Exercícios

5. PORTAS DE ENTRADA E SAÍDA (I/Os)

5.1 – Qual o tempo aproximado e exato gasto pelo ATmega para a execução da sub-rotina abaixo?

```
ATRASO:
        LDI  R19,X     //carrega R19 com o valor X (dado abaixo)
   volta:
        DEC  R18       //decrementa R18, começa com 0x00
        BRNE volta     //enquanto R18 > 0 volta a decrementar R18
        DEC  R19       //decrementa R19
        BRNE volta     //enquanto R19 > 0 vai para volta
        RET
```

a) X = 0, f_{clk} = 16MHz.

b) X = 10, f_{clk} = 1MHz.

5.2 – Qual o tempo aproximado gasto para a execução da sub-rotina abaixo para uma frequência de operação do ATmega de 8 MHz (fluxograma na fig. 4.4b)?

```
ATRASO:
        LDI  R19,0x02
    volta:
        DEC  R17       //decrementa R17, começa com 0x00
        BRNE volta     //enquanto R17 > 0 fica decrementando R17
        DEC  R18       //decrementa R18, começa com 0x00
        BRNE volta     //enquanto R18 > 0 volta a decrementar R18
        DEC  R19       //decrementa R19
        BRNE volta     //enquanto R19 > 0 vai para volta
        RET
```

5.3 – Desenvolva uma sub-rotina em *assembly* para produzir um atraso de aproximadamente 0,5 s para o ATmega operando a 16 MHz.

5.4 – Para o programa abaixo, escrito na linguagem C, qual é o tempo aproximado gasto pelo ATmega para a sua execução (f_{clk} = 20 MHz). Considere que são gastos 3 ciclos de *clock* para que uma repetição do laço **for** seja realizada.

```
unsigned int i, j;

for(i=256; i!=0; i--)
{
     for(j=65535; j!=0; j--);
}
```

5.5 – Faça um programa em *assembly* para piscar um LED a cada 1 s.

5.6 – Utilizando a macro para complementar um bit (cpl_bit), faça um programa para piscar um LED a cada 500 ms.

5.7 – Desenvolva um programa para piscar um LED rapidamente 3 vezes e 3 vezes lentamente.

5.8 – Utilizando o deslocamento de bits crie um programa em *assembly* que ligue 8 LEDs[1] (ver a fig. 5.4a), da seguinte forma:

a) Ligue sequencialmente 1 LED da direita para a esquerda (o LED deve permanecer ligado até que todos os 8 estejam ligados, depois eles devem ser desligados e o processo repetido).

b) Ligue sequencialmente 1 LED da esquerda para a direita, mesma lógica da letra a.

c) Ligue sequencialmente 1 LED da direita para a esquerda, desta vez somente um LED deve ser ligado por vez.

d) Ligue sequencialmente 1 LED da esquerda para a direita e vice-versa (vai e volta), só um LED deve ser ligado por vez.

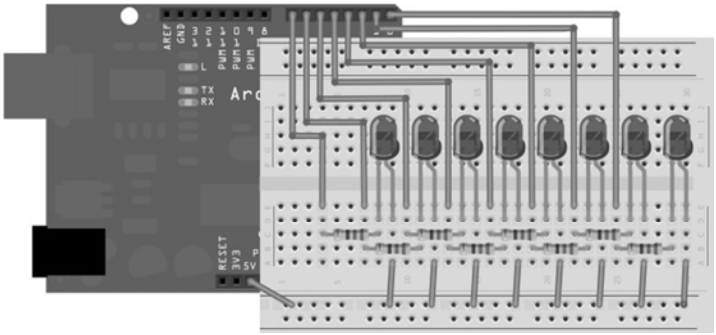
e) Ligue todos os LEDs e apague somente 1 LED de cada vez, da direita para a esquerda e vice-versa (vai e volta), somente um LED deve ser apagado por vez.

[1] Como o Arduino utiliza um programa de *boot loader*, ele emprega os pinos TXD e RXD para a gravação da memória de programa (pinos PD0 e PD1). Quando não se utiliza a IDE do Arduino para a gravação, esse pinos devem ser configurados explicitamente pelo programador para serem pinos de I/O genéricos, caso contrário os pinos não se comportarão como esperado (configuração *default*). Em C é necessário acrescentar a seguinte linha de código: **UCSR0B = 0x00;** //desabilita RXD e TXD.

