11. TÉCNICAS DE MULTIPLEXAÇÃO

Neste capítulo, algumas técnicas importantes de multiplexação são apresentadas. Elas são utilizadas para diminuir o número de componentes externos ao microcontrolador e/ou número de I/Os necessários. Trata-se de técnicas de multiplexação de sinais para emprego em *displays*, acionamento de conjuntos de LEDs (matriz) e outros dispositivos eletrônicos.

Quando são necessários vários pinos de I/O para o acionamento de um determinado circuito e o microcontrolador não os dispõem, é fundamental o emprego da multiplexação: técnica para transitar com vários dados em uma mesma via ou barramento. A multiplexação também é empregada para diminuir o número de vias e pode diminuir a complexidade física das placas de circuito impresso.

A ideia da multiplexação é dividir as atividades no tempo, empregando o mesmo meio físico para isso. A desoneração do hardware é substituída por um aumento na complexidade do software de controle e no tempo de execução das tarefas. Entretanto, devido à considerável velocidade de processamento dos sistemas envolvidos, geralmente isto não é um problema. As melhores técnicas de multiplexação empregam o menor número possível de componentes externos para cumprir as funções que devem ser desempenhadas pelo hardware.

11.1 EXPANSÃO DE I/O MAPEADA EM MEMÓRIA

Uma das principais técnicas de multiplexação empregada em sistemas microprocessados é ilustrada na fig. 11.1. Existem dois barramentos: um de dados e um de endereços. Cada dispositivo ligado aos barramentos responde a um ou mais endereços exclusivos e comunica-se com o microprocessador através do barramento de dados. Essa técnica é denominado expansão de I/O mapeada em memória.

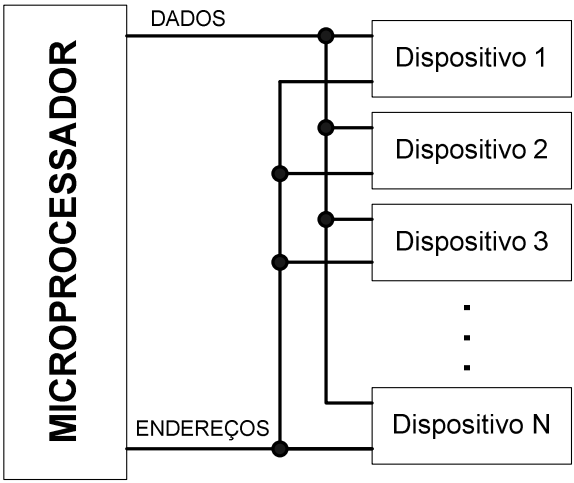

Fig. 11.1 – Controle de vários dispositivos em um sistema microprocessado.

O número de dispositivos diferentes que podem ser ligados ao barramento de dados depende do número de bits do barramento de endereços e é expresso por uma potência de 2. Por exemplo, se o barramento de endereços for de 6 bits, podem ser endereçados 64 dispositivos diferentes (2^6).

Se houver necessidade de escrita e leitura no dispositivo, pode-se empregar o bit menos significativo do seu endereço para essa informação. Supondo que um determinado dispositivo responda ao endereço 0b10101x,

quando o bit menos significativo desse endereço for 0, a operação pode ser de escrita no dispositivo e quando for 1, o dispositivo pode escrever no barramento de dados. Nesse caso, com 6 bits, poderiam ser endereçados 32 dispositivos diferentes (2^{6-1}).

No sistema da fig. 11.1, cada dispositivo é responsável pela decodificação do seu endereço e não deve responder a nenhum endereço diferente do seu. Assim, quando não estiver sendo acessado, o barramento de dados do dispositivo deve se portar como uma alta impedância, caso contrário, haverá colisão de dados (curtos-circuitos).

Na fig. 11.2, é apresentado um exemplo de multiplexação em que 32 entradas e 32 saídas são controladas com 12 pinos do ATmega328. O CI 74238 decodifica o endereço de entrada (pinos A, B e C), habilitando um único pino de saída (Y0-Y7); com 3 bits de entrada, tem-se, então, 8 saídas independentes (2^3). Nessas, os quatro bits menos significativos habilitam 4 *latches*, os CIs 74373 (saídas). Cada *latch* habilitada colocará os dados do barramento de dados na saída e os manterá lá até a ocorrência de um novo sinal de habilitação. Na parte mais significativa dos bits de saída do CI 74238, encontram-se 4 *buffers tri-state* (CIs 74244). Esses CIs quando habilitados, transferem os dados de suas entradas para o barramento de dados, e quando desabilitados, mantém suas saídas em alta impedância. O microcontrolador escreve o endereço do CI que deseja acessar no barramento de endereços, e faz sua escrita ou leitura. O pino de habilitação (E1) do 74238 foi empregado para desabilitar todos os CIs quando desejado.

