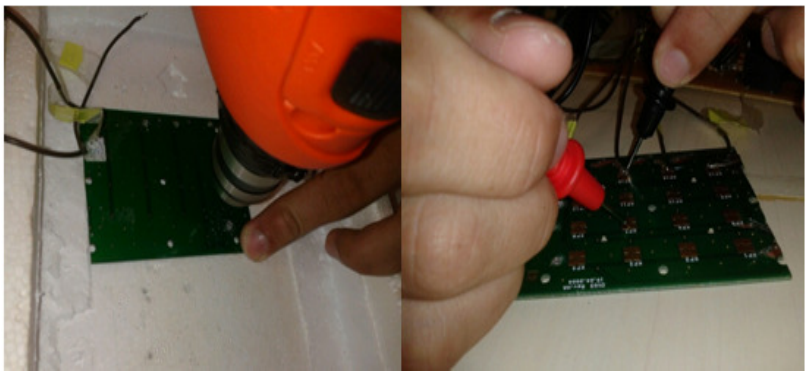


Figura 23: Furação e soldagem do teclado

Em seguida se refez o teste de continuidade da placa e percebeu-se que a mesma não se encontrava mais em curto.

Posteriormente fez-se o teste do display LCD, verificou-se que o mesmo ligava sem nenhum problema, mas não escrevia na tela a mensagem esperada. Percebeu-se então que o problema não erra ocasionado pelo display e sim pelo PIC que não mandava o comando ao mesmo.

Após inúmeros testes sem obter êxito e devido ao prazo final de apresentação do TCC estar próximo, decidiu-se utilizar o Kit de desenvolvimento MacLab 3 da Mosaico, para fins de testes e funcionamento do sistema.

Analisando os possíveis problemas chegou-se a conclusão de que não havia mais problemas na placa, mas sim na gravação do PIC, e como não havia modo de testar a programação do PIC, pois a mesma foi desenvolvida pela Revista Saver Eletrônica, decidiu-se então a fazer testes na placa MACLAB 3.

3.4 O Kit MACLAB 3

A placa MACLAB 3 possui um microcontrolador PIC 18F4550 e um conjunto de periféricos que nos permite simular diversas situações (Figura 25). O programa pode ser desenvolvido e testado na placa e assim após o funcionamento do mesmo pode-se confeccionar a placa de circuito impresso do sistema (SOUSA, 2012).

Figura 24: Kit de teste MACLAB

O programa foi reescrito no sistema MacLab 3 em linguagem C18 com auxilio do prof. Adriano de Andrade Bresolin (orientador).

Não houve tempo hábil para desenvolver uma nova placa de circuito impresso, mas o sistema funcionou perfeitamente e conforme o previsto inicialmente.

3.5 Programação do Leitor de Código de Barras

O O leitor MinyScan veio como uma configuração original de fabrica, como mostra a figura 9. Esta configuração precisou ser alterada de modo que suprisse os requisitos necessários para ser compatível com o PIC 18F4550, e assim realizar a sua determinada função.

Para que isso pode ser feito utilizamos cartões de programação de cada parâmetro com os valores e dados necessários para seu uso:

Primeiramente, seguindo o passo a passo de configuração fornecido pelo fabricante, indica que passe o (cartão de entrada de programação) conforme figura 26, que vai habilitar a programação de cada parâmetro escolhido.

Figura 25: Cartão de entrada da programação
Fonte: Manual do MinyScan

Posteriormente passaram-se os códigos dos parâmetros necessários de acordo com a compatibilidade do PIC utilizado, como; *baud rate* para (9600 bts), tamanho do bit para (8 bits), *stop bits* para (1 stop bits), paridade (sem paridade) não precisando assim passar nenhum cartão.

Todos os códigos foram passados um por vez, sem que fosse necessário seguir uma sequencia exata, da ordem dos parâmetros. Para a análise de cada código de determinada função escolhida pode-se observar a figura 27.

Figura 26: parâmetros de configuração do MinyScan
Fonte: Manual do MinyScan

Uma vez escolhidos todos os parâmetros necessários para o leitor, finalizou-se os a configuração com o cartão (fechar programação) como mostra a figura 28.

Com o leitor já configurado testou-se o mesmo utilizando cartões/crachás de alunos e professores, ambos foram reconhecidos e a leitura da numeração foi realizada e visualizada pelo *hyper terminal* programa de comunicação fornecido pelo Windows.

Figura 27: cartão de enceramento de manutenção
Fonte: Manual do MinyScan

4. CONSIDERAÇÕES FINAIS

A solução apresentada propõe um sistema de acesso microcontrolado por código de barras para acesso a ambientes restritos.

Durante o desenvolvimentos do projeto, surgiram diversas dificuldades relacionadas a compra de componentes, lógica de programação, funcionamento de alguns componentes principalmente do PIC 16F876 I/P, que após varias simulações e pesquisas foram solucionadas fazendo com que o nosso projeto funcionasse como o esperado. O leitor de código de barras com acesso serial levou cerca de 5 meses para ser adquirido.

A placa de circuito impresso desenvolvida originalmente mostrou-se ineficaz. O circuito elétrico apresentava vários erros de lógica e de alimentação. O programa gravado não funcionou e o display não apresentava nenhuma das mensagens descritas esperadas.

Deste modo, optou-se pela simulação do sistema através do Kit MacLab 3, onde o programa foi reescrito em linguagem C18, pois a original estava em linguagem C tipo CSS.

O sistema funcionou conforme o previsto no Kit MacLab3. Os custos apresentados no desenvolvimento deste sistema foram em torno de R$ 430 (quatrocentos e trinta reais).