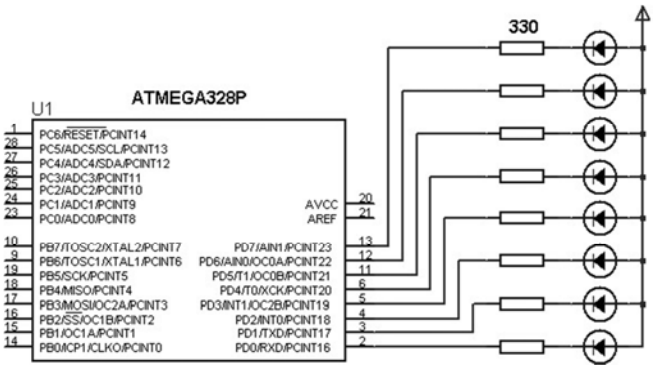
Importante: quando se grava o Arduino, os pinos 0 e 1 (PD0 e PD1) não devem estar conectados a nenhum circuito, caso contrário, poderá haver erro de gravação. Se algum *shield* utilizar os referidos pinos, é aconselhável gravar primeiro o Arduino e somente depois conectá-lo.

f) Mostre uma contagem binária crescente (0-255) com passo de 250 ms.

g) Mostre uma contagem binária decrescente (255-0) com passo de 250 ms.



a)



b)

Fig. 5.4 – Sequencial com 8 LEDs: a) montagem para o Arduino e b) esquemático.

5.9 – Elaborar um programa para ligar imediatamente um LED após o pressionar de um botão, com uma rotina de atraso de 10 ms para eliminação do *bounce*.

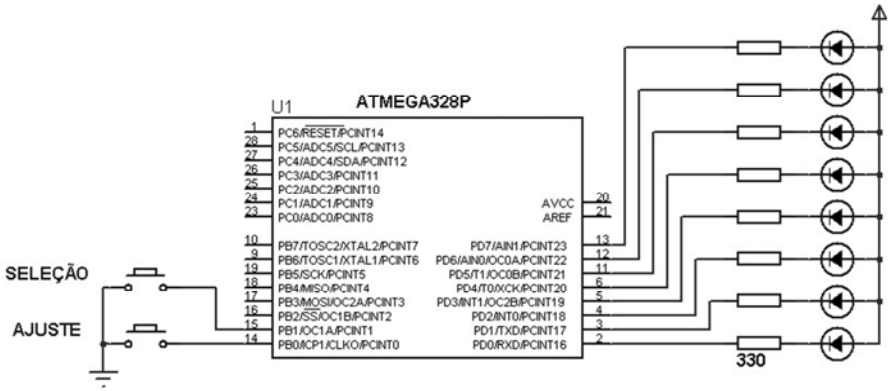
5.10 – Elaborar um programa que troque o estado do LED se o botão continuar sendo pressionado. Utilize uma frequência que torne agradável o piscar do LED.

5.11 – Elaborar um programa para aumentar a frequência em que um LED liga e desliga, enquanto um botão estiver sendo pressionado, até o momento em que o LED ficará continuamente ligado. Quando o botão é solto o LED deve ser desligado.

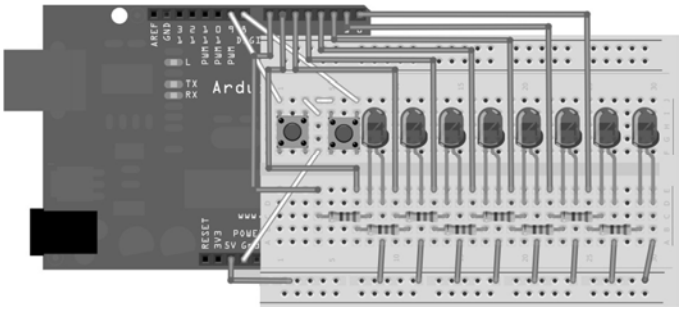
Qual a aplicação prática dessa técnica, imaginando que o botão pode ser o sinal proveniente de algum sensor e o LED algum dispositivo sinalizador?

