8. TECLADO MATRICIAL

Neste capítulo, apresenta-se a técnica de varredura para um teclado matricial. Essa técnica é comum em sistemas microcontrolados, sendo empregada para maximizar o uso dos pinos de I/O do microcontrolador.

Uma forma muito comum de entrada de dados em um sistema microcontrolado é através de teclas (botões ou chaves tácteis). Quando o número delas é pequeno, cada uma pode ser associada a um pino de I/O do microcontrolador. Entretanto, quando o seu número é grande, não é conveniente utilizar muitos pinos de I/O. Um teclado convencional emprega 3×4 teclas (12) ou 4×4 teclas (16) (fig. 8.1). Ao invés de se empregar 12 ou 16 pinos para a leitura desses teclados, empregam-se 7 ou 8 pinos, respectivamente.

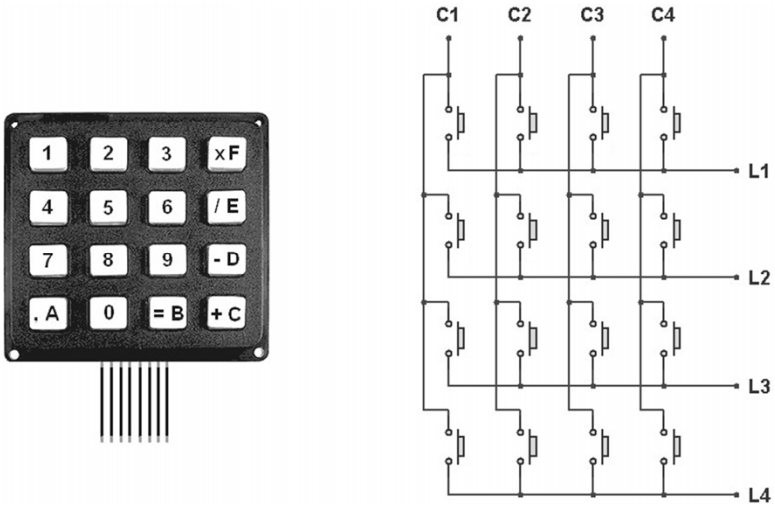

Fig. 8.1 – Teclado matricial hexadecimal: 4×4.

Para a leitura de teclados com várias teclas, é necessário empregar o conceito de matriz, em que os botões são arranjados em colunas e linhas, conforme é mostrado na fig. 8.1; o teclado é lido utilizando-se uma varredura. O pressionar de uma tecla produzirá um curto-circuito entre uma coluna e uma linha e o sinal da coluna se refletirá na linha ou vice-versa. A ideia é trocar sucessivamente o nível lógico das colunas, a chamada varredura, e verificar se esse nível lógico aparece nas linhas (ou vice-versa). Se, por exemplo, a varredura for feita nas colunas e houver alteração no sinal lógico de alguma linha, significa que alguma tecla foi pressionada e, então, com base na coluna habilitada, sabe-se qual tecla foi pressionada. Nesse tipo de varredura, o emprego de resistores de *pull-up* ou *pull-down* nas vias de entrada é fundamental, visto que para a leitura as entradas sempre devem estar em um nível lógico conhecido. Na fig. 8.2, dois teclados com resistores de p*ull-up* e *pull-down* são ilustrados, onde o sinal de varredura é aplicado às colunas, as saídas do sistema de controle.

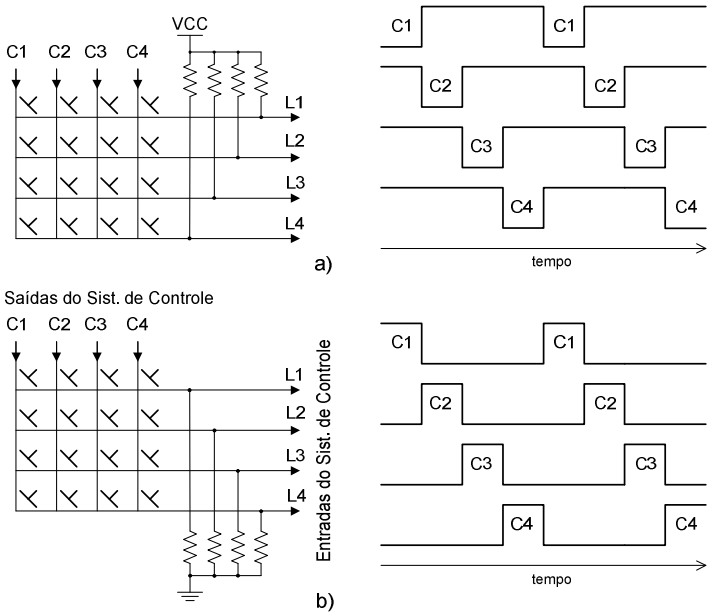

Fig. 8.2 – Teclados com resistores de *pull-up* (a) e *pull-down* (b) para as entradas do sistema de controle. A varredura é feita nas colunas.