Como trabalho futuro sugere-se a confecção da placa de circuito impresso do sistema testado no Kit MacLab 3. Sugere-se ainda utilizar a comunicação serial (canal 2) para fazer a comunicação com um Microcomputador, deste modo podem-se ampliar as funções do sistema passando a ter controles de horários de acesso por exemplo.

REFERÊNCIAS

MICROCHIP PIC 16F876 – Datasheet. Disponível em: <http://www.microchip.com/>. Acesso em: 29 mar. 2013.

MINYSCAN: Leitor Manual de Código de barras – Manual do Usuário. Versão 3.3. Disponível em: <http://cis.com.br/home/wp-content/uploads/2011/07/minyscan-rev3-3.pdf>. Acesso em: 29 mar. 2013.

MUNIZ, Laurentino B. **Sistema de Controle Digital da Temperatura da Água do chuveiro Elétrico**. Jun. 2009. Disponível em: <http://aldeia3.computacao.net/greenstone/collect/trabalho/index/assoc/HASH01cd.dir/doc.pdf>. Acesso em: 10 jan. 2013

OLIVEIRA, Jonatas. **Tocando som com o Arduino**. Nerd Who?. 07 set. 2011. Disponível em: <http://www.nerdwho.com/blog/42/tocando-som-com-o-arduino/>. Acesso em 12 fev. 2013

SOARES, Márcio J. **Controle de Acesso Microcontrolado com Código de Barras**. Saber Eletronica, 10 out. 2011. Disponivel em: <http://www.sabereletronica.com.br/secoes/leitura/1876#>. Acesso em: 25 out. 2012.

SOUSA, Daniel R.; SOUZA, Davi J. **Desbravando o Microcontrolador PIC18: PIC 18F1220.** Ed. São Paulo: Érica, 2012.

ANEXOS

ANEXO A – PROGRAMA DE CONTROLE DE ACESSO DESENVOLVIDO NO KIT MCLAB3 DO PIC18F4550

C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Descrição geral *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */



/* -------------------------------------------------------------------- *
* -------------------------------------------------------------------- *
* DEFINIÇÃO PIC *
* -------------------------------------------------------------------- *
* -------------------------------------------------------------------- */

#include <p18F4550.h> //Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <pwm.h> //PWM library functions
#include <adc.h> //ADC library functions
#include <timers.h> //Timer library functions
#include <delays.h> //Delay library functions
#include <i2c.h> //I2C library functions
#include <stdlib.h> //Library functions
#include <usart.h> //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Configurações para gravação *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF
#pragma config XINST = OFF
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Definição e inicialização das variáveis globais *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
//Este programa não utiliza nenhuma variável de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Constantes internas *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
//Este programa não utiliza nenhuma constante de usuário
//A definição de constantes facilita a programação e a manutenção.
#define TAM 27 //tamanho das matrizes de mensagem
#define TBUF 30 //tamanho máximo do buffer de entrada do leitor

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Declaração dos flags de software *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
//Este programa não utiliza nenhum flag de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Definição e inicialização dos port's *
1
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */



/* ------------------------------------------------------------------- *
* ------------------------------------------------------------------- *
* PROTOTIPAGEM DE FUNÇÕES, VARIAVEIS E CONSTANTES *
* ------------------------------------------------------------------- *
* ------------------------------------------------------------------- */
//-- Funcoes de controle do LCD
void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);

//-- Funcoes de Controle do TECLADO
void tecla_precionada(void);
void le_teclado(void);

//-- Funcoes de Mensagens na Tela
void tela_principal(void);
void tela_passecartao(void);
void tela_cartaomaster(void);
void tela_cartaoinvalido(void);
void tela_senhaconfere(void);
void tela_senhanaoconfere(void);
void tela_passecartaonovo(void);
void tela_cartaonovook(void);
void tela_gravacaoook(void);
void tela_cartaovalido(void);
void tela_erro(void);
void tela_acessoautor(void);
void tela_acessonaoautor(void);

//-- Funcoes de controle e processamento Cod. Barra
void le_cartao(void);
int cartaomaster(void);
```

```
115 int senhamaster(void);
116 int valida_cartao(void);
117 int valida_senha(void);
118 void grava_cartaosenha(void);
119
120 //-- Funcões de memoria
121 void write_EEPROM(unsigned char endereco, unsigned char dado);
122 unsigned char read_EEPROM(unsigned char endereco);
123 //void write_FLASH(unsigned addr, char *buffer);
124 //unsigned short read_FLASH(unsigned endereco);
125
126
127 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
128 * Definição e inicialização das variáveis *
129 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
130 //Neste bloco estão definidas as variáveis globais do programa.
131 unsigned char filtro; // Filtro para teclas
132 unsigned char num_linha; // Armazena o número da linha ativada
133 unsigned char num_coluna; // Armazena o número da coluna
134
135 //-- variaveis acrescidas leitura codigo barras
136 unsigned char bufferlido[25];
137 unsigned char senhalida[25];
138 unsigned char cbarranew[25];
139 unsigned char csenharead[25];
140
141 unsigned char tecla_digito;
142 unsigned char contabarra;
143 unsigned char contasenha;
144 unsigned char cbarra[8]; // variavel para armazenar codido barra lido
145 unsigned char csenha[4]; // variavel para armazenar codido barra lido
146 unsigned char carac_temp;
147 int i, cont;
148
149 // Variáveis de Memoria
150 char nc;
151 int tamanho;
152 long int endmemoria;
2
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
153 unsigned char cont_mem_dados1= 0;
154 unsigned char cont_mem_dados2= 0;
155 unsigned short cont_mem_prog;
156
157 //-- variavel para controle do fluxo do programa
158 unsigned char fluxo_programa;
159 unsigned char fluxo_cmaster;
160
161
162 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
163 * Constantes internas *
164 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
165 //A definição de constantes facilita a programação e a manutenção.
166 //-- Constante que define numero do cartao master
167 const unsigned char cmaster[]={"87654321"}; //declaracao em forma de numeros
168 const unsigned char smaster[]={'4','5','6','7'}; //declaracao em forma de carac.
169
170
171
172 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
173 * Declaração dos flags de software *
174 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
175 //A definição de flags ajuda na programação e economiza memória RAM.
176 union{
177 struct{
```