5.12 – No exercício 5.8, foram propostas 7 animações com 8 LEDs. Crie outra animação, totalizando 8. Depois empregue dois botões: um será o AJUSTE, que quando pressionado permitirá que o outro botão (SELEÇÃO) selecione a função desejada, ver a fig. 5.11. Cada vez que o botão SELEÇÃO for pressionado, um dos oito LEDs deverá acender para indicar a função escolhida; exemplo: 00000100 => LED 3 ligado, função 3 selecionada. Quando o botão de AJUSTE for solto, o sistema começa a funcionar conforme a função escolhida.

Desenvolva a programação por partes, unindo-as e testando com cuidado. Não esqueça: um bom programa é bem comentado e organizado!



a)



b)

Fig. 5.11 – Sequencial de LEDs: a) esquemático e b) montagem no Arduino.

5.13 – Elaborar um programa para apresentar em um *display* de 7 segmentos um número aleatório[2] entre 1 e 6 quando um botão for pressionado, ou seja, crie um dado eletrônico. Empregue o mesmo circuito da fig. 5.14.

5.14 – Elaborar um programa para apresentar nos LEDs da fig. 5.15 um número aleatório entre 1 e 6, formando os números de um dado (mesma lógica do exercício acima).

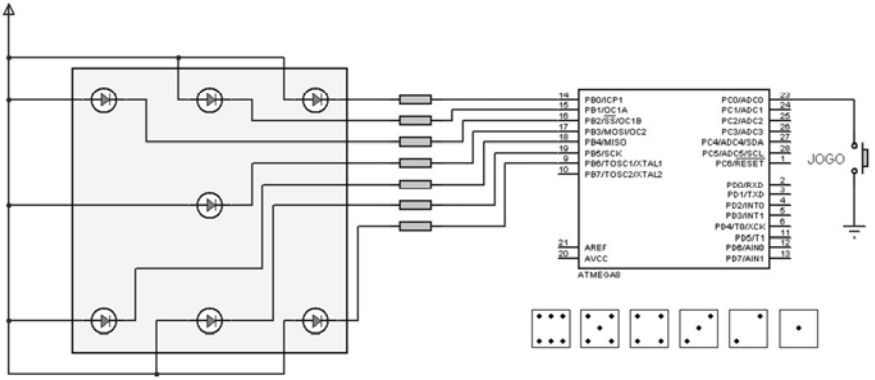


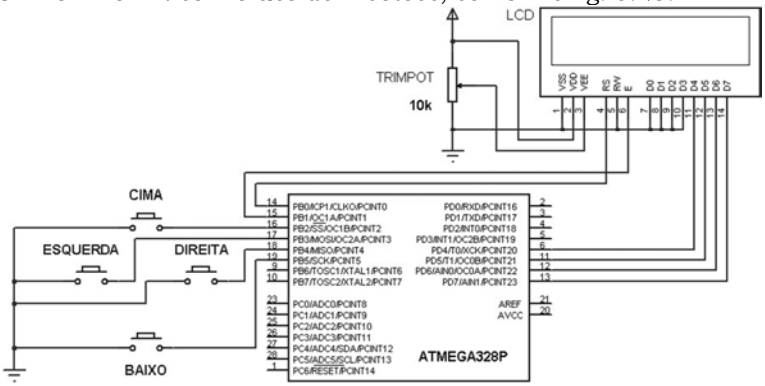
Fig. 5.15 – Dado eletrônico com LEDs. Obs.: a pinagem do ATmega8 é igual a do ATmega328.

[2] Na verdade, criar um número puramente aleatório é difícil, o mais fácil é um pseudoaleatório. Neste exercício, o objetivo é não empregar as bibliotecas padrão do C. A ideia é utilizar o botão para gerar o evento de sorteio do número. Dessa forma, um contador pode ficar contando continuamente de 1 até 6 e, quando o botão for pressionado, um número da contagem será selecionado.

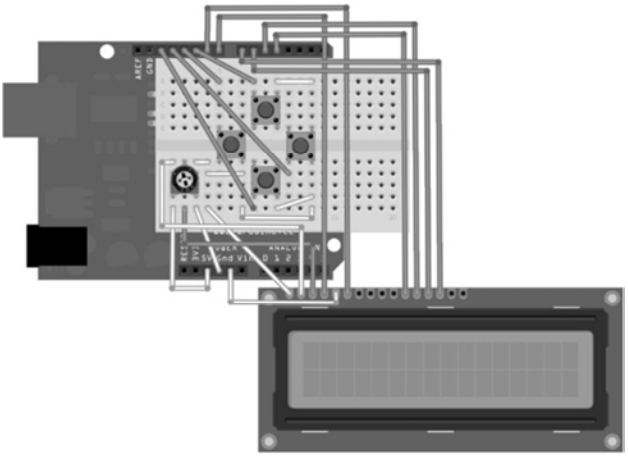
5.15 – Elaborar um programa para deslocar um caractere ‘ * ’ (asterisco) no LCD da fig. 5.18, da esquerda para a direita, ao chegar ao final da linha o caractere deve retornar (vai e vem).