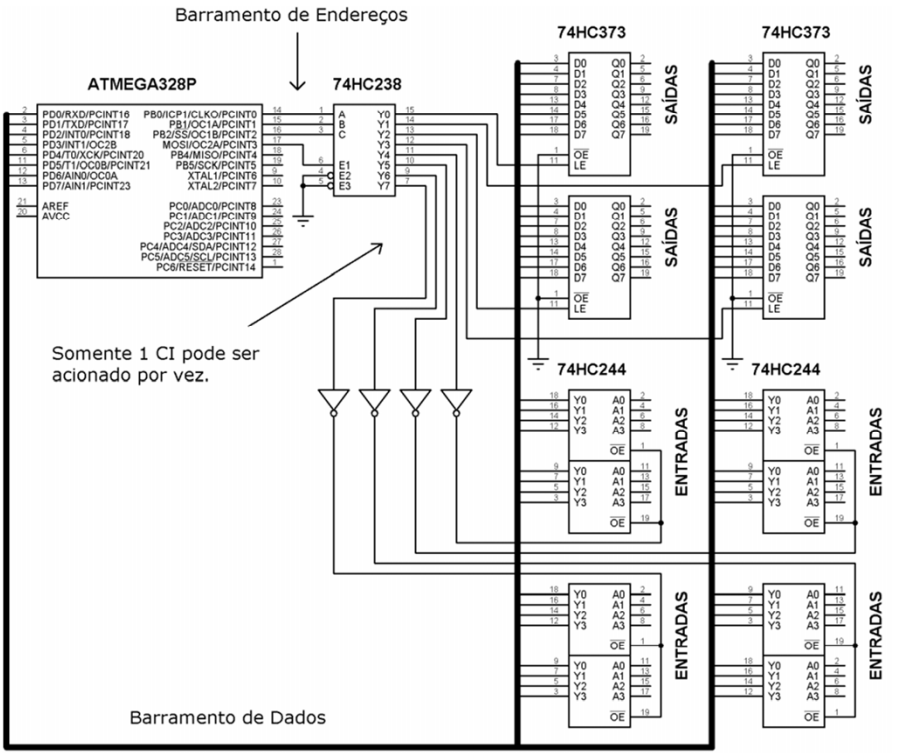

Fig. 11.2 – Exemplo de um sistema multiplexado com 32 saídas e 32 entradas, empregando 12 vias de controle.

Com o circuito da fig. 11.2, houve uma expansão considerável de entradas e saídas do microcontrolador. Com um barramento de dados de 8 bits e um de endereços de 3 bits conseguiu-se, com componentes externos, uma expansão para 32 saídas e 32 entradas.

Outra possibilidade para utilizar menos pinos de I/O do microcontrolador no endereçamento de dispositivos externos é o uso de um contador de década, que ativa uma de suas saídas a cada pulso de *clock* na sua entrada. Na fig. 11.3, o circuito que ilustra esse conceito é apresentado. O contador 4017 poderia substituir o 74238 da fig. 11.2 realizando uma contagem contínua de 0 até 7. Agora, poderiam ser acessados os mesmos I/Os que no circuito da fig. 11.2, com uma

diminuição de 2 pinos do barramento de dados. Entretanto, dessa forma, não é mais possível acessar qualquer um dos dispositivos do barramento aleatoriamente (com um endereço), é necessário realizar o acesso sequencial deles através dos pulsos de *clock* enviados ao 4017.

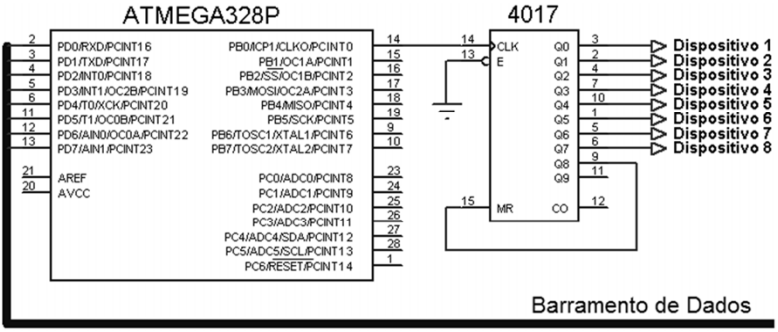

Fig. 11.3 – Sistema multiplexado com 32 saídas e 32 entradas, empregando 9 vias de controle.

11.2 CONVERSÃO SERIAL-PARALELO

Outra técnica para a expansão de I/Os é a conversão de dados seriais para paralelos, servindo para aumentar o número de saídas de um sistema. Muitos projetos necessitam dessa forma de expansão de I/O para cumprirem eficientemente suas funções com um mínimo de componentes externos. O custo no uso dessa técnica é um aumento da complexidade do software de controle e uma diminuição da velocidade de processamento devido aos componentes envolvidos.

Circuitos integrados interessantes para a conversão serial-paralelo são o 74595 e o 4094 (*shift registers*). Eles possuem saídas com alta impedância e *latchs* internos para que os dados fiquem armazenados nas saídas, permitindo a ligação dos CIs em série (cascateamento). Na fig. 11.4, são apresentados três 4094 ligados em cascata, nos quais a saída de dados de um é ligada na entrada de dados do outro e, assim, sucessivamente. O microcontrolador colocará 24 bits de dados seriais no pino PB0, seguidos

por 24 pulsos de *clock* no pino PB1. A cada pulso de *clock*, o bit de entrada do primeiro 4094 empurra o bit de entrada anterior na fila, preenchendo os registradores internos dos CIs em cascata. Funciona semelhante a uma fila: a entrada de um bit desloca o próximo e, assim, sucessivamente até o preenchimento de todas as vagas. Depois que os 4094 foram preenchidos, é gerado o sinal de *strobe* (pulso) no pino PB2, o que resulta na transmissão dos dados internos dos *latchs* para os pinos de saída, os quais ficam presos lá até um novo sinal de *strobe*. Em resumo, preenchem-se todos os *shift registers* e depois se habilita a saída dos seus dados. Estes ficarão inalterados nas saídas até um novo sinal de *strobe* e nesse intervalo, os *shift registers* podem receber novas informações.

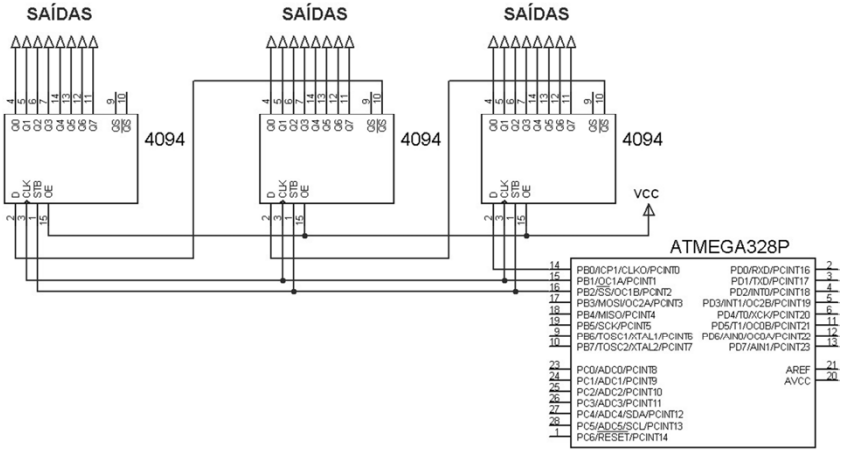

Fig. 11.4 – Emprego do CI 4094 para a conversão serial-paralelo na expansão do número de pinos de saídas de um sistema.

