5197 - Sistema Digitais

Bacharelado de Informática

UEM – DIN - Prof. Elvio

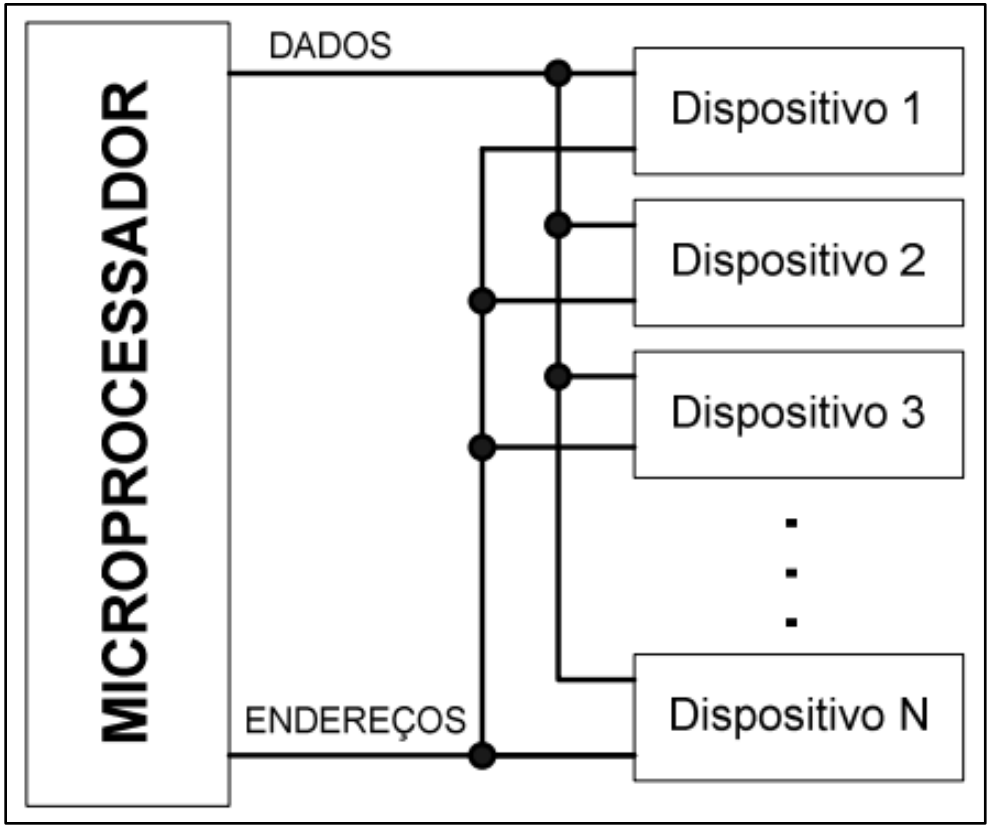
2016

Roteiro

- Multiplexação
  - Expansão de E/S
  - Conversão Série-Paralelo
  - Conversão Paralelo-Série
  - Compartilhando LEDs em pinos
  - Matriz de LEDs
  - Teclado Matricial

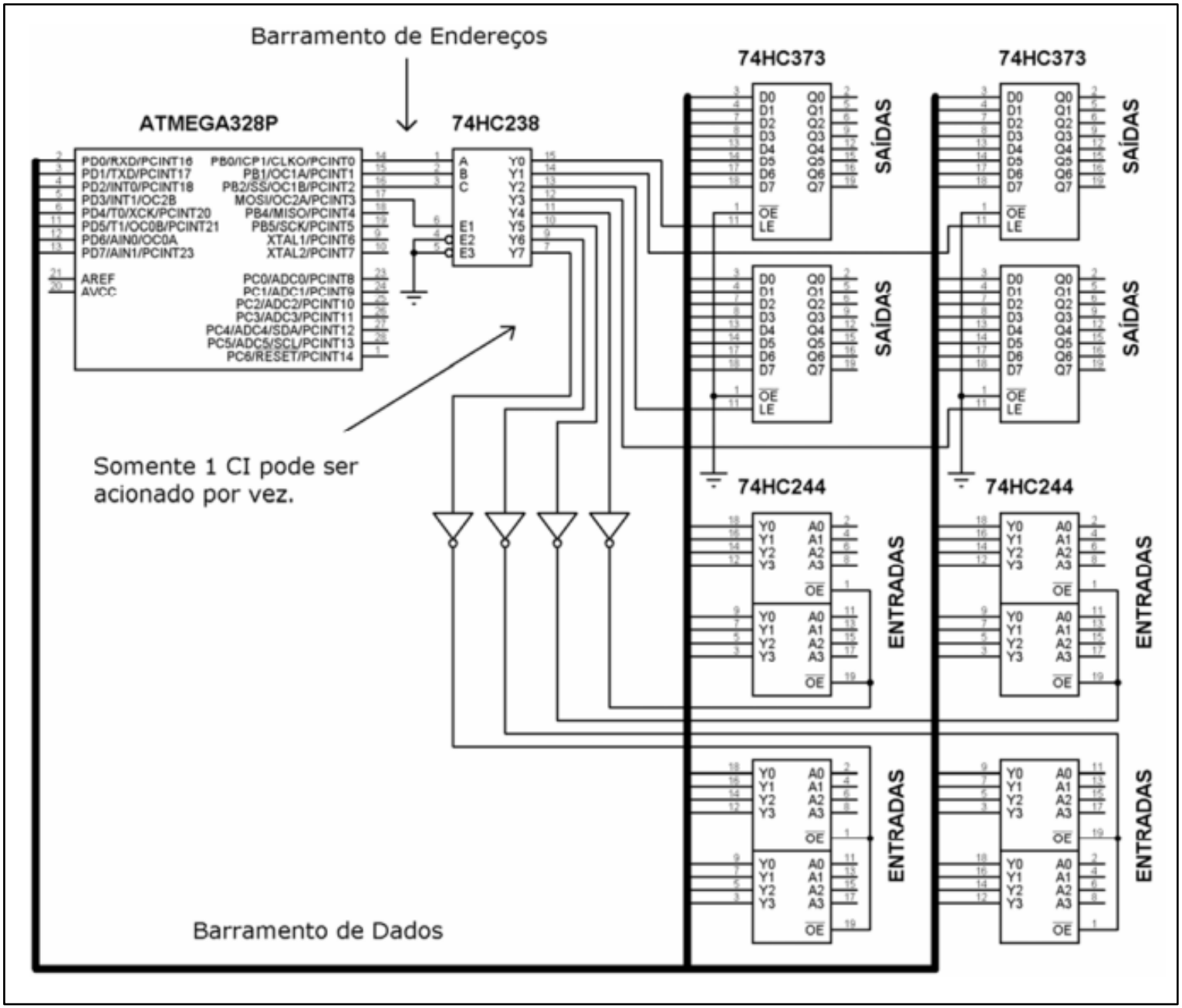
Expansão de E/S

- Permite acrescentar linhas de E/S através do uso de circuitos (hardware) adicional