5.16 – Repetir o exercício 5.15 empregando as duas linhas do LCD. Ao chegar ao final da linha superior, o asterisco começa na linha inferior (endereço 0xD3). Dessa forma, na linha superior o asterisco se desloca da esquerda para a direita e na linha inferior, da direita para a esquerda.

5.17 – Elaborar um programa para realizar o movimento de um cursor num LCD 16 × 2 com o uso de 4 botões, conforme fig. 5.23.



a)



b)

Fig. 5.23 – Exercício 5.17: a) esquemático e b) montagem no Arduino.

5.18 – Desenvolva um programa para realizar uma animação sequencial na linha superior e inferior de um LCD 16×2. Escreva um caractere por vez.

5.19 – Crie um caça-níquel eletrônico empregando 3 caracteres diferentes apresentados em 3 posições do LCD. Utilize um botão no pino PB2 do ATmega para o sorteio.

5.20 – Crie oito caracteres novos para o LCD 16×2. Comece a criar seu próprio conjunto de funções.

5.21 – Usando as duas linhas do LCD, crie um cronômetro com passo de 1 s. Utilize os números grandes (4 caracteres por digito) e o pino PB2 do ATmega para o início/parada. Essa ideia é exemplificada na fig. 5.30.

Fig. 5.30 – Números grandes para um cronômetro.

5.22 – Crie um programa que conte até 10.000 apresentando a contagem decimal em um LCD 16×2. Utilize um tempo de 100 ms para o incremento dos números, os dígitos devem ficar à direita do LCD e o número zero na frente do digito mais significativo não deve ser apresentado.

5.23 – Repita o programa acima, só que desta vez, para apresentar um número hexadecimal em uma contagem até 0x3FF.

5.24 – Conte o número de vezes que um botão foi pressionado e apresente o valor em um LCD 16 × 2. O botão pode ser conectado ao pino PB2 do ATmega e o circuito do LCD pode ser o mesmo da fig. 5.23a.

6. INTERRUPÇÕES

6.1 – Supondo que sejam habilitadas 3 interrupções: INT0, INT1 e PCINT0, e que INT0 e PCINT0 sempre estão ativas ao termino da interrupção INT1, quando a interrupção PCINT0 será executada?

6.2 – Teste o exemplo da interrupção INT1 interrompendo a INT0 (seção 6.2.1). Por que se consegue ver que o LED2 do laço principal está piscando quando um dos botões é mantido pressionado? Por que a interrupção INT0 não consegue interromper a INT1?
Exclua o trecho de código dentro da interrupção INT0 que salva o SREG e o restaura. O que acontece?

6.3 – Faça um programa para testar a interrupção PCINT2, usando dois botões nos pinos PD6 e PD7, onde cada um deve ligar um LED nos pinos PD0 e PD1.

7. GRAVANDO A EEPROM

7.1 – Faça um programa para gravar 0xAA em toda a memória EEPROM do ATmega328. Certifique-se que os dados foram corretamente gravados, lendo-os após a gravação. Pode ser empregado um *display* para informar sobre o sucesso da operação ou um LED. Depois disso, com base no manual do ATmega328, empregue a interrupção da EEPROM para repetir o processo.

7.2 – Supondo que exista um circuito integrado ligado ao ATmega328 e que a primeira vez que o sistema é energizado esse circuito necessite uma inicialização específica. O programa do microcontrolador deve interagir com o usuário e perguntar se a inicialização do circuito integrado deve ser feita; caso afirmativo, essa ação será feita somente uma vez e, na próxima vez que o microcontrolador for inicializado, o programa não fará mais nenhuma pergunta.

Faça um programa que utilize um byte da EEPROM para armazenar a informação de que a inicialização do circuito integrado foi feita. Empregue dois LEDs e um botão. A primeira vez, um dos LEDs acende, indicando que a inicialização necessita ser feita; quando o botão for pressionado, o segundo LED deve ser ligado. Ao se reenergizar o circuito, somente o segundo LED deve ficar ligado, indicando que a inicialização já foi feita e o processo de inicialização não é mais necessário.