O número de pinos de saída de uma expansão serial-paralelo depende exclusivamente do número de CIs ligados em cascata, visto que o número de pinos de controle será sempre o mesmo.

Na sequência, é apresentado um programa com um sub-rotina exemplificando o envio de 3 bytes para os 4094 de acordo com o circuito da fig. 11.4. O sinal de *strobe* é gerado pelo software de controle para

transferir a informação para as saídas dos 4094. No caso do programa, a sub-rotina de envio é utilizada 3 vezes (envio de 3 bytes) antes da geração do pulso de *strobe*.

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
}
//===============================================================================
```

O arquivo **def_principais.h** foi visto nos capítulos anteriores.

11.3 CONVERSÃO PARALELO-SERIAL

Dual ao sistema serial-paralelo, a conversão de dados paralelo para serial serve para aumentar o número de entradas de um determinado sistema. Na fig. 11.5, é apresentado um circuito exemplo empregando três CIs 74165 conectados em cascata. Quando o sinal de carga (*load*) é aplicado, as informações constantes na entrada dos CIs são transferidas para os seus registradores internos. Após, à medida que os pulsos de *clock* são gerados, os dados são transferidos serialmente para a saída de dados do sistema. Assim, com 24 pulsos de *clock* consegue-se ler os três 74165 da fig. 11.5. O número de entradas depende exclusivamente do número de conversores paralelo-serial conectados em cascata; para o controle são empregados sempre três pinos.

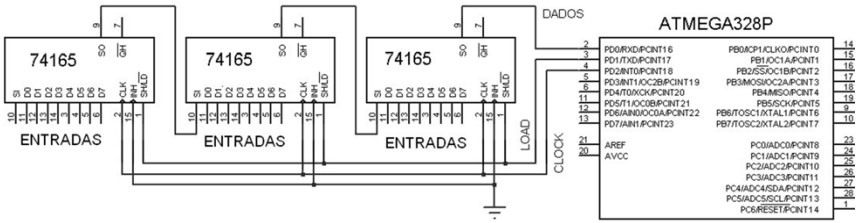

Fig. 11.5 – Emprego do CI 74165 para a conversão paralelo-serial na expansão de entradas de um sistema.

11.4 MULTIPLEXAÇÃO DE DISPLAYS DE 7 SEGMENTOS

Nossos olhos são incapazes de perceber a alteração de um sinal luminoso com frequência acima de 24 Hz, essa característica humana é chamada persistência da visão. Isso significa que se um LED piscar 24 vezes em um segundo, ele aparentará aos olhos humanos como ligado constantemente. Todavia, um sinal luminoso com frequência de 24 Hz ainda produz cintilação na sua percepção. Desta forma, para uma visualização agradável, emprega-se um sinal de pelo menos 48 Hz.

Sem utilizar a persistência da visão, para ligar diretamente 4 *displays* de 7 segmentos, seriam necessários 8×4 = 32 pinos do microcontrolador (considerando o uso do ponto do *display).* O problema, além do grande número de pinos, seria a corrente máxima que poderia chegar a 320 mA, no caso de se usar 10 mA por segmento do *display*. Para resolver esse problema, emprega-se a multiplexação temporal baseada na persistência da visão. Na fig. 11.6, é ilustrado um circuito para executar essa função, *displays* anodo comum ligados através de transistores PNP são empregados, ativos quando a base deles é colocada em 0 V (nível lógico 0). Os LEDs são ligados com o mesmo nível de tensão e o circuito de controle drena a corrente deles. O circuito será similar para *displays* catodo comum, nesse caso deve-se empregar transistores NPN, os níveis de ativação serão de 5V (ou nível lógico 1) e o sistema de controle fornecerá corrente aos LEDs dos *displays*.