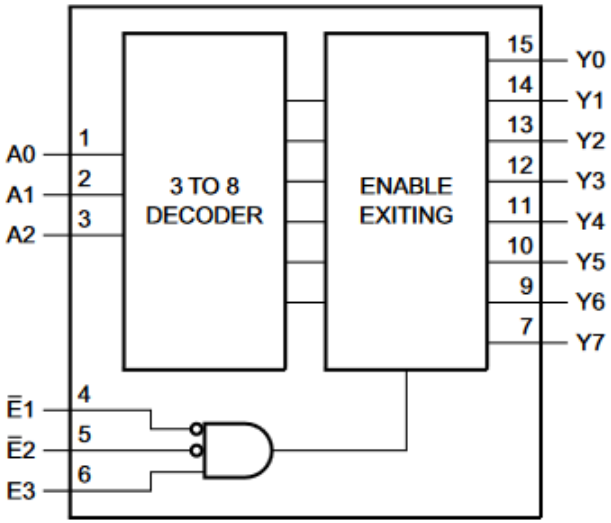
Expansão de E/S

- Exemplo de circuito

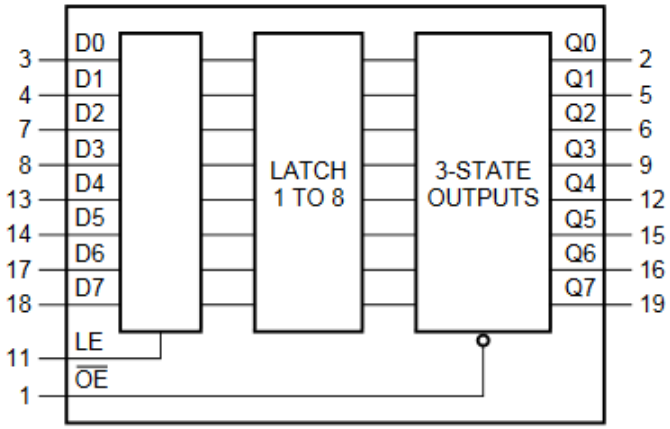


Expansão de E/S

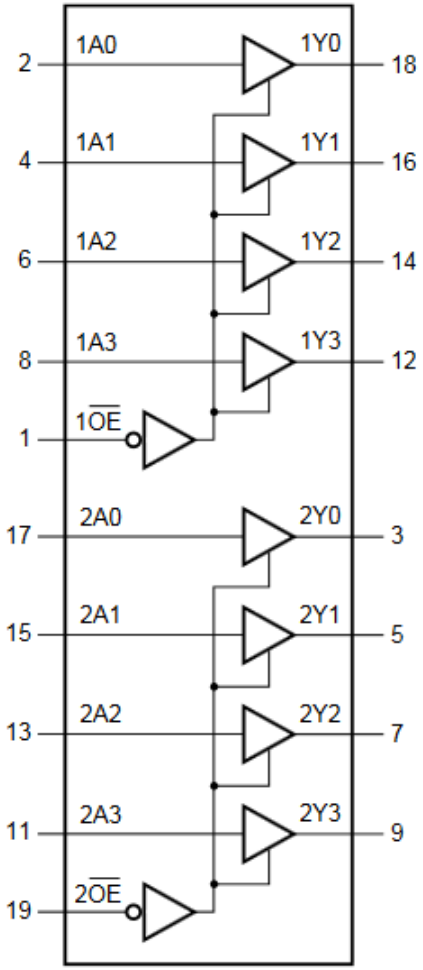
- HC238



- HC373

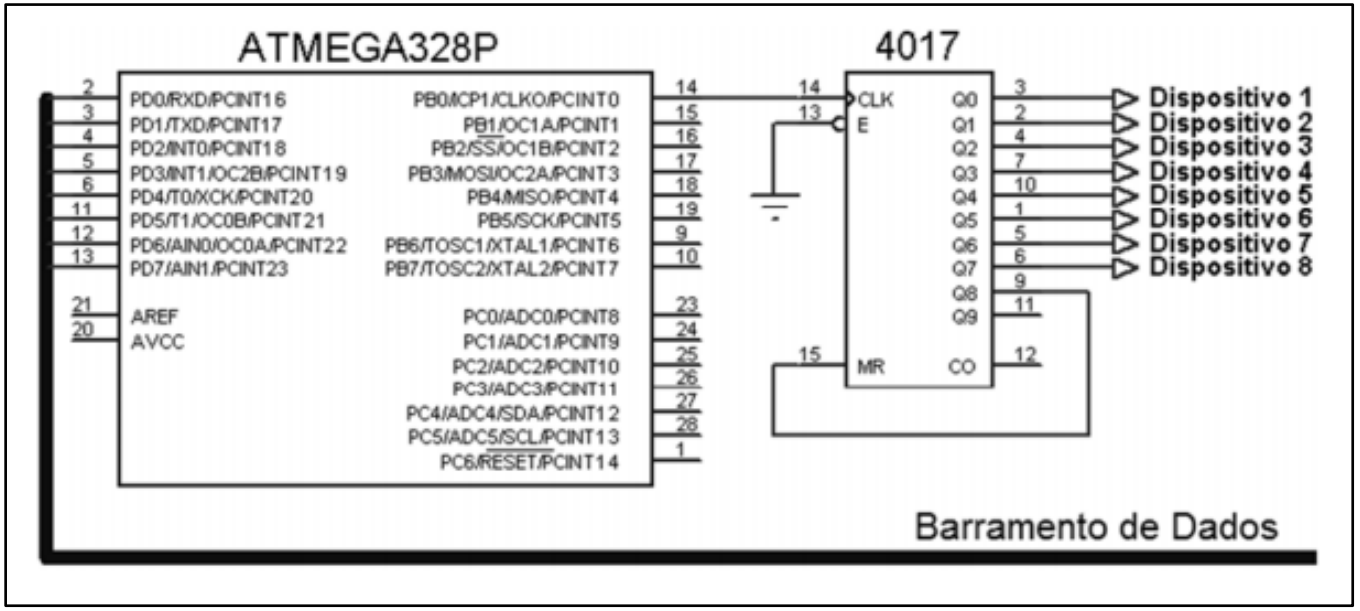


- HC244



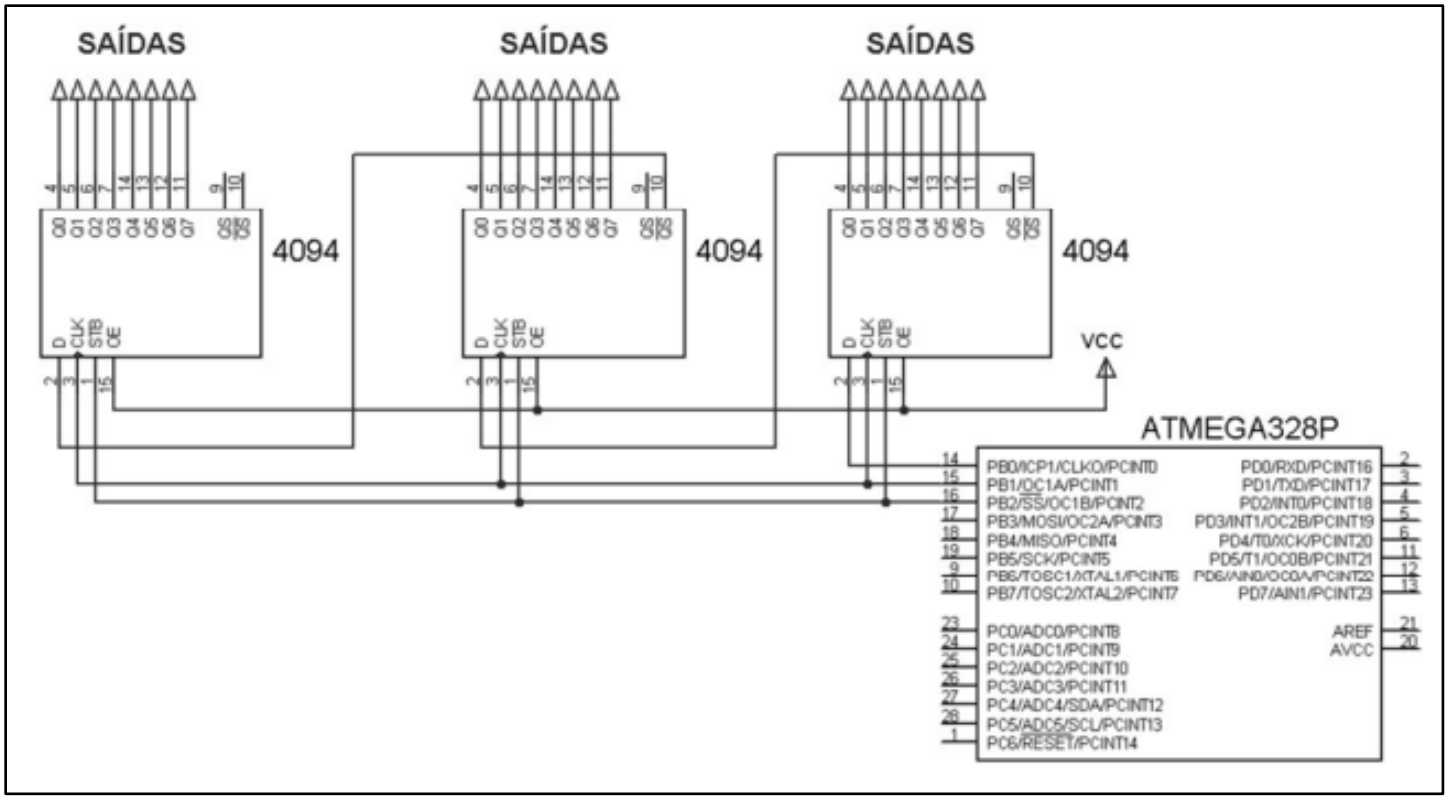
Expansão de E/S

- Usando um contador



Conversão Série-Paralelo

- Exemplo de circuito



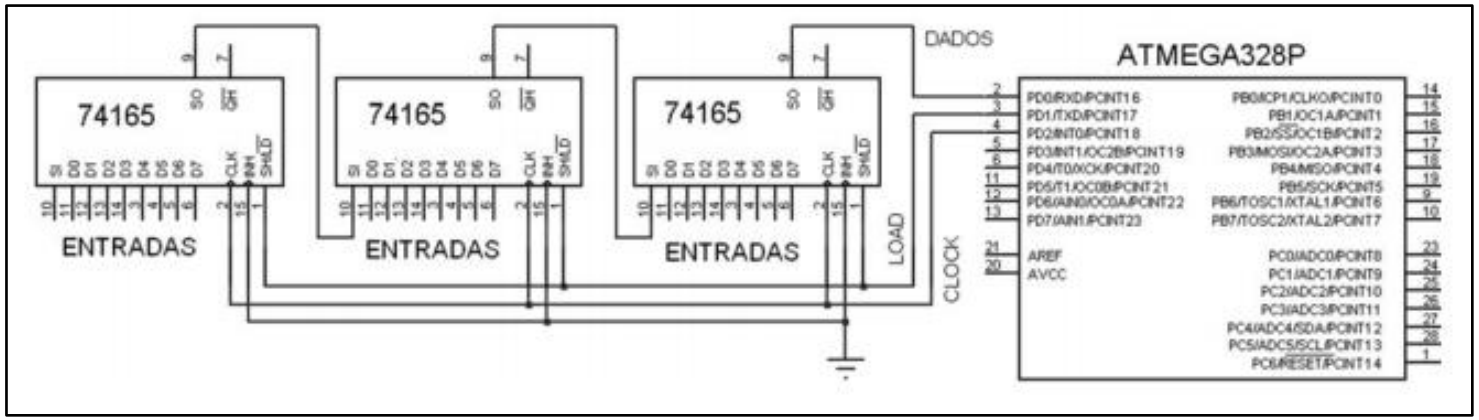
Conversão Série-Paralelo

Serial_paralelo.c

```
//=============================================================================//
//             Enviando 3 bytes para o 4094                                    //
//=============================================================================//
#include "def_principais.h"

#define D       PB0                 //pino de dados para o 4094
#define CLK     PB1                 //pino clock para o 4094
#define STB     PB2                 //pino de strobe para o 4094

#define pulso_CLK()  set_bit(PORTB,CLK);  _delay_us(10); clr_bit(PORTB,CLK)
#define pulso_STB()  set_bit(PORTB,STB);  _delay_us(10); clr_bit(PORTB,STB)
//-----------------------------------------------------------------------------
// Sub-rotina que envia 1 byte para o 4094 - serial/paralelo
//-----------------------------------------------------------------------------
void serial_paral(unsigned char c)
{
   unsigned char i=8;               //envia primeiro o MSB

   do
   { i--;

          if(tst_bit(c,i))          //se o bit for 1, ativa o pino de DADOS
               set_bit(PORTB,D);
          else                      //se não, o zera
               clr_bit(PORTB,D);

          pulso_CLK();

   } while (i!=0);
 }
//-----------------------------------------------------------------------------
int main(void)
{
     unsigned char j;
     unsigned char Dados[3]= {0x58, 0xF1, 0xAA};

     DDRB  = 0b00000111;  //pinos PB0:2 como saídas
     PORTB = 0b11111000;  //zera saídas

     for(j=0; j<3;j++)
               serial_paral(Dados[j]);//envia os 3 dados para os 4094 (primeiro o 0x58)

     pulso_STB();/*depois de enviar os 3 dados dá o pulso de Strobe, neste instante os
                                                          dados passam para as saída*/
     while(1)
     {}                  //laço infinito
}
//=============================================================================
```