8. TECLADO MATRICIAL

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.

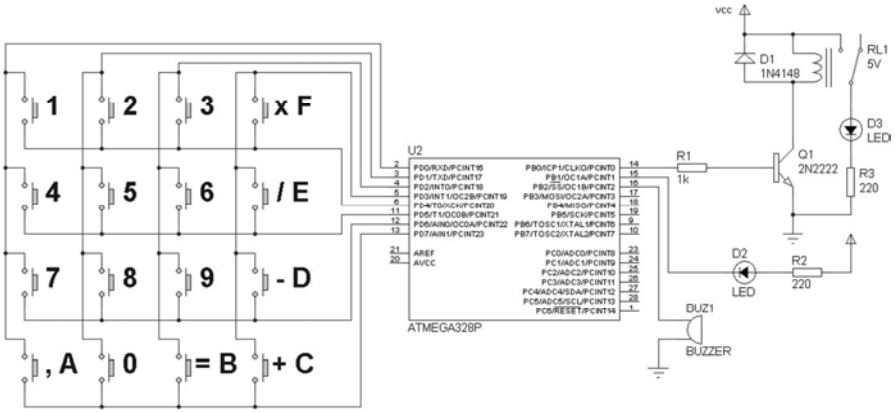


Fig. 8.4 – Controle de acesso por senha numérica.

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.

9. TEMPORIZADORES/CONTADORES

9.1 – Considerando o TC0 trabalhando a 8 MHz com um *prescaler* de 256, quanto tempo leva para o TC0 gerar um estouro de contagem?

9.2 – Considerando-se o PWM rápido configurado para que o valor máximo de contagem do TC0 seja dado pelo valor de OCR0A e que o pino OC0B esteja configurado para gerar o sinal PWM, determine:

- Qual o período do sinal PWM se o ATmega328 estiver rodando a 12 MHz e o valor de OCR0A for 99?
- Supondo que OCR0A é 200, qual deve ser o valor de OCR0B para que o ciclo ativo do sinal PWM (OC0B) seja de 75 %?

9.3 – Verifique a configuração dada ao TC0 para os exemplos acima. Analise os bits dos registradores envolvidos.

9.4 – Com o emprego de um osciloscópio, teste os sinais gerados pelos códigos exemplo dados previamente, gravando-os no Arduino. Altere os valores dos registradores OCR0A e OCR0B para analisar o comportamento do microcontrolador. Para a análise, também pode ser empregado o software de simulação Proteus (ISIS).

9.5 – Elaborar um programa para ler 6 teclas utilizando a interrupção externa INT0 do ATmega328 (ver o capítulo 6). O programa principal não deve ficar responsável pela varredura do teclado (ver o capítulo 8); a interrupção do TC0 deve ser empregada para esse fim. O tratamento do teclado será transparente ao programa principal, devendo ser feito nas interrupções. Assim, quando uma tecla for pressionada será requisitado um pedido de interrupção e o LED correspondente deve ser ligado. O circuito abaixo ilustra o circuito necessário.

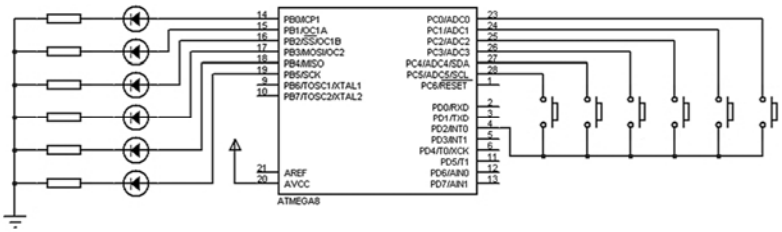


Fig. 9.7 – Seis teclas gerando um pedido de interrupção com o emprego de apenas uma interrupção externa (obs.: a pinagem do ATmega8 é igual a do ATmega328).

9.6 – Qual o valor do divisor de frequência para que o TC2, trabalhando com um cristal externo de 32,768 kHz, produza um estouro a cada 1/4 de segundo?

9.7 – O uso de um cristal 32,768 kHz em relógios digitais tem um motivo óbvio, qual?