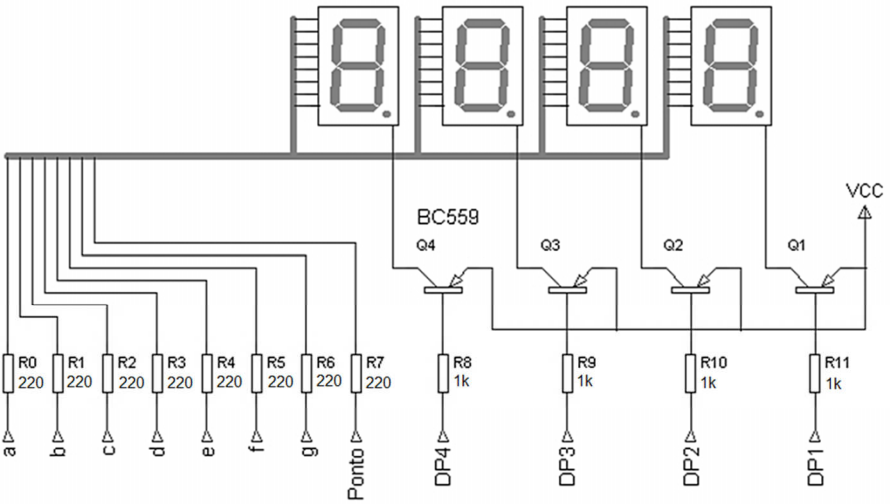

Fig. 11.6 - Acionamento de 4 *displays* de 7 segmentos[30].

[30] Se o valor de VCC for maior que a tensão de desligamento (nível lógico) na base dos transistores PNP, eles poderão ficar diretamente polarizados e conduzir indevidamente. Transistores PNP funcionam adequadamente quando a tensão de alimentação dos *displays* (VCC) for a mesma do circuito digital de controle, caso contrário, transistores NPN devem ser utilizados.

Num sistema microcontrolado, a multiplexação temporal para *displays* é feita através da chamada varredura. Para os *displays* da fig. 11.6, isso é realizado da seguinte maneira:

1. No início, os *displays* estão apagados e os transistores desabilitados.
2. O dado referente ao primeiro *display* é colocado no barramento.
3. O *display* referente ao dado é ligado, o transistor adequado é acionado.
4. Espera-se o tempo necessário para o acionamento do próximo *display*.
5. Apaga-se o *display* ligado.
6. Coloca-se o dado referente ao próximo *display* da sequência.
7. O *display* é ligado, o transistor adequado é acionado.
8. O processo se repete de forma contínua para cada *display*.

Para que a persistência da visão funcione, cada *display* deve ser ligado com uma frequência mínima de 48 Hz, é usual empregar-se 50 Hz ou mais. Desta forma, considerando-se 4 *displays*, a frequência de varredura do sistema deve ser de 200 Hz (4 × 50 Hz). Isso significa que cada *display* ficará ligado durante um ciclo da varredura e 3 desligados, resultando em 5 ms de acionamento para cada um (1/200). Obviamente, o brilho conseguido com um sistema multiplexado é menor do que em um sistema sem multiplexação. Assim, para se ter uma boa luminosidade é importante acionar os LEDs com a maior corrente possível.

Na programação de um microcontrolador, a forma mais elegante de apresentar a mensagem num conjunto de *displays* é fazer a varredura dentro da rotina de interrupção de algum contador, liberando o programa principal para outras atividades e simplificando o código. Na fig. 11.7, é apresentado o fluxograma de um programa de controle para 4 *displays* empregando a interrupção de um contador. Na sequência, um código

exemplo[31] desenvolvido para o Arduino, conforme circuito da fig. 11.8 e que apresenta o número 3210 (fig. 11.9).

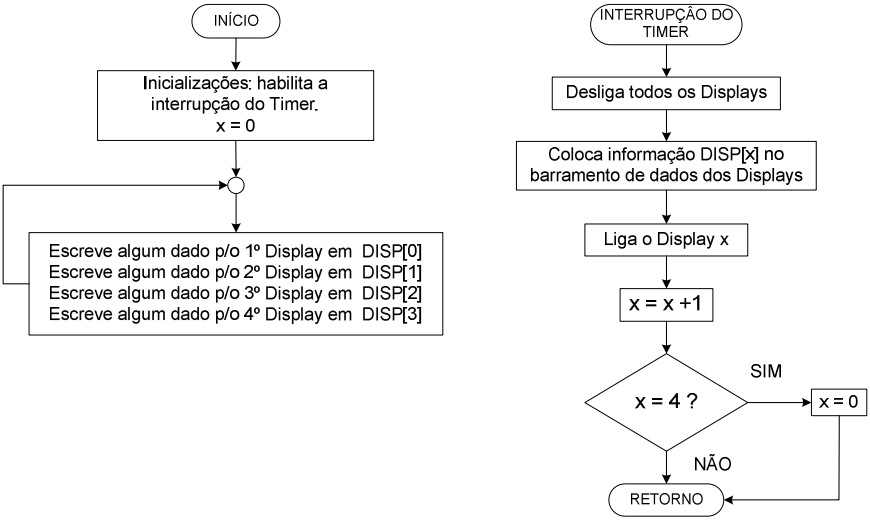

Fig. 11.7 - Fluxograma para o acionamento de 4 *displays* de 7 segmentos.

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

[31] Na seção 5.4, foram apresentados o *display* de 7 segmentos e seus códigos de decodificação (tab. 5.2).