No ATmega, a conexão de um teclado é facilmente obtida, pois existem resistores de *pull-up* habilitáveis em todos os pinos de I/O. Assim, um possível circuito para o trabalho com um teclado é apresentado na fig. 8.3. Um código exemplo com uma função para leitura desse teclado é apresentado na sequência.

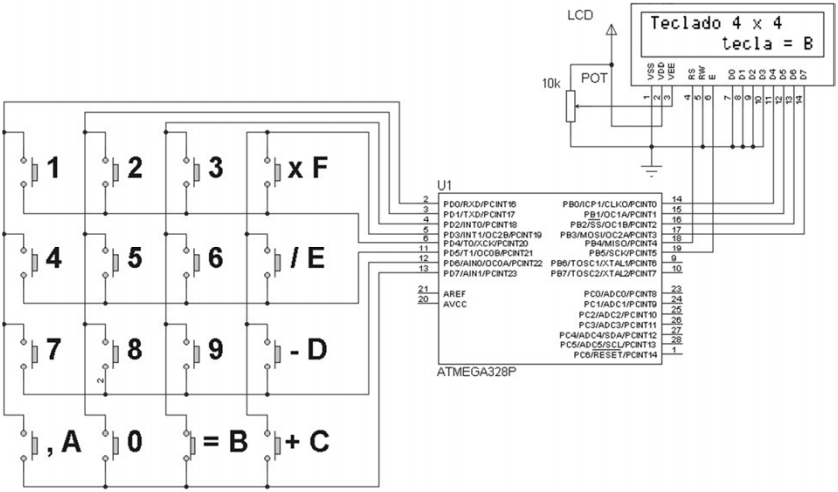

Fig. 8.3 – Teclado 4 × 4 controlado pelo ATmega328.

Teclado_hexa.c (programa principal)

```
//=====================================================================    //
//              LEITURA DE UM TECLADO 4 x 4                                //
//=====================================================================    //
#include "def_principais.h"    //inclusão do arquivo com as principais definições
#include "LCD.h"
#include "teclado.h"

//definição para acessar a memória flash como ponteiro
prog_char mensagem1[] = "Teclado 4 x 4\0";//mensagem armazenada na memória flash
prog_char mensagem2[] = "tecla =\0";     //mensagem armazenada na memória flash

int main()
{
     unsigned char nr;

     DDRB = 0xFF;         //LCD esta no PORTB
     DDRD = 0x0F;         //definições das entradas e saídas para o teclado
     PORTD= 0xFF;         //habilita os pull-ups do PORTD e coloca colunas em 1
     UCSR0B = 0x00;       //para uso dos PORTD no Arduino

     inic_LCD_4bits();
     escreve_LCD_Flash(mensagem1);
     cmd_LCD(0xC7,0);     //desloca cursor para a 2a linha do LCD
     escreve_LCD_Flash(mensagem2);
```

```
     while(1)
     {
          nr = ler_teclado();  //lê constantemente o teclado

          if(nr!=0xFF)//se alguma tecla foi pressionada mostra seu valor
          {
               cmd_LCD(0xCF,0);//desloca cursor para a última posição da 2a linha
               cmd_LCD(nr,1);  //nr já está em formato ASCII
          }
     }
}
//=============================================================================
```

teclado.h (arquivo de cabeçalho do teclado.c)

```
#ifndef _TECLADO_H
#define _TECLADO_H

#include "def_principais.h"

#define LINHA        PIND      //registrador para a leitura das linhas
#define COLUNA       PORTD     //registrador para a escrita nas colunas

//protótipo da função
unsigned char ler_teclado();

#endif
```

teclado.c (arquivo com a função de trabalho para o teclado)

```
/*=============================================================================
     Sub-rotina para o trabalho com um teclado com 16 teclas (4 colunas e 4 linhas)
     organizados como:
                C1 C2 C3 C4
                 x  x  x  x    L1
                 x  x  x  x    L2
                 x  x  x  x    L3
                 x  x  x  x    L4
     onde se deve empregar um único PORT conectado da seguinte maneira:
     PORT =  L4 L3 L2 L1 C4 C3 C2 C1 (sendo o LSB o C1 e o MSB o L4)
=============================================================================  */
#include "teclado.h"

/*matriz com as informações para decodificação do teclado,
organizada de acordo com a configuração do teclado, o usuário
pode definir valores números ou caracteres ASCII, como neste exemplo*/

const unsigned char teclado[4][4] PROGMEM = {{'1', '2', '3', 'F'},
                                             {'4', '5', '6', 'E'},
                                             {'7', '8', '9', 'D'},
                                             {'A', '0', 'B', 'C'}};
//-----------------------------------------------------------------------------
```

```
unsigned char ler_teclado()
{
     unsigned char n, j, tecla=0xFF, linha;

     for(n=0;n<4;n++)
     {
          clr_bit(COLUNA,n);   //apaga o bit da coluna (varredura)
          _delay_ms(10);       /*atraso para uma varredura mais lenta, também elimina
                                                                        o ruído da tecla*/
          linha = LINHA >> 4;  //lê o valor das linhas

          for(j=0;j<4;j++)     //testa as linhas
          {
               if(!tst_bit(linha,j))//se foi pressionada alguma tecla,
               {                    //decodifica e retorna o valor
                    tecla = pgm_read_byte(&teclado[j][n]);
                    //while(!tst_bit(LINHA>>4,j));/*para esperar soltar a tecla, caso
                                                  desejado, descomentar essa linha*/
               }
          }
          set_bit(COLUNA,n);   //ativa o bit zerado anteriormente
     }
     return tecla;  //retorna o valor 0xFF se nenhuma tecla foi pressionada
}
//---------------------------------------------------------------------------------
```