9.8 – Elaborar um programa para que o hardware da fig. 9.8 funcione como um relógio 24 h. O ajuste do horário deve ser feito nos botões específicos.

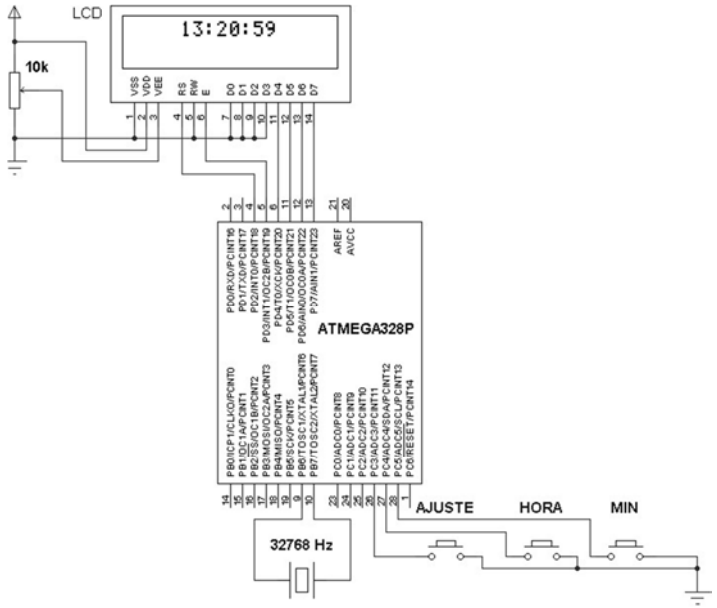


Fig. 9.8 – Relógio com contagem baseado em cristal externo.

9.9 – Considerando o ATmega328 rodando a 16MHz e o TC1 operando no modo normal (65536 contagens), quanto tempo levará para ocorrer um estouro do TC1 para cada *prescaler* possível?

9.10 – Elaborar um programa para que o ATmega328 funcione como frequencímetro digital, apresentando o valor lido em um LCD 16×2. Empregue um circuito adequado (sugestão na fig. 9.13).

Sugestão: utilize um TC de 8 bits para gerar uma base de tempo de 1 s e o TC1, de 16 bits, para contar os eventos externos. A frequência será dada pelo número de contagem do TC de 16 bits (se houver estouro, o programa deve levar esse fato em conta no cômputo da frequência).

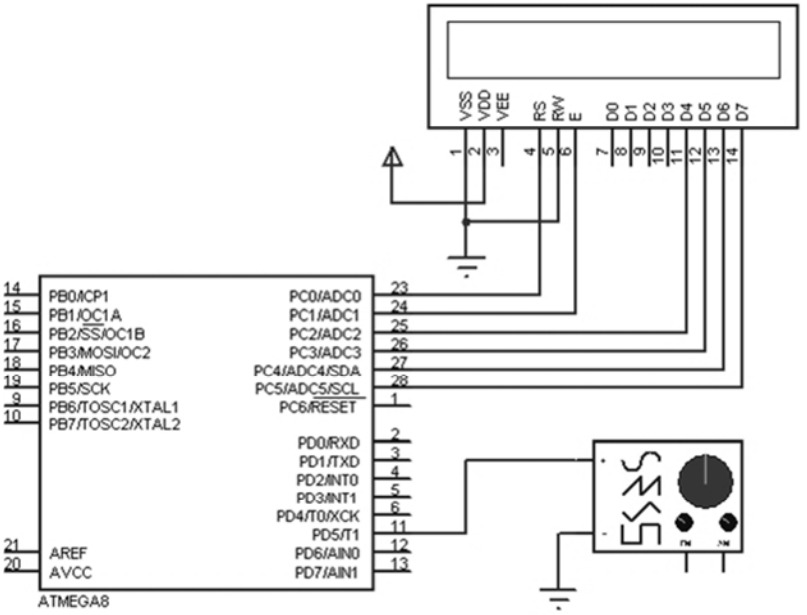


Fig. 9.13 – Circuito para o exercício 9.10 (obs.: a pinagem do ATmega8 é igual a do ATmega328).

9.11 – Faça um programa para gerar 4 sinais PWM com resolução de 100, 400, 1000 e 5000 pontos. Escolha a forma de trabalho do TC empregado.