```
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
     }
}
//============================================================================
```

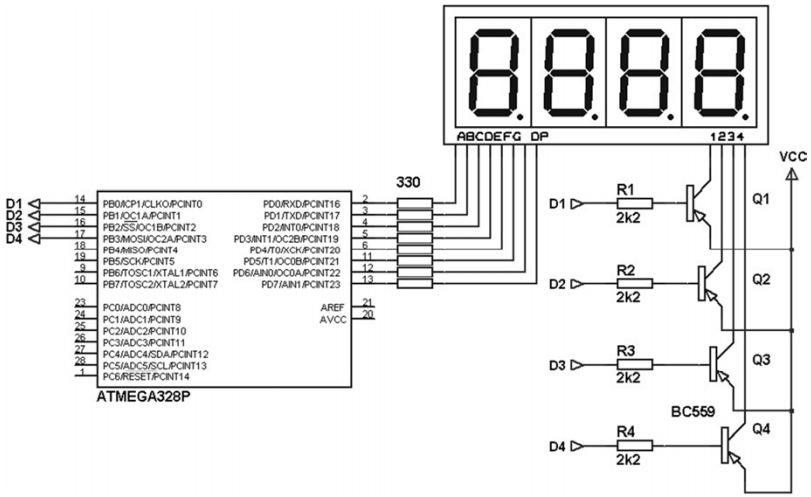

Fig. 11.8 Circuito para o acionamento de 4 *displays* de 7 segmentos anodo comum.

Fig. 11.9- Resultado da escrita multiplexada em 4 *displays* de 7 segmentos.

Exercícios:

11.1 – Baseado no exercício 5.12, fig. 5.11a, ligue sequencialmente os 8 LEDs. Comece com uma frequência visível e a aumente progressivamente até que os LEDs pareçam estar todos ligados. Qual o tempo que cada LED ficou ligado para a persistência da visão?

11.2 – Empregando dois *displays* de 7 segmentos e um botão, desenvolva um programa para o sorteio aleatório e com mesma probabilidade de ocorrência dos números de 1 até 60 (Mega Sena). O número sorteado não deve voltar ao sorteio.

11.3 – Elaborar um programa para que o hardware da fig. 11.10 funcione como relógio 24 h. A entrada de sinal para contagem dos segundos é de 60 Hz. O ajuste do horário deve ser feito nos botões específicos.

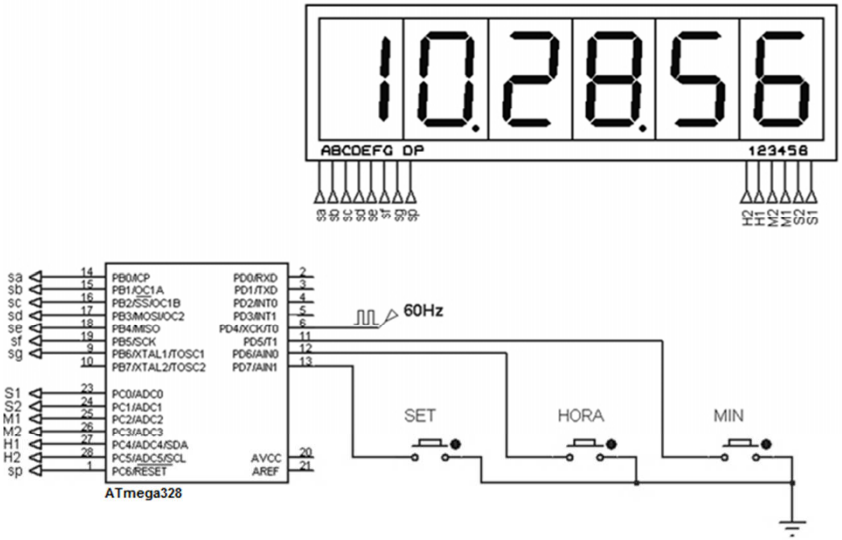

Fig. 11.10 – Relógio com contagem baseado na frequência da rede elétrica (circuito simplificado).

11.5 UTILIZANDO MAIS DE UM LED POR PINO

Como é possível três estados no pino do microcontrolador (0, 1 e alta impedância - entrada sem *pull-up*), um arranjo adequado de LEDs pode resultar em um pino acionando mais de um LED. Na fig. 11.11, é apresentado o circuito para controlar 2 LEDs utilizando apenas um pino de I/O do microcontrolador. Nesse caso, para apagar os LEDs basta colocar o pino em alta impedância; para ligar os dois LEDs em conjunto é necessário empregar a multiplexação temporal, ligando rapidamente cada um deles. Como existe um divisor de tensão entre os resistores R1 e R2, LEDs com uma tensão de trabalho superior à tensão sobre R1 ou R2 não funcionarão adequadamente, pois a tensão fornecida não será suficiente para ligá-los.

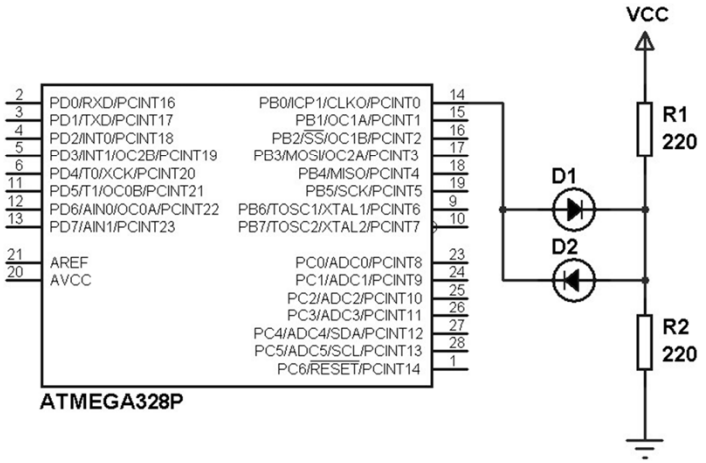

Fig. 11.11 – Empregando 1 pino do microcontrolador para controlar 2 LEDs.

Para ligar 6 LEDs com 3 pinos, pode ser empregado o circuito da fig. 11.12. Na tab. 11.1, é apresentada a configuração que deve ser feita nos pinos do circuito e sua correspondência no controle dos LEDs.

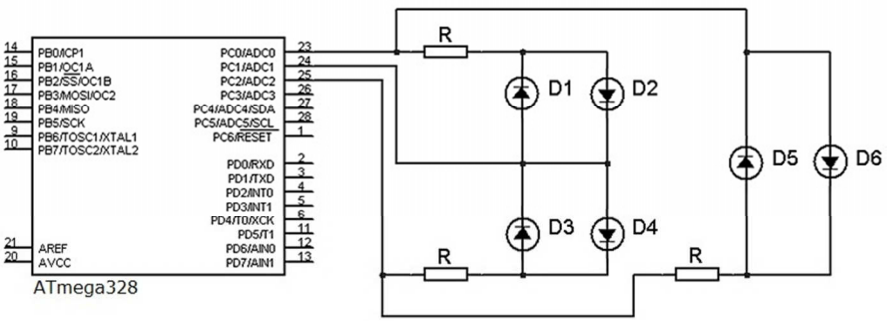

Fig. 11.12 – Empregando 3 pinos do microcontrolador para controlar 6 LEDs.

Tab. 11.1 – Configuração dos pinos PC0..2 da fig. 11.10 e sua correspondência no controle de 6 LEDs (1 indica LED aceso).

Pinos				LEDs					
PC0	**PC1**	**PC2**		**1**	**2**	**3**	**4**	**5**	**6**
0	0	0		0	0	0	0	0	0
0	1	Hi-Z		1	0	0	0	0	0
1	0	Hi-Z		0	1	0	0	0	0
Hi-Z	0	1		0	0	1	0	0	0
Hi-Z	1	0		0	0	0	1	0	0
0	Hi-Z	1		0	0	0	0	1	0
1	Hi-Z	0		0	0	0	0	0	1
0	0	1		0	0	1	0	1	0
0	1	0		1	0	0	1	0	0
0	1	1		1	0	0	0	1	0
1	0	0		0	1	0	0	0	1
1	0	1		0	1	1	0	0	0
1	1	0		0	0	0	1	0	1
1	1	1		0	0	0	0	0	0

Se houver a necessidade do emprego de alguma configuração de LEDs que não conste na tab. 11.1, é necessário empregar a persistência da visão para acionar sequencialmente os LEDs desejados.