Os arquivos **def_principais.h**, **LCD.h** e **LCD.c** foram apresentados no capítulo 5 (LCD 16×2 com via de dados de 4 bits). Houve alteração do arquivo **LCD.h** para se adequar a nova configuração das conexões do LCD, resultando em:

```
#define DADOS_LCD         PORTB
#define nibble_dados      0
#define CONTR_LCD         PORTB
#define E                 PB5
#define RS                PB4
```

O programa apresentado lê constantemente o teclado e apresenta o valor da tecla pressionada no LCD, a tecla B no caso exemplo da fig. 8.3. A função **ler_teclado( )** emprega dois laços `for`, o mais externo para realizar a varredura das colunas, feita pela macro `clr_bit(COLUNA,n)`, cuja variável **n** determina qual coluna será colocada em zero. Na sequência, tem-se o tempo da varredura por coluna. O laço `for` interno verifica, para cada coluna selecionada, se alguma linha foi ativa; em caso afirmativo, o valor da tecla correspondente é atribuído para a variável tecla de acordo com a organização do teclado definido na variável **teclado[4][4]**.

Quando se faz a varredura do teclado, é importante utilizar uma frequência adequada. Se ela for muito alta, podem aparecer ruídos

espúrios; se for baixa, a tecla pode ser pressionada e solta sem que o sistema detecte o seu acionamento. Na função **ler_teclado( )** foi utilizado um tempo de 10 ms para a varredura. Dessa forma, pela estruturação da função, é consumido um tempo aproximado de 40 ms na sua execução. Considerando-se que o ruído produzido pelo botão tem duração em torno de 10 ms, o tempo de execução da função poderia ser reduzido quatro vezes.

Exercícios:

8.1 – Elaborar um programa para um controle de acesso por senha numérica. A senha pode conter 3 dígitos. Toda vez que uma tecla for pressionada, um pequeno alto-falante deve ser acionado. Quando a senha for correta, o relé deve ser ligado por um pequeno tempo (utilize um LED de sinalização). Preveja que a senha possa ser alterada e salva na EEPROM. O circuito abaixo exemplifica o hardware de controle.

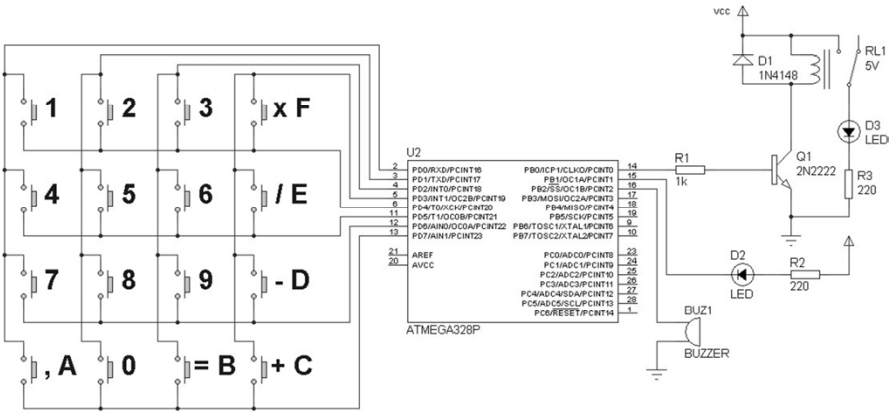

Fig. 8.4 – Controle de acesso por senha numérica.

8.2 – Elaborar um programa para ler um teclado alfanumérico, similar ao usado nos telefones celulares.

8.3 – Elaborar um programa para executar as funções matemáticas básicas de uma calculadora (números inteiros), conforme circuito da fig. 8.3. Consulte o apêndice B para a tabela de instruções do LCD, mude o sentido de deslocamento da mensagem para a esquerda na entrada de um novo caractere (0x07) e comece a escrita dos caracteres na última coluna da primeira linha.