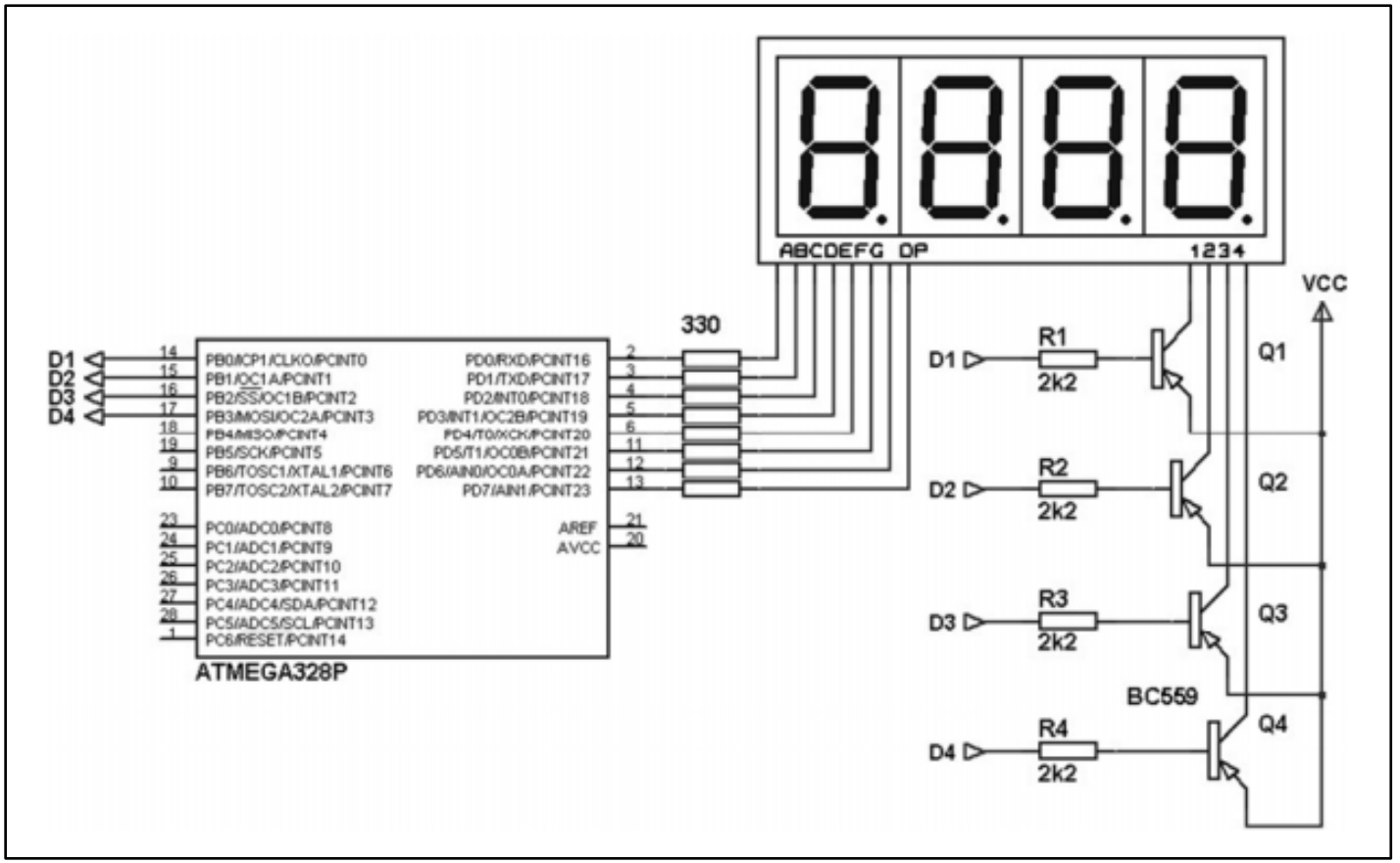
Conversão Paralelo-Série

- Exemplo de circuito



Multiplexação de Display de 7 Segmentos

- Exemplo de circuito



Multiplexação de Display de 7 Segmentos

Varredura_display_7seg.c

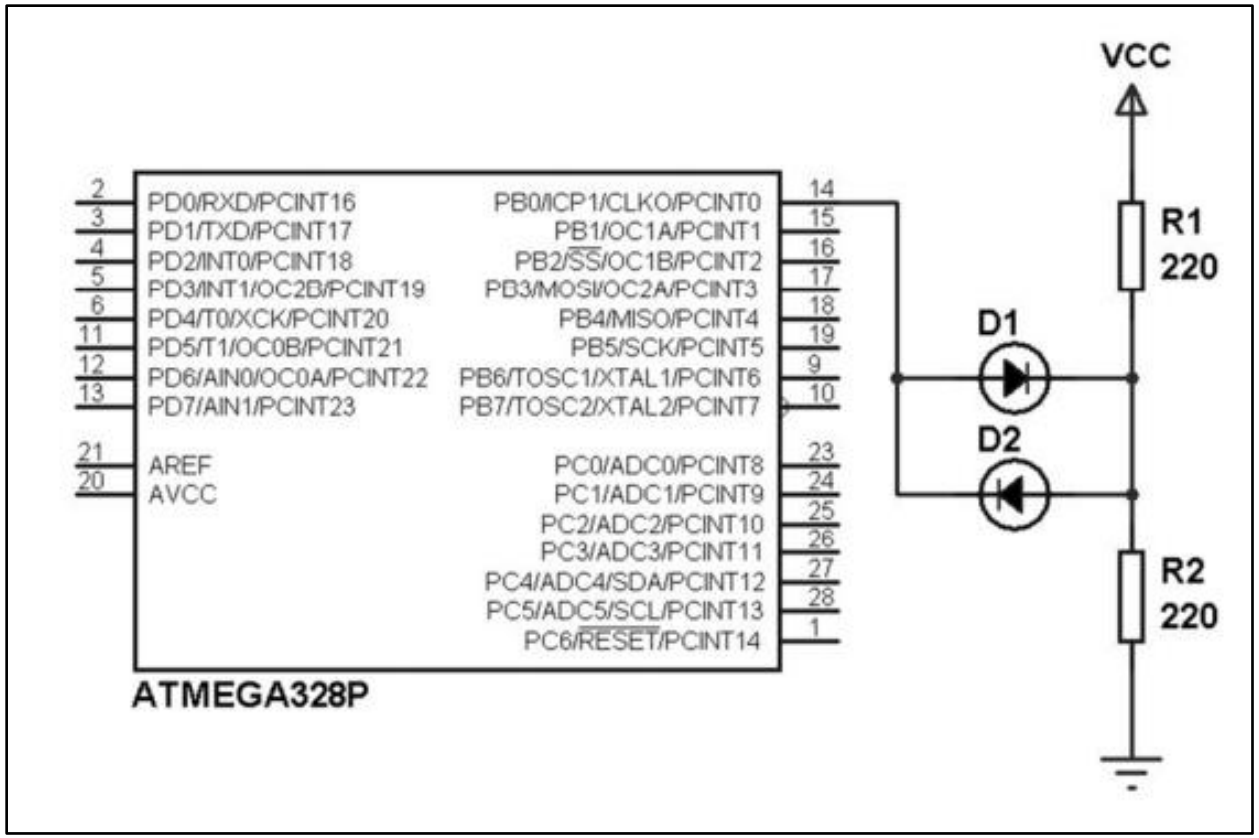
```
//=============================================================================== //
//        VARREDURA DE DISPLAYS DE 7 SEGMENTOS                                    //
//=============================================================================== //
#define F_CPU 16000000UL

#include <avr/io.h>
#include <avr/interrupt.h>
#define   clr_bit(Y,bit_x)     (Y&=~(1<<bit_x))

unsigned  char DISP[4];        //valores para os displays
//-------------------------------------------------------------------------------
//INTERRUPCAO - VARREDURA DOS DISPLAYS DE 7 SEGMENTOS
//-------------------------------------------------------------------------------
ISR(TIMER0_OVF_vect)
{
     static unsigned char x;

     PORTB |= 0x0F;//apaga todos os displays (o controle dos displays está nos pinos (PB0:PB3)
     PORTD = DISP[x];     //coloca a informação do display no porta correspondente
     clr_bit(PORTB,x);    //habilita o display correspondente (PB0:PB3)
     x++;

     if(x==4)   x = 0;     //após 4 rotações inicializa para o primeiro display
}
//-------------------------------------------------------------------------------
```

Multiplexação de Display de 7 Segmentos

```
int main()
{
     DDRD = 0xFF;         //dados dos displays
     DDRB = 0x0F;         //controle dos displays
     PORTB = 0xFF;        //apaga displays e liga pull-ups

     UCSR0B = 0x00;       //para usar os pinos do PORTD no Arduino

     //TC0 gerando interrupção
     TIMSK0 = 1<<TOIE0;   //habilita a interrupção por estouro do timer 0
     TCCR0B = 1<<CS02;    //CLK/256 prescaler (CLK=16MHz), estouro a cada 4ms
     sei();     //habilita a interrupção global

     while(1)  //qualquer escrita em DISP[x] é automaticamente apresentada nos displays
     {
          DISP[0]= 0xC0;  //valor para o número zero
          DISP[1]= 0xF9;  //valor para o número um
          DISP[2]= 0xA4;  //valor para o número dois
          DISP[3]= 0xB0;  //valor para o número três
     }
}
//============================================================================
```