O número de LEDs controlados dependerá do número de pinos utilizados e pode ser aumentado usando a mesma lógica acima. O número

de LEDs que podem ser controlados, de acordo com o número de pinos, é dado por:

$$\mathrm{Nr}_{\mathrm{LEDs}} = \mathrm{Nr}_{\mathrm{pinos}} \times (\mathrm{Nr}_{\mathrm{pinos}} - 1) \qquad (11.1)$$

Exercício:

11.4 – Baseado no exercício 5.14, crie um dado eletrônico empregando somente 3 pinos para o controle dos LEDs.

11.6 MATRIZ DE LEDs

Um sistema muito comum para a divulgação visual de mensagens publicitárias e informativas é a matriz de LEDs (ver a fig.11.13), onde um conjunto organizado de LEDs, formando um painel de pixels (1 LED = 1 pixel), é comandado por um sistema microcontrolado. Como a quantidade de informação para formar uma imagem é grande, existe a necessidade da multiplexação dos dados e o emprego de algum sistema de varredura (o sistema baseia-se na persistência da visão).

Os LEDs em uma matriz são organizados em linhas e colunas[32]. Assim, na fig. 11.13, por exemplo, todos os anodos estão conectados à linha que alimenta todo o conjunto de LEDs; por sua vez, os catodos dos LEDs estão conectados às colunas. O sistema funciona da seguinte maneira: as informações referentes à primeira linha da matriz são preenchidas; todas as colunas conterão as informações referentes a cada LED e, então, a linha é alimentada, ligando-se os LEDs correspondentes. O processo é repetido para cada linha, num processo de varredura. Como o processo é feito rapidamente, a mensagem na matriz parecerá estática aos olhos humanos.

[32] Existem matrizes comerciais onde os LEDs são encapsulados em um único bloco. Os formatos usuais são: 7 × 5, 8 × 8 e 16 × 16.

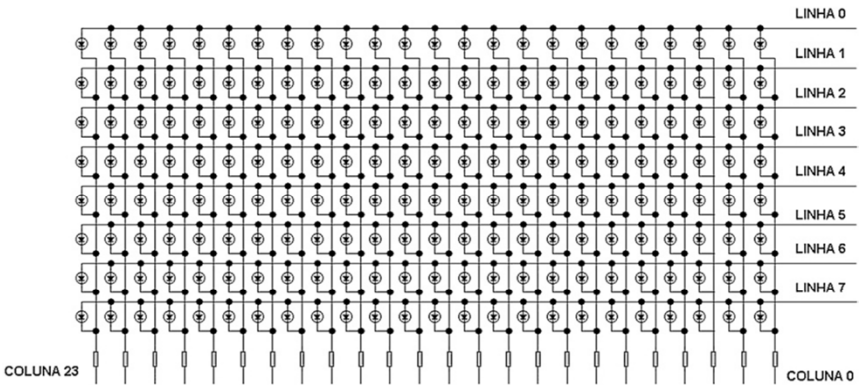

Fig. 11.13 – Matriz de LEDs com 8 linhas e 24 colunas.

Um circuito simplificado de controle para os 192 LEDs (24×8) da figura anterior é apresentado na fig. 11.14. Todos os LEDs são controlados com apenas 3 sinais, empregando-se a conversão serial-paralelo com o registrador de deslocamento 74HC595. Cada linha é alimentada com o emprego de um transistor PNP. Existe a necessidade de se empregar apenas um resistor por coluna, pois, na varredura, apenas um LED da coluna pode estar ligado. O programa de controle envia 24 bits de dados, correspondentes aos LEDs das colunas, mais 8 bits correspondentes à chave transistorizada a ser ligada (linha). Após esses 32 bits, o sinal de *strobe* habilita todos os LEDs da linha correspondente. Com oito linhas, serão oito trocas de linha na varredura.

No projeto do circuito, o transistor de cada linha deve ser capaz de suprir a corrente máxima dada pelo número de LEDs da linha multiplicado pela corrente máxima individual por LED. No caso acima, o 74HC595 pode drenar em torno de 8 mA por pino. Desta forma, se todos os LEDs de uma linha forem ligados, a corrente total da linha será de 192 mA (24 × 8 mA). Como na varredura das linhas (supondo 8), cada linha ficará ligada durante um período e por sete desligada, para a obtenção do maior brilho, é importante trabalhar com os LEDs na máxima corrente possível. Da mesma forma, o componente eletrônico responsável pela informação da