```
178 unsigned BIT0:1;
179 unsigned BIT1:1;
180 unsigned BIT2:1;
181 unsigned BIT3:1;
182 unsigned BIT4:1;
183 unsigned BIT5:1;
184 unsigned BIT6:1;
185 unsigned BIT7:1;
186 }; //ARMAZENA OS FLAGS DE CONTROLE
187 unsigned char coluna_teclado;
188 }coluna_ativa;
189
190 #define coluna1 coluna_ativa.BIT0 /* LINHA_ATIVA,4, BIT 4 DO REGISTRADOR LINHA_ATIVA
191 REPRESENTA A LINHA 1 DO TECLADO 4x4
192 1 -> LINHA ESTÁ ATIVADA
193 0 -> LINHA ESTÁ DESATIVADA */
194
195 #define coluna2 coluna_ativa.BIT1 /* LINHA_ATIVA,5, BIT 5 DO REGISTRADOR LINHA_ATIVA
196 REPRESENTA A LINHA 2 DO TECLADO 4x4
197 1 -> LINHA ESTÁ ATIVADA
198 0 -> LINHA ESTÁ DESATIVADA */
199
200 #define coluna3 coluna_ativa.BIT2 /* LINHA_ATIVA,6, BIT 6 DO REGISTRADOR LINHA_ATIVA
201 REPRESENTA A LINHA 3 DO TECLADO 4x4
202 1 -> LINHA ESTÁ ATIVADA
203 0 -> LINHA ESTÁ DESATIVADA */
204
205 #define coluna4 coluna_ativa.BIT3 /* LINHA_ATIVA,7, BIT 7 DO REGISTRADOR LINHA_ATIVA
206 REPRESENTA A LINHA 4 DO TECLADO 4x4
207 1 -> LINHA ESTÁ ATIVADA
208 0 -> LINHA ESTÁ DESATIVADA */
209
210 #define fim coluna_ativa.BIT0 /* REPRESENTA O FINAL DA VARREDURA
211 1 -> LINHA ESTÁ ATIVADA
212 0 -> LINHA ESTÁ DESATIVADA */
213
214 #define coluna coluna_ativa.coluna_teclado
215
216
217
218
219 /* ------------------------------------------------------------------- *
220 * ------------------------------------------------------------------- *
221 * ENTRADAS E SAIDAS DO PROGRAMA *
222 * ------------------------------------------------------------------- *
223 * ------------------------------------------------------------------- */
224
225 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
226 * ENTRADAS *
227 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
228 // As entradas devem ser associadas a nomes para facilitar a programação e
3
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
229 //futuras alterações do hardware.
230 #define linha1 PORTDbits.RD4 /* PINO DE ENTRADA DA LINHA 1
231 1 -> ALGUMA TECLA DESTA LINHA ESTÁ PRESSIONADA
232 0 -> NENHUMA TECLA DESTA LINHA ESTÁ PRESSIONADA
*/
233
234 #define linha2 PORTDbits.RD5 /* PINO DE ENTRADA DA LINHA 2
235 1 -> ALGUMA TECLA DESTA LINHA ESTÁ PRESSIONADA
236 0 -> NENHUMA TECLA DESTA LINHA ESTÁ PRESSIONADA
*/
237
238 #define linha3 PORTDbits.RD6 /* PINO DE ENTRADA DA LINHA 3
```

```
239 1 -> ALGUMA TECLA DESTA LINHA ESTÁ PRESSIONADA
240 0 -> NENHUMA TECLA DESTA LINHA ESTa PRESSIONADA
*/
241
242 #define linha4 PORTDbits.RD7 /* PINO DE ENTRADA DA LINHA 4
243 1 -> ALGUMA TECLA DESTA LINHA ESTÁ PRESSIONADA
244 0 -> NENHUMA TECLA DESTA LINHA ESTÁ PRESSIONADA
*/
245 //-- rele
246 #define RELE1 PORTBbits.RB4 //PORTA DO LED
247 //0 -> DESLIGADO
248 //1 -> LIGADO
249
250
251 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
252 * SAÍDAS *
253 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
254 // As saídas devem ser associadas a nomes para facilitar a programação e
255 //futuras alterações do hardware.
256
257 #define COLUNA_1 PORTDbits.RD0 //PINO PARA ATIVAR COLUNA 1 DO TECLADO
258 //MATRICIAL
259 //1 -> COLUNA ATIVADA
260 //0 -> COLUNA DESATIVADA
261
262 #define rs PORTEbits.RE0 // via do lcd que sinaliza recepção de dados ou
comando
263 #define rw PORTBbits.RB7 // via do lcd que configura leitura ou escrita no
barramento
264 #define enable PORTEbits.RE1 // enable do lcd
265
266
267
268
269 /* ------------------------------------------------------------------------ *
270 * ------------------------------------------------------------------------ *
271 * ROTINAS PARA CONTROLE DO LCD - FUNCOES *
272 * ------------------------------------------------------------------------ *
273 * ------------------------------------------------------------------------ */
274
275 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
276 * Rotina que envia um COMANDO para o LCD *
277 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
278
279 void comando_lcd(unsigned char caracter)
280 {
281 TRISD = 0b00000000; // configura portd como saída
282 rs = 0; // seleciona o envio de um comando
283 PORTD = caracter; // carrega o PORTD com o caracter
284 enable = 1; // gera pulso no enable
285 Delay10TCYx(1); // espera 10 microsegundos
286 enable = 0; // desce o pino de enable
287 TRISD = 0b11110000; // CONFIGURA O PORTD PARA O TECLADO
288 Delay10TCYx(4); // espera mínimo 40 microsegundos
289 COLUNA_1 = 1; // ativa a coluna 1 do teclado
290 }
291
292 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
293 * Rotina que envia um DADO a ser escrito no LCD *
294 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
295
296 void escreve_lcd(unsigned char caracter)
297 {
298 TRISD = 0b00000000; // configura portd como saída
4
```