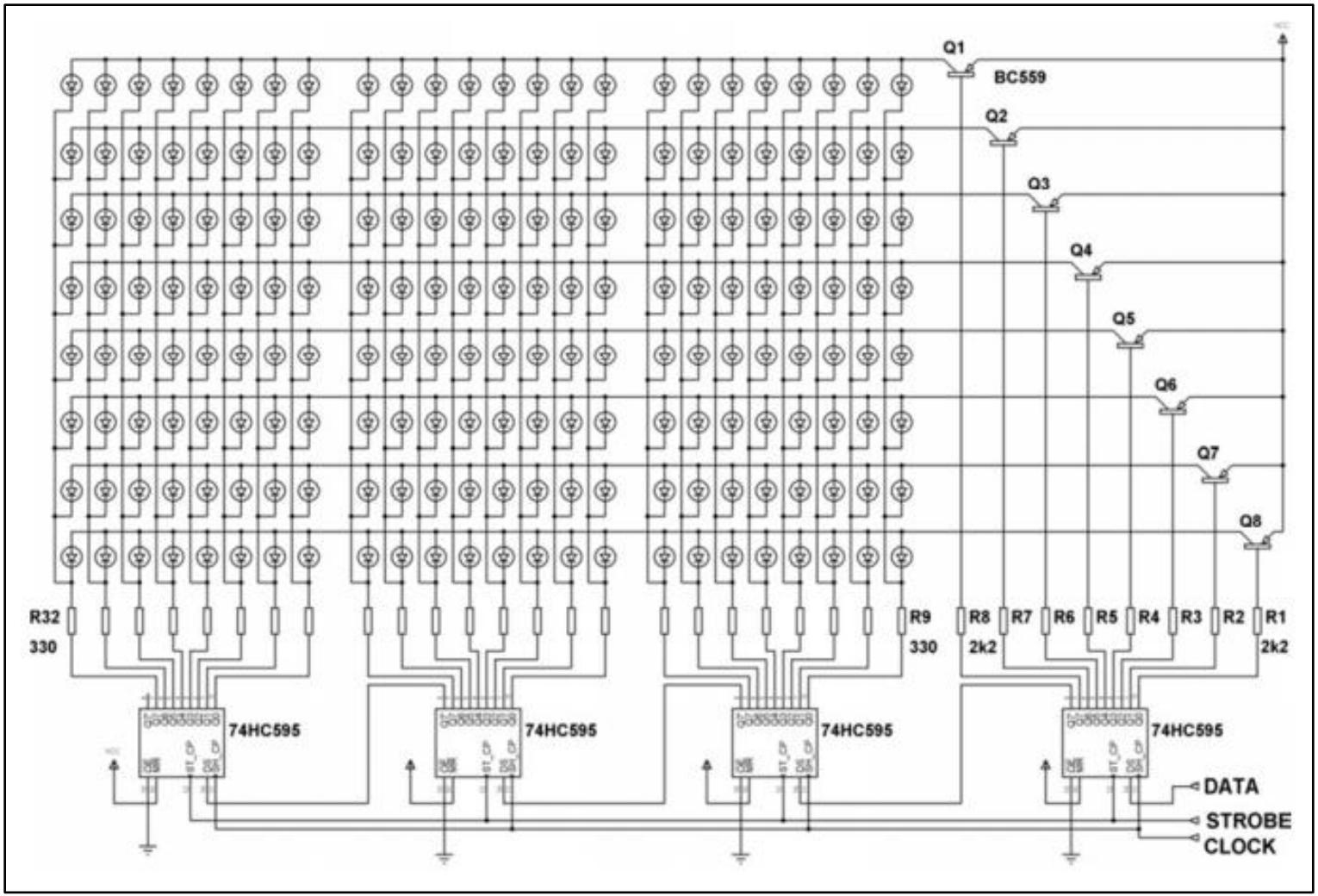
Compartilhando LEDs em Pinos

- Exemplo de circuito



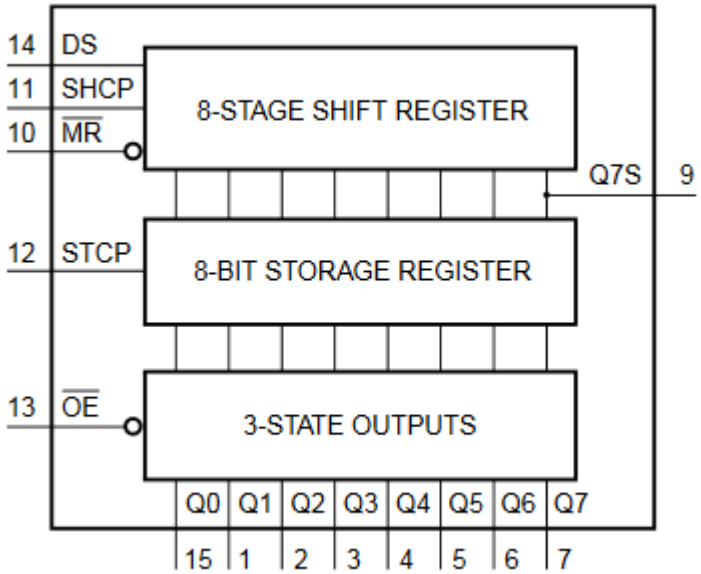
Matrix de LEDs

- Exemplo de circuito