coluna (registrador de deslocamento) deve ser capaz de drenar a corrente exigida pelo LED da coluna. Caso ele não suporte a corrente desejada, um *driver* de corrente adequado[33] deve ser empregado (ver o apêndice D).

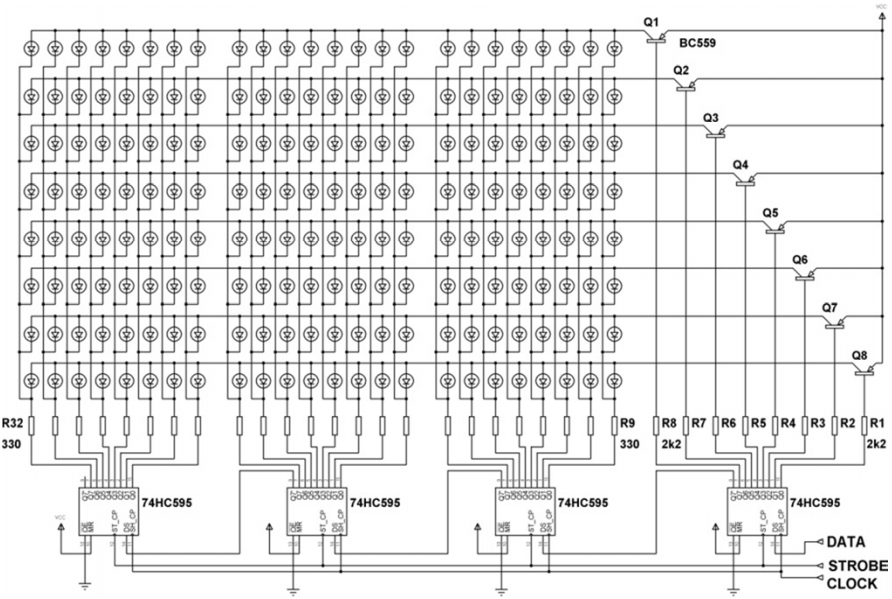

Fig. 11.14 – Circuito simplificado para o controle de uma matriz de 8 × 24 LEDs.

Na confecção da placa de circuito impresso, o desenhista colocará os LEDs da matriz na posição adequada de acordo com o desenho do circuito, ou utilizará matrizes comerciais. Na fig. 11.15, é ilustrada a visão tridimensional de uma placa de circuito impressa obtida no programa ARES (Proteus) para um circuito com 256 LEDs (8 × 32).

[33] O 74HC595 suporta em torno de 8 mA por pino, caso se deseje uma corrente maior, é necessário o emprego de um *driver* de corrente, como por exemplo o ULN2803. Existem também registradores de deslocamento com capacidade bem maior de corrente, como o TPIC6B595.

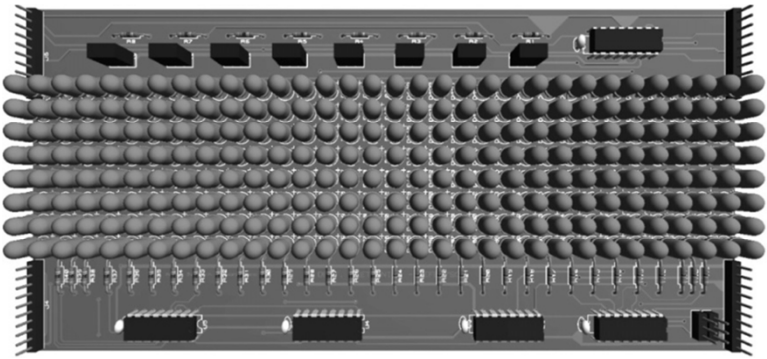

Fig. 11.15 – Visão tridimensional de uma matriz de LEDs .

Na programação de uma matriz de LEDs, a informação é colocada em uma tabela na memória do sistema de controle, que se encarrega de ler a informação, preencher adequadamente as informações dos registradores de deslocamento e gerar os sinais de controle. Em sistemas mais elaborados, como os grandes painéis de publicidade, um computador é o responsável pelo controle, com um software adequado convertendo as imagens para a matriz de LEDs do painel. Em sistemas complexos, que permitem a reprodução de várias cores, são empregados LEDs tricolores (RGB – *Red-Green-Blue*), tornando mais complexos o circuito e o software de controle.

11.7 CUBO DE LEDS

Um cubo de LEDs[34] é um conjunto de LEDs organizado de forma tridimensional. É empregado para animações gráficas, proporcionando um efeito visual interessante. O espaçamento entre os LEDs é tal, que a maioria dos LEDs da estrutura podem ser vistos (fig. 11.16).

[34] Cubos comerciais podem ser encontrados em www.seekway.com.cn.

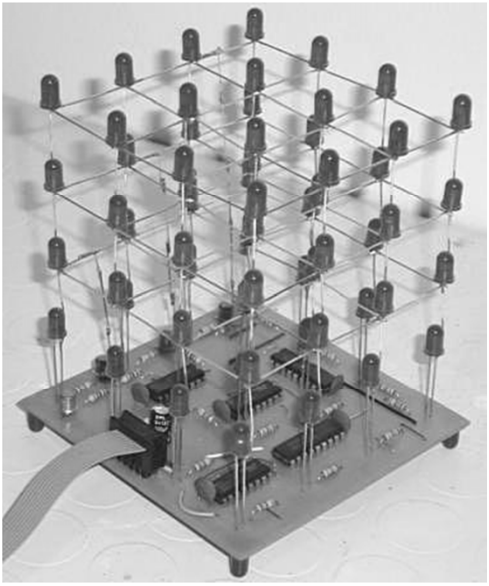

Fig. 11.16 – Cubo de LEDs de 4 × 4 × 4 (64).

A mesma técnica empregada para o acionamento de uma matriz de LEDs é utilizada para o cubo (ver a seção anterior). Em uma matriz, os LEDs são dispostos em linhas e colunas, formando um plano bidimensional, onde todos os LEDs de uma linha são habilitados ao mesmo tempo, formando a imagem por varredura. Em um cubo, os LEDs devem ser organizados em planos horizontais sobrepostos, formando um arranjo tridimensional, conforme apresentado na fig. 11.16. Nela, cada plano é composto por 16 LEDs (4 × 4) e, ao todo, são 4 planos totalizando 64 LEDs (4 × 16).

Para a formação de uma imagem no cubo, pode ser empregada a varredura por plano. Ou seja, um plano horizontal inteiro de LEDs é acionado ao mesmo tempo. Assim, o acionamento sequencial dos planos forma uma imagem tridimensional. Em resumo, numa matriz de LEDs, uma linha inteira é energizada ao mesmo tempo na varredura, no cubo, um plano. Na geração dos dados para o cubo, é necessário que o