```
rs = 1; // seleciona o envio de um dado
PORTD = caracter; // carrega o PORTD com o caracter
enable = 1; // gera pulso no enable
Delay10TCYx(1); // espera 10 microsegundos
enable = 0; // desce o pino de enable
TRISD = 0b11110000; // CONFIGURA O PORTD PARA O TECLADO
Delay10TCYx(4); // espera mínimo 40 microsegundos
COLUNA_1 = 1; // ativa a coluna 1 do teclado
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Função para limpar o LCD *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void limpa_lcd(void)
{
comando_lcd(0x01); // limpa lcd
Delay1KTCYx(2);
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Inicialização do Display de LCD *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd(void)
{
rw = 0; // pino rw em 0
comando_lcd(0x30); // envia comando para inicializar display
Delay1KTCYx(4); // espera 4 milisengundos
comando_lcd(0x30); // envia comando para inicializar display
Delay10TCYx(10); // espera 100 microsengundos
comando_lcd(0x30); // envia comando para inicializar display
comando_lcd(0x38); // liga o display, sem cursor e sem blink
limpa_lcd(); // limpa lcd
comando_lcd(0x0c); // display sem cursor
comando_lcd(0x06); // desloca cursor para a direita
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Função para escrever uma frase no LCD *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void escreve_frase(const rom char *frase)
{
do
{
escreve_lcd(*frase);
}while(*++frase);
}
/* -------------------------------------------------------------------- *
* -------------------------------------------------------------------- *
* ROTINAS PARA CONTROLE DA GRAVAÇÃO DA MEMORIA *
* -------------------------------------------------------------------- *
* -------------------------------------------------------------------- */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Função de escrita da EEPROM interna *
```

```
363 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
364
365 void write_EEPROM(unsigned char endereco, unsigned char dado)
366 {
367 EEDATA = dado; //carrega dado
368 EEADR = endereco; //carrega endereço
369
370 EECON1bits.CFGS = 0; //habilita acesso a EEPROM
371 EECON1bits.EEPGD = 0; //aponta para memória de dados
372 EECON1bits.WREN = 1; //habilita escrita
373 INTCONbits.GIE = 0; //desabilita todas as interrupções
374 EECON2 = 0x55; //escreve 0x55 em EECON2 (obrigatório)
5
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
375 EECON2 = 0xAA; //escreve 0xAA em EECON2 (obrigatório)
376 EECON1bits.WR = 1; //inicia a escrita
377 INTCONbits.GIE = 1; //habilita todas as interrupções
378 while(EECON1bits.WR); //aguarda bit WR de EECON1 = 0
379 EECON1bits.WREN = 0; //desabilita escrita
380 }
381
382 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
383 * Função de leitura da EEPROM interna *
384 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
385
386 unsigned char read_EEPROM(unsigned char endereco)
387 {
388 EEADR = endereco; //carrega endereço
389
390 EECON1bits.CFGS = 0; //habilita acesso a EEPROM
391 EECON1bits.EEPGD = 0; //aponta para memória de dados
392 EECON1bits.RD = 1; //habilita leitura
393
394 return(EEDATA); //retorna dado disponível em EEDATA
395 }
396
397
398 /* ------------------------------------------------------------------------ *
399 * ------------------------------------------------------------------------ *
400 * ROTINAS PARA ESCRITA NO LCD- MENSAGENS *
401 * ------------------------------------------------------------------------ *
402 * ------------------------------------------------------------------------ */
403
404 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
405 * Tela Principal - Tela Inicial *
406 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
407
408 void tela_principal(void)
409 {
410 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
411 escreve_frase("CONTROLE ACESSO ");
412 }
413
414 void tela_passecartao(void)
415 {
416 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
417 escreve_frase("PASSE O CARTAO ");
418 comando_lcd(0xC0); // posiciona o cursor na linha 0, coluna 0
419 escreve_frase(" ");
420 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
421 escreve_lcd(0x30 + fluxo_programa);
422 }
423
424 void tela_cartaomaster(void)
425 {
```