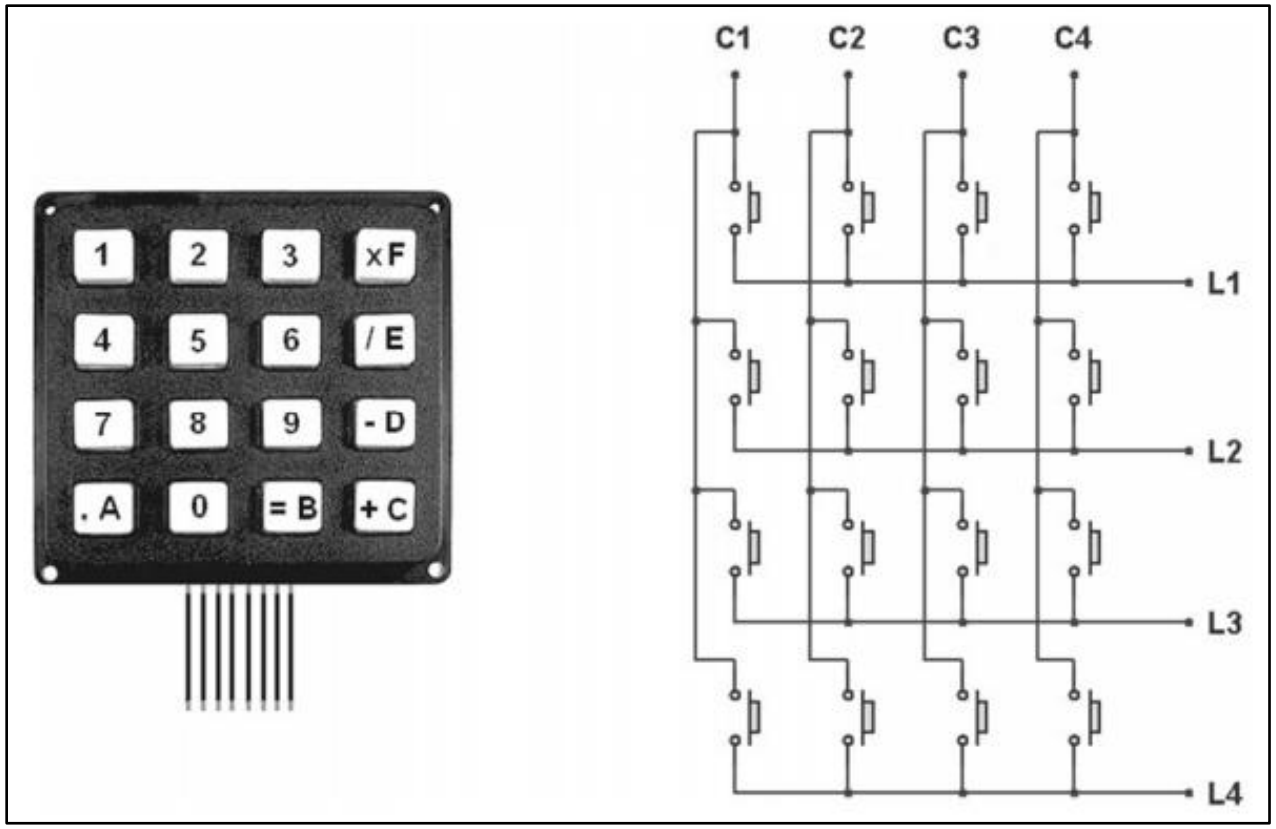
Matrix de LEDs

- HC595



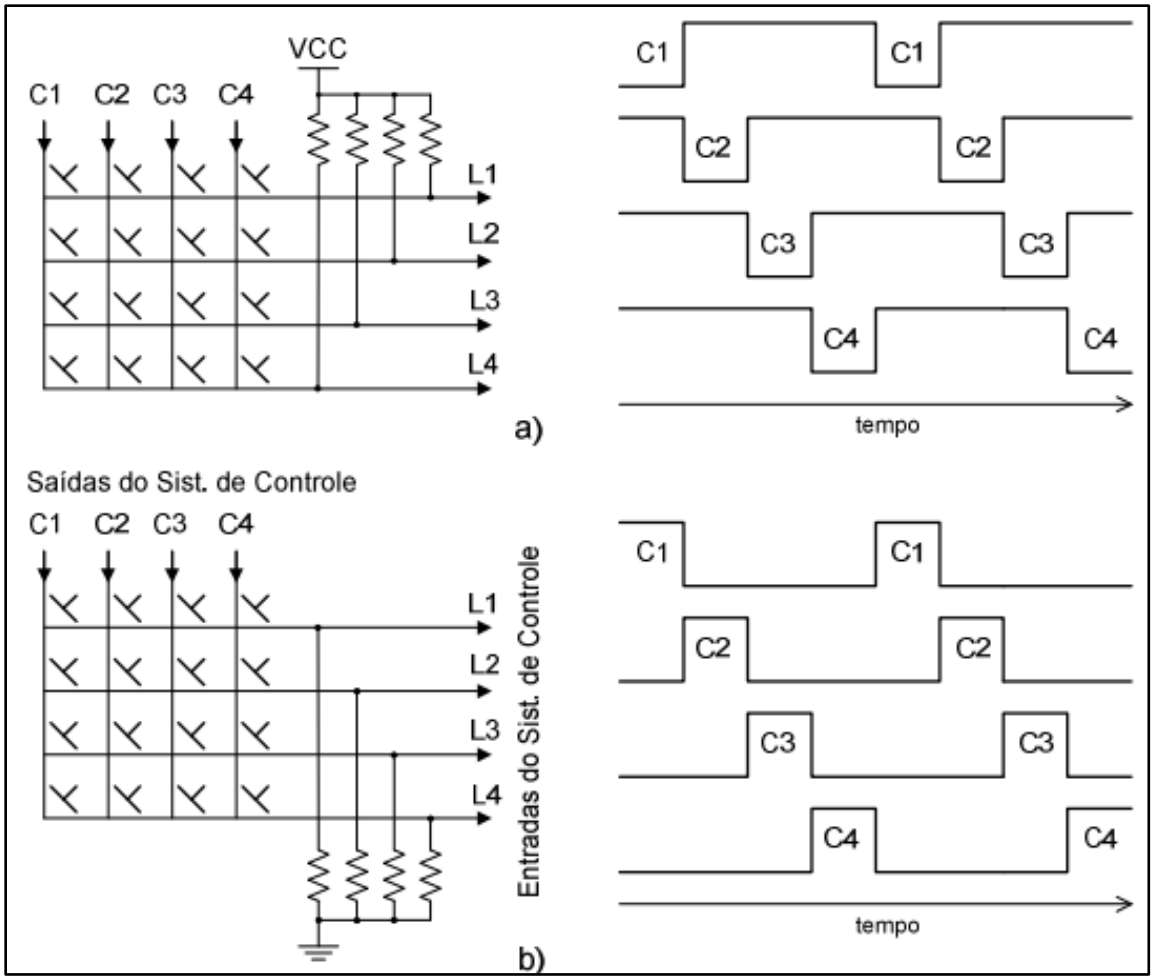
Teclado Matricial

- Teclado com organização matricial (para economizar pinos de E/S)