programador consiga visualizar tridimensionalmente a animação que deseja realizar e organizar adequadamente a varredura de acordo com o circuito de acionamento.

Na fig. 11.17, é apresentado o circuito eletrônico do cubo da fig. 11.16, pronto para ser conectado a um microcontrolador. O circuito emprega o CI ULN2803 para drenar a corrente dos LEDs (ver a seção D.2 do apêndice). Em cada plano horizontal, os anodos dos LEDs estão conectados, sendo a alimentação realizada através de uma chave transistorizada. Na sobreposição dos planos, onde os LEDs estarão uns sobre os outros, verticalmente, os catodos estão conectados e ligados às saídas dos CIs 4094[35] (responsáveis pela informação que irá aparecer em cada LED). Por exemplo, no circuito da fig. 11.17, os LEDs 1a, 2a, 3a e 4a estarão alinhados verticalmente.

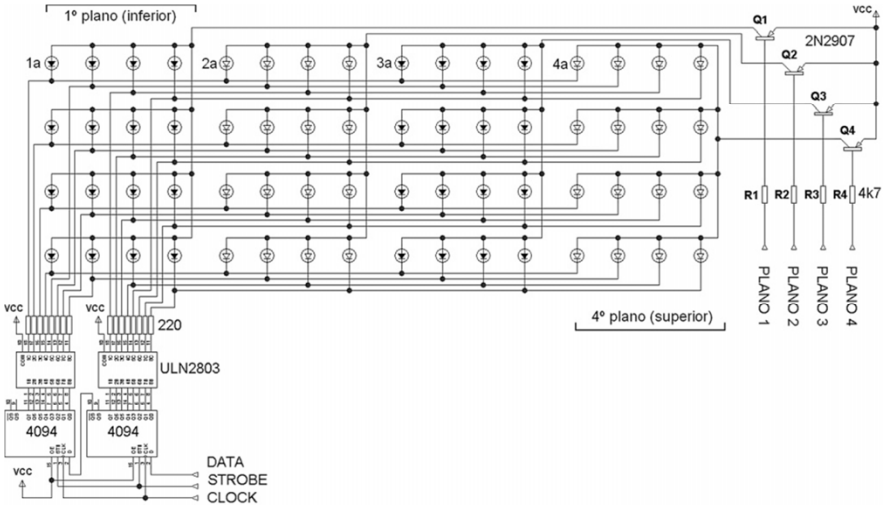

Fig. 11.17 – Circuito de um cubo de LEDs de 4 × 4 × 4.

[35] Outro registrador de deslocamento similar ao 74HC595, entretanto com menor capacidade para suprir ou fornecer corrente.

A grande dificuldade no projeto de um cubo de LEDs é a montagem da estrutura tridimensional; quanto maior o número de LEDs, maior é a complexidade. Uma possibilidade é montar os planos horizontais individualmente com o emprego de um gabarito, e depois, um a um, soldá-los para a sobreposição. Na fig. 11.16, o plano inferior foi soldado diretamente na placa de circuito impresso, os demais sobrepostos a ele, através da solda dos catodos dos LEDs.

Da mesma forma que em uma matriz, é possível empregar LEDs tricolores no cubo (RGB). Entretanto, a montagem será bem mais difícil e o circuito de controle bem mais complexo.

Exercícios:

11.5 – Na fig. 11.18, é apresentado um sistema para o controle de uma matriz com 192 LEDs (8 × 24). O sistema alimenta uma linha por vez. Os dados correspondentes a cada coluna, incluindo qual linha será alimentada, são fornecidos por um conjunto de registradores de deslocamento (*shift register* – 74HC594, conversor serial-paralelo). Para controlar todo o sistema são empregadas 3 saídas do microcontrolador.

- Faça um programa para apresentar uma mensagem estática na matriz de LEDs. Antes da programação pesquise os módulos de matriz de LEDs disponíveis no mercado.
- Como o programa pode apresentar mensagens em movimento? Neste caso o pino de limpeza dos registradores (MR) teria um papel importante?

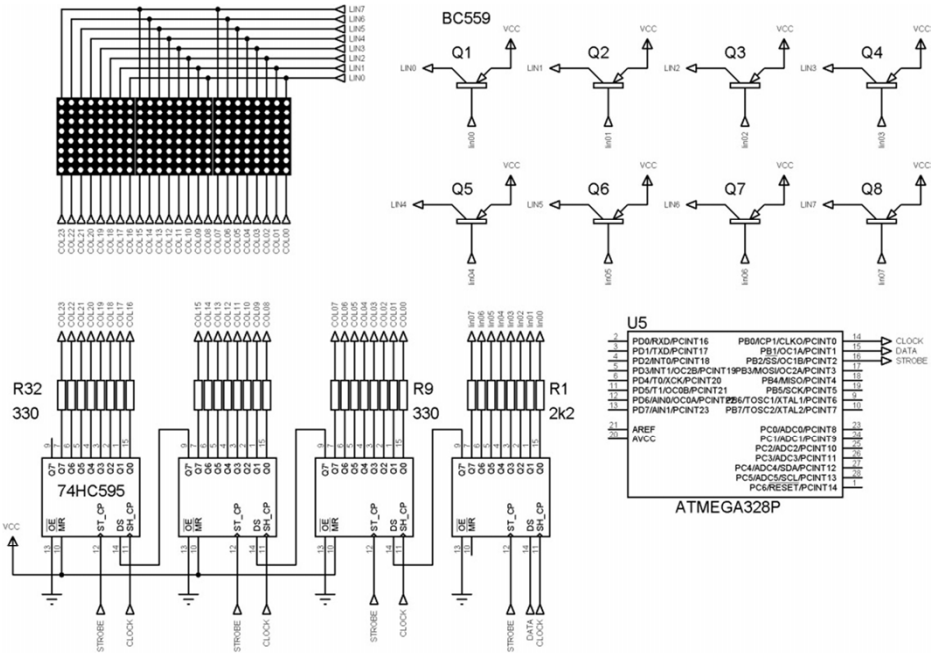

Fig. 11.18 – Matriz de LEDs e o 74HC595.

11.6 – Outra possibilidade para a montagem de um cubo de LEDs é ligar fisicamente os LEDs em colunas independentes, de forma a não existir ligação fisica entre as colunas, não sendo realizadas conexões na forma de planos. Como seria o circuito para o controle de tal cubo?

11.7 – Baseado no processo de multiplexação, empregando o ATmega328, projete um CLP (Controlador Lógico Programável) básico com 64 saídas e 64 entradas digitais.