```
426 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
427 escreve_frase("CARTAO MASTER! ");
428 comando_lcd(0xC0); // posiciona o cursor na linha 1, coluna 15
429 escreve_frase("SENHA: ");
430 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
431 escreve_lcd(0x30 + fluxo_programa);
432 }
433
434 void tela_cartaoinvalido(void)
435 {
436 limpa_lcd();
437 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
438 escreve_frase("CARTAO INVALIDO ");
439 }
440
441 void tela_senhaconfere(void)
442 {
443 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 5
444 escreve_frase("SENHA VALIDA! ");
445 comando_lcd(0xC0); // posiciona o cursor na linha 0, coluna 0
446 escreve_frase(" ");
447 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
448 escreve_lcd(0x30 + fluxo_programa);
449 }
450
6
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
451 void tela_senhanaoconfere(void)
452 {
453 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 3
454 escreve_frase("SENHA INVALIDA! ");
455 comando_lcd(0xC0); // posiciona o cursor na linha 0, coluna 0
456 escreve_frase(" ");
457 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
458 escreve_lcd(0x30 + fluxo_programa);
459 }
460
461 void tela_passecartaonovo(void)
462 {
463 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
464 escreve_frase("CARTAO NOVO ? ");
465 comando_lcd(0xC0); // posiciona o cursor na linha 0, coluna 0
466 escreve_frase(" ");
467 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
468 escreve_lcd(0x30 + fluxo_programa);
469 }
470
471 void tela_cartaonovook(void)
472 {
473 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
474 escreve_frase("OK CARTAO NOVO ");
475 comando_lcd(0xC0); // posiciona o cursor na linha 0, coluna 0
476 escreve_frase("SENHA: ");
477 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
478 escreve_lcd(0x30 + fluxo_programa);
479 }
480
481 void tela_gravacaoook(void)
482 {
483 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
484 escreve_frase(" GRAVACAO OK! ");
485 comando_lcd(0xC0); // posiciona o cursor na linha 0, coluna 0
486 escreve_frase(" ");
487 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
488 escreve_lcd(0x30 + fluxo_programa);
```

```
489 }
490
491 void tela_cartaovalido(void)
492 {
493 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
494 escreve_frase("CARTAO VALIDO ");
495 comando_lcd(0xC0); // posiciona o cursor na linha 0, coluna 0
496 escreve_frase("SENHA: ");
497 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
498 escreve_lcd(0x30 + fluxo_programa);
499 }
500
501 void tela_erro(void)
502 {
503 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 0
504 escreve_frase("----- ERRO -----");
505 }
506
507 void tela_acessoautor(void)
508 {
509 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 3
510 escreve_frase(" OK! ACESSO ");
511 comando_lcd(0xC0); // posiciona o cursor na linha 1, coluna 0
512 escreve_frase(" AUTORIZADO! ");
513 }
514
515 void tela_acessonaoautor(void)
516 {
517 comando_lcd(0x80); // posiciona o cursor na linha 0, coluna 3
518 escreve_frase(" ACESSO NAO ");
519 comando_lcd(0xC0); // posiciona o cursor na linha 1, coluna 3
520 escreve_frase(" AUTORIZADO! ");
521 }
522
523
524 /* ------------------------------------------------------------------- *
525 * ------------------------------------------------------------------- *
526 * ROTINAS PARA CONTROLE DO TECLADO - FUNCOES *
7
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
527 * ------------------------------------------------------------------- *
528 * ------------------------------------------------------------------- */
529
530 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
531 * Identifica a Tecla Pressionada *
532 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
533
534 void tecla_precionada(void)
535 {
536 //-- Testa Linha 1 com todas as colunas
537 //limpa_lcd();
538 if (num_linha==1)
539 {
540 switch (num_coluna)
541 {
542 case 1:
543 tecla_digito= '1';
544 break;
545 case 2:
546 tecla_digito= '2';
547 break;
548 case 3:
549 tecla_digito= '3';
550 break;
551 case 4:
```

```
552 tecla_digito= '17'; //-- Equivale a letra A;
553 break;
554 }
555 }
556 //-- Testa Linha 2 com todas as colunas
557 if (num_linha==2)
558 {
559 switch (num_coluna)
560 {
561 case 1:
562 tecla_digito= '4';
563 break;
564 case 2:
565 tecla_digito= '5';
566 break;
567 case 3:
568 tecla_digito= '6';
569 break;
570 case 4:
571 tecla_digito= 'B'; //-- 18 Equivale a letra B;
572 break;
573 }
574 }
575 //-- Testa Linha 3 com todas as colunas
576 if (num_linha==3)
577 {
578 switch (num_coluna)
579 {
580 case 1:
581 tecla_digito= '7';
582 break;
583 case 2:
584 tecla_digito= '8';
585 break;
586 case 3:
587 tecla_digito= '9';
588 break;
589 case 4:
590 tecla_digito= 'C'; //-- 19 Equivale a letra C;
591 break;
592 }
593 }
594 //-- Testa Linha 4 com todas as colunas
595 if (num_linha==4)
596 {
597 switch (num_coluna)
598 {
599 case 1:
600 tecla_digito= '0';
601 break;
602 case 2:
8
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
603 tecla_digito= 'F'; //-- 22 Equivale a letra F(ponto `.´);
604 break;
605 case 3:
606 tecla_digito= 'E'; //-- 21 Equivale a letra E;
607 break;
608 case 4:
609 tecla_digito= 'D'; //-- 20 Equivale a letra D;
610 break;
611 }
612 }
613 //comando_lcd(0xC0); // posiciona o cursor na linha 1, coluna 0
614 //escreve_lcd(0x30 + tecla_digito); //
```

```
comando_lcd(0xC8); // posiciona o cursor na linha 1, coluna 15
escreve_lcd(tecla_digito); //
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Le Tecla Pressionada *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void le_teclado(void)
{
tecla_digito= 100; //-- valor padrao do teclado indica nao pressionado
if(filtro == 250)
{
coluna = coluna >> 1;
if(!coluna) coluna = 0b00001000;
PORTD = coluna;
}
if(coluna4) num_coluna = 4; // verifica qual é a linha ativa
if(coluna3) num_coluna = 3;
if(coluna2) num_coluna = 2;
if(coluna1) num_coluna = 1;