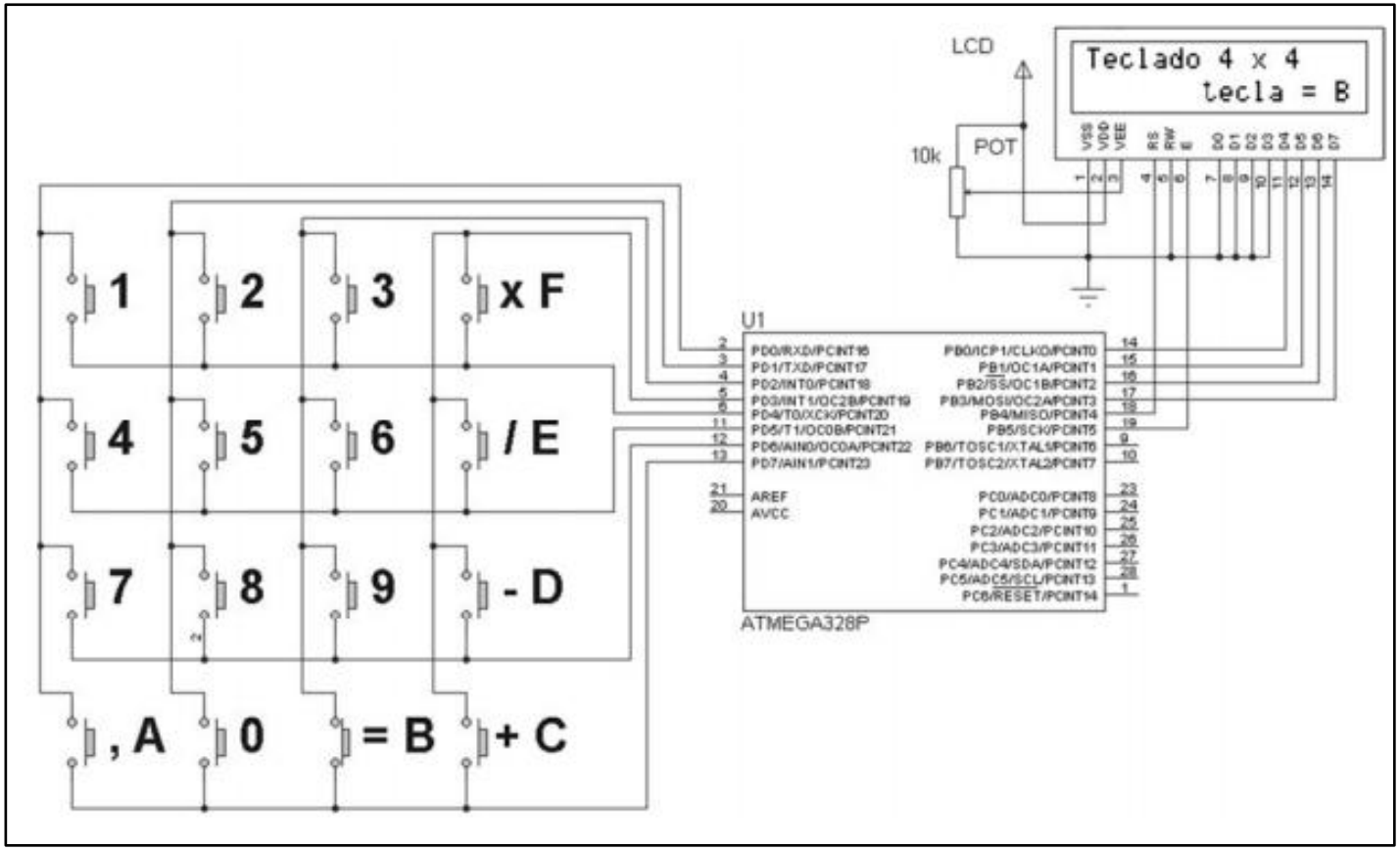
Teclado Matricial

- Utiliza varredura

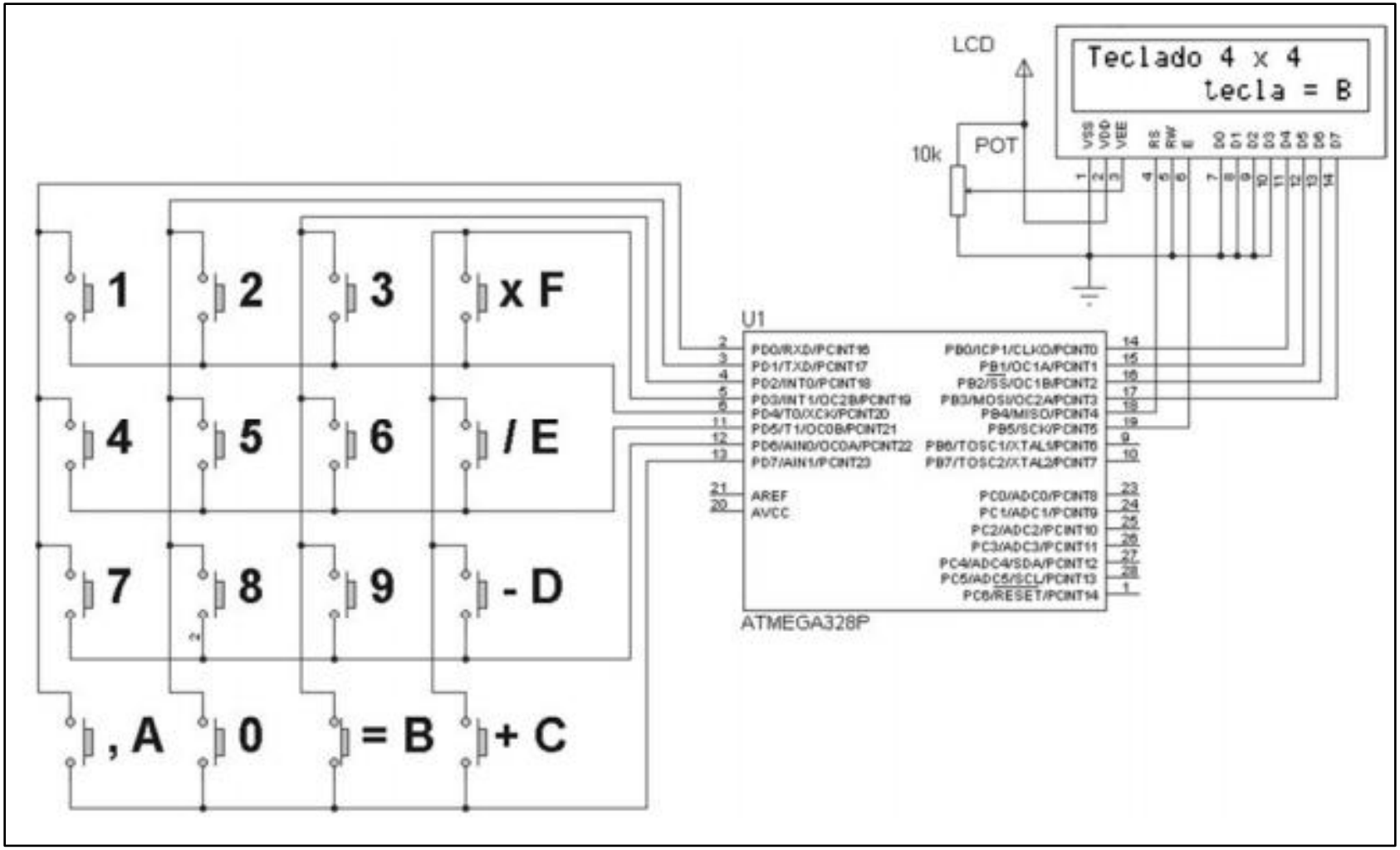


Teclado Matricial

- Exemplo



Teclado Matricial



Teclado Matricial

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 Matricial

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 Matricial

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
}
//--------------------------------------------------------------------------------
```