//-- conforme a coluna le a linha
switch(PORTD & 0b11110000)
{
case 16:
num_linha = 1; // verifica qual é a linha ativa
tecla_precionada();
break;

case 32:
num_linha = 2; // verifica qual é a linha ativa
tecla_precionada();
break;

case 64:
num_linha = 3; // verifica qual é a linha ativa
tecla_precionada();
break;

case 128:
num_linha = 4; // verifica qual é a linha ativa
tecla_precionada();
break;

default:
filtro = 250;
PORTB = 0;
PORTD = 0;
PORTB = coluna;
break;
}
}


/* ------------------------------------------------------------------- *
* ------------------------------------------------------------------- *
* PROCESSAMENTO DO CODIGO DE BARRAS e SENHAS *
* ------------------------------------------------------------------- *
* ------------------------------------------------------------------- */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Le Código de Barras *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c

```
679 // Função le_cartao leitor RS232 - le dado da USART e insere em cbarra[]
680 // Leitor de Codigo de Barras MinyScan
681
682 void le_cartao(void)
683 {
684 unsigned char j;
685
686 j= 0;
687 cont= 0;
688 comando_lcd(0xC8);
689 while((carac_temp != 0x03) && (cont<500))
690 {
691 while(!DataRdyUSART());
692 carac_temp = ReadUSART();
693
694 //-- caracter 0x03 indica fim de leitura
695 if(carac_temp != 0x03)
696 {
697 escreve_lcd(carac_temp); //máscara para caracter
698 cbarra[j]= carac_temp;
699 j++;
700 }
701 cont++;
702 }
703
704 //-- Se houve leitura muda fluxo do programa
705 if (j>0)
706 fluxo_programa= 1;
707 }
708
709 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
710 * Testa Cartao Master *
711 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
712 //Função compara valor do buffer com valor gravado como MASTER
713 //retorna - 1 leitura é cartão master, 0 não é cartão master
714
715 int cartaomaster(void)
716 {
717 int k,ret;
718
719 ret=1; //retorno como cartão MASTER
720 for (k=0;k<8;k++) //prepara comparação
721 if(cbarra[k] != cmaster[k]) //se valor diferente
722 {
723 ret=0; //não é master
724 }
725
726 return ret; //retorna valor
727 }
728
729 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
730 * Testa Senha Master *
731 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
732 //Função compara valor do buffer com valor gravado como SENHA MASTER
733 //retorna - 1 leitura é senha master, 0 não é senha master
734
735 int senhamaster(void)
736 {
737 int k,ret;
738
739 ret=1; //retorno como cartão MASTER
740 for (k=0;k<4;k++) //prepara comparação
741 if(csenha[k] != smaster[k]) //se valor diferente
742 {
```

```
ret=0; //não é master
}

return ret; //retorna valor
}



/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Testa Validade do Cartao *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
int valida_cartao(void)
```



```
{
int j, k, z, y, w, retv;
char contax, qtde_cartoes, contay;
long int endereco;

//-- inicio
retv= 0; // Cartao Invalido

//-- Le cartoes da Memoria
for(j=0;j<11;j++) // J = quantidade de cartoes
{
endereco= j * 12; // endereco de memoria

//-- le memoria codigo e senha
for (k=0;k<12;k++) // K = le Bytes da Memoria
bufferlido[k]= read_EEPROM(endereco+k);

//-------------------------------
//-- Testa Cartao lido com Memoria
contax= 0;
for(z=0;z<8;z++)
{
if(bufferlido[z] == cbarra[z]) //se valor diferente
contax++;
}


if (contax>=7)
{
//-------------------------------
//-- Teste Senha
tela_cartaovalido();
retv=1;
endmemoria= endereco;
//comando_lcd(0xC7); // posiciona o cursor na linha 1, coluna 15
//escreve_lcd(0x30+j);
}
}

return retv; //retorna valor
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Testa Validade da Senha *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
int valida_senha(void)
{
int z, j, k, retz;
unsigned char contax;
unsigned char contasenhax;
```

```
806 long int endereco;
807
808 //-- inicio
809 retz= 0; // Senha Invalido
810
811 endereco= endmemoria;
812
813 for (k=0;k<12;k++) // K = le Bytes da Memoria
814 bufferlido[k]= read_EEPROM(endereco+k);
815
816 //-- Le Teclado
817 contasenhax= 0;
818 while (contasenhax<4) // Le 4 teclas do teclado
819 {
820 le_teclado();
821 Delay10KTCYx(5); // delay 5 ms
822 comando_lcd(0xCC);
823 if (tecla_digito!=100)
824 {
825 comando_lcd(0xCC+contasenhax);
826 senhalida[contasenhax]= tecla_digito; // armazena senha digitada
827 escreve_lcd(tecla_digito);
828 contasenhax++;
829 tecla_digito= 100;
830 }
11
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
831 }
832
833 //---------------------------------
834 //-- Testa Senha MASTER com digitada
835 if (contasenhax>=3)
836 {
837 //---------------------------------
838 //-- Testa Cartao lido com Memoria
839 contax= 0;
840 for(z=8;z<12;z++)
841 {
842 if(bufferlido[z] == senhalida[z-8]) //se valor diferente
843 contax++;
844 }
845
846 if (contax>=3)
847 {
848 //-------------------------------
849 //-- Teste Senha
850 retz= 1;
851 }
852 }
853
854 return retz; //retorna valor
855 }
856
857
858 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
859 * Grava Novo Cartao e Senha *
860 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
861
862 void grava_cartaosenha(void)
863 {
864 int j,k,z,y;
865 unsigned char qtde_cartoes;
866 unsigned char buffer[25];
867 long int endereco;
868
```

```
869 fluxo_programa= 4;
870 qtde_cartoes = read_EEPROM(0xFF); //LÊ MEMÓRIA EEPROM qtde cartoes arquivados
871
872 //-- Verifica cartao atual
873 if ((qtde_cartoes>0) && (qtde_cartoes<11))
874 qtde_cartoes++;
875 else
876 qtde_cartoes=1;
877
878 //-- grava na memoria a nova quantidade de cartoes
879 write_EEPROM(0xFF, qtde_cartoes);
880 qtde_cartoes = read_EEPROM(0xFF);
881 comando_lcd(0xCF);
882 escreve_lcd(qtde_cartoes);
883
884 //-- Testa se quantidade menor que 10 e grava
885 if (qtde_cartoes < 11)
886 {
887 //-- gera buffer para gravacao
888 for(j=0;j<12;j++)
889 {
890 if (j<8)
891 buffer[j]= cbarra[j];
892 else
893 buffer[j]= csenha[j-8];
894 }
895 }
896
897 //-- grava na memoria EEPROM
898 if (qtde_cartoes==1)
899 endereco= 0;
900 else
901 endereco= (qtde_cartoes-1)* 12; //valor do novo endereço
902 // grava cartao cod. barra
903 for (z=0;z<12;z++)
904 write_EEPROM(endereco+z, buffer[z]); //escreve o cod. barra na EEPROM
905
906
12
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
907 //-- Mostra valores gravados
908 limpa_lcd();
909 comando_lcd(0x80);
910 for (k=0;k<12;k++)
911 escreve_lcd(buffer[k]);
912
913 //-- Mostra dados lidos
914 //-- le memoria para mostrar gravacao
915 for (k=0;k<12;k++)
916 cbarranew[k]= read_EEPROM(endereco+k);
917
918 comando_lcd(0xC0);
919 for (k=0;k<12;k++)
920 escreve_lcd(cbarranew[k]);
921
922 }
923
924
925 /* ------------------------------------------------------------------------ *
926 * ------------------------------------------------------------------------ *
927 * ROTINA PRINCIPAL DO PROGRAMA *
928 * ------------------------------------------------------------------------ *
929 * ------------------------------------------------------------------------ */
930
931 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
932 * Configurações do Pic *
933 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
934
935 void main(void)
936 {
937 char t;
938 PORTA = 0x00; //Limpa PORTA
939 PORTB = 0x00; //Limpa PORTB
940 PORTC = 0x00; //Limpa PORTC
941 PORTD = 0x00; //Limpa PORTD
942 PORTE = 0x00; //Limpa PORTE
943
944 LATA = 0x00; //Limpa PORTA
945 LATB = 0x00; //Limpa PORTB
946 LATC = 0x00; //Limpa PORTC
947 LATD = 0x00; //Limpa PORTD
948 LATE = 0x00; //Limpa PORTE
949
950 TRISA = 0b11000011; //CONFIG DIREÇÃO DOS PINOS PORTA
951 TRISB = 0b00000000; //CONFIG DIREÇÃO DOS PINOS PORTB
952 TRISC = 0b10111101; //CONFIG DIREÇÃO DOS PINOS PORTC
953 TRISD = 0b00000000; //CONFIG DIREÇÃO DOS PINOS PORTD
954 TRISE = 0b00000100; //CONFIG DIREÇÃO DOS PINOS PORTE
955
956 ADCON1 = 0b00001111; //DESLIGA CONVERSORES A/D
957
958 //-- Configura Comunicação Serial RS232
959 //-- Baud Rate= 9600 : 8 Bits : Sem paridade : Stop Bit = 1;
960 OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &
961 USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_HIGH,25);
962
963 //-- Inicialização do Programa
964 inicializa_lcd(); // configura o lcd
965 tela_principal();
966 Delay10KTCYx(100);
967 limpa_lcd();
968 tela_passecartao();
969
970 contasenha= 0; // contador de passagem
971 tecla_digito= 100; // controle de tecla pressionada
972 fluxo_programa= 0; // variavel que controla o fluxo do prog.
973 fluxo_cmaster= 0; // variavel de controle cartao master
974 coluna = 0b00001000; // variavel inicial para coluna teclado
975 comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
976 escreve_lcd(0x30 + fluxo_programa);
977
978 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
979 * Rotina principal *
980 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
981
982 //-- le da memoria a nova quantidade de cartoes
13
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
983 t= read_EEPROM(0xFF);
984 comando_lcd(0xCF); // posiciona o cursor na linha 1, coluna 15
985 escreve_lcd(0x30+t);
986
987 //contabarra= 0; // contador de digitos lidos do Cod. Barra
988 //fluxo_programa= 0; // variavel que controla o fluxo do prog.
989 //comando_lcd(0x8F); // posiciona o cursor na linha 1, coluna 15
990 //escreve_lcd(0x30 + fluxo_programa); //escreve no LCD o atual fluxo do prog.
991 //limpa_lcd();
992 //tela_passecartao();
993
994
```

```
995 while(1) // rotina principal
996 {
997 //ClrWdt(); //Inicia o watch-dog timer
998
999 //---------------------------------------------------------------------
1000 //-- Passo 0: Inicio - Pede para passar o cartao e Le codigo de barras
1001 //---------------------------------------------------------------------
1002 t= read_EEPROM(0xFF);
1003 comando_lcd(0xCF); // posiciona o cursor na linha 1, coluna 15
1004 escreve_lcd(0x30+t);
1005
1006 while(!DataRdyUSART()); // Verifica se foi lido algo na USART
1007 carac_temp = ReadUSART(); // Armazena caracter lido
1008
1009 //-- Se caracter lido for de inicio de leitura Le cartao
1010 if(carac_temp == 0x02) // caracter 0x02 indica inicio leitura do CB
1011 le_cartao();
1012
1013
1014 //---------------------------------------------------------------------
1015 //-- Passo 1: Testa Codigo de Barras lido= CARTAO MASTER e SENHA MASTER
1016 //---------------------------------------------------------------------
1017 //-- Se leu cartao fluxo do programa passa a ser 1
1018 if ((fluxo_programa==1) && (fluxo_cmaster==0))
1019 {
1020 //-----------------------------
1021 //-- Testa se é o Cartao MASTER
1022 if (cartaomaster()) // se for Master retorno == 1
1023 {
1024 fluxo_programa= 2; // cartao MASTER OK fluxoprograma =2
1025 fluxo_cmaster= 0;
1026 tela_cartaomaster(); // mostra tela mensagem de Cartao Master OK
1027 //-- Le Teclado
1028 contasenha= 0;
1029 while (contasenha<4) // Le 4 teclas do teclado
1030 {
1031 le_teclado();
1032 Delay10KTCYx(5); // delay 5 ms
1033 comando_lcd(0xCC);
1034 if (tecla_digito!=100)
1035 {
1036 comando_lcd(0xCC+contasenha);
1037 csenha[contasenha]= tecla_digito; // armazena senha digitada
1038 escreve_lcd(tecla_digito);
1039 contasenha++;
1040 tecla_digito= 100;
1041 }
1042 }
1043
1044 //---------------------------------
1045 //-- Testa Senha MASTER com digitada
1046 if (contasenha>=3)
1047 {
1048 fluxo_programa= 3;
1049 //-- Testa se a Senha é Master
1050 if (senhamaster()) // se for Senhar Master retorno == 1
1051 {
1052 tela_senhaconfere(); // tela mostra senha OK
1053 Delay10KTCYx(400); // delay 400ms
1054 fluxo_programa= 0;
1055 fluxo_cmaster= 1;
1056 tela_passecartaonovo(); // tela mostra passe cartao Novo
1057 //-- proximo passo ler cartao novo Fluxo_programa= 4
```

```
1058 }
1059 else
1060 {
1061 //---------------------------
1062 //-- Senha não confere
1063 tela_senhanaoconfere(); // tela mostra senha Incorreta
1064 Delay10KTCYx(400);
1065 fluxo_programa= 0;
1066 fluxo_cmaster=0;
1067 tecla_digito= 100;
1068 contasenha= 0;
1069 tela_passecartao(); // volta a tela inicial
1070 }
1071 }
1072
1073 }
1074 //---------------------------
1075 //-- Não é Cartao Master
1076 else
1077 {
1078 //-----------------------------
1079 //-- Testa cartão MEMORIA EEPROM
1080 if (valida_cartao())
1081 {
1082 //-----------------------------
1083 //-- Testa valida senha
1084 if (valida_senha())
1085 {
1086 //acesso autorizado
1087 tela_acessoautor();
1088 RELE1 = 1;
1089 Delay10KTCYx(400);
1090 Delay10KTCYx(400);
1091 RELE1 = 0;
1092 //-- volta ao inicio
1093 fluxo_programa= 0;
1094 fluxo_cmaster= 0;
1095 tecla_digito= 100;
1096 contasenha= 0;
1097 tela_passecartao();
1098 }
1099 else
1100 {
1101 tela_acessonaoautor();
1102 Delay10KTCYx(400);
1103 Delay10KTCYx(400);
1104 //-- Cartao Invalido para MASTER
1105 fluxo_programa= 0;
1106 fluxo_cmaster= 0;
1107 tecla_digito= 100;
1108 contasenha= 0;
1109 tela_passecartao();
1110 }
1111 }
1112 //-- Cartao Invalido
1113 else
1114 {
1115 //-- Cartao Invalido para MASTER
1116 tela_cartaoinvalido();
1117 Delay10KTCYx(400);
1118 fluxo_programa= 0;
1119 fluxo_cmaster= 0;
1120 tecla_digito= 100;
1121 contasenha= 0;
1122 tela_passecartao();
```

```
1123 }
1124 }
1125 }
1126
1127 //---------------------------------------------------------------------
1128 //-- Passo 2: le cartao e senha NOVOS e Grava na EEPROM
1129 //---------------------------------------------------------------------
1130
1131 if ((fluxo_programa==1) && (fluxo_cmaster==1))
1132 {
1133 //-- cartao novo armazenado em cbarra[x];
15
C:\Projetos PIC18\TCCs\Acesso2\Acesso Prog McLab3 030413.c
1134 Delay10KTCYx(400);
1135 fluxo_programa= 4; // cartao novo passado
1136 tela_cartaonovook(); // tela mostra passe cartao Novo
1137 //-- Le Teclado Nova Senha
1138 contasenha= 0;
1139 while (contasenha<4) // Le 4 teclas do teclado
1140 {
1141 le_teclado();
1142 Delay10KTCYx(5); // delay 5 ms
1143 comando_lcd(0xCC);
1144 if (tecla_digito!=100)
1145 {
1146 comando_lcd(0xCC+contasenha);
1147 csenha[contasenha]= tecla_digito; // armazena senha digitada
1148 escreve_lcd(tecla_digito);
1149 contasenha++;
1150 tecla_digito= 100;
1151 }
1152 }
1153 //-- Senha Armazenada em csenha[x]
1154 if (contasenha>=3)
1155 {
1156 //-- grava senha e cartao
1157 tela_gravacaoook();
1158 Delay10KTCYx(400);
1159 grava_cartaosenha();
1160 Delay10KTCYx(400);
1161 fluxo_programa= 0;
1162 fluxo_cmaster=0;
1163 tecla_digito= 100;
1164 contasenha= 0;
1165 tela_passecartao();
1166 }
1167 }
1168 } //-- FIM DO While programa principal
1169 } // FIM DO PROGRAMA
1170
```