Qual é a frequência dos sinais gerados se for empregado uma frequência para o trabalho da CPU de 20 MHz?

9.12 – Utilizando o circuito da fig. 9.16, elaborar um programa para controlar, através de um sinal PWM, um motor DC com 256 níveis de velocidade. O nível da velocidade selecionado deve ser apresentado no *display* (valor hexadecimal) e armazenado na memória EEPROM para a inicialização do motor. O display de 7 segmentos deve apresentar um número entre 00 e FF.

Obs.: é aconselhado que a alimentação do motor seja independente do circuito de controle, pois podem haver picos de corrente prejudiciais. No caso do microcontrolador, ele pode ser reinicializado. Tudo vai depender das características da fonte de alimentação

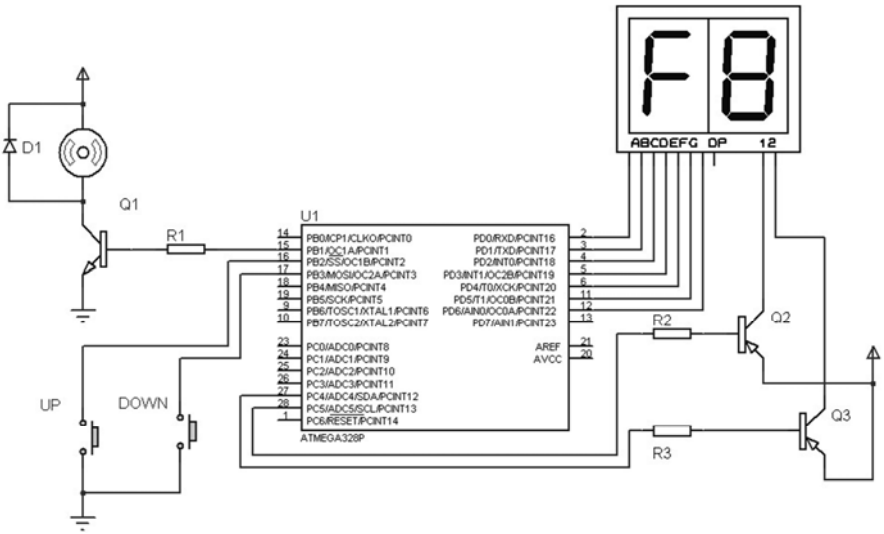


Fig. 9.16 – Controle de um motor DC com um sinal PWM.

9.13– Existem circuitos integrados específicos para o controle do sentido de rotação de motores DC, tal como o L298. Consulte o manual do fabricante para entender o seu funcionamento. Quais seriam os pinos do L298 onde um sinal PWM poderia ser empregado?

Para o Arduino, existem módulos prontos para o trabalho com motores DC, como o apresentado na fig. 9.17, o qual emprega o L298 e o ULN2803.

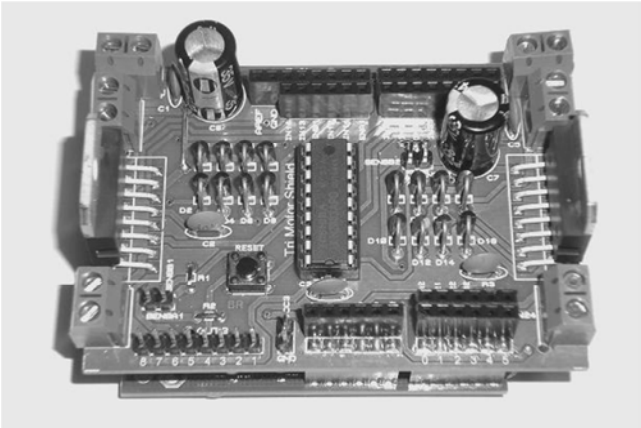


Fig. 9.17 – Módulo para controle de motores para o Arduino[3].

9.14 – Elaborar um programa para controlar o ângulo de giro de um micro servo motor de acordo com dois botões, fig. 9.20. Começando com um pulso de 0,5 ms até 2,5 ms, com passo de incremento de 0,1 ms. Repita o procedimento para um passo de 0,05 ms. Qual a relação angular de giro com a largura de pulso?

Obs.: é aconselhado que a alimentação do motor seja independente do circuito de controle, pois podem haver picos de corrente prejudiciais, no caso do microcontrolador, ele pode ser inicializado. Tudo vai depender das características da fonte de alimentação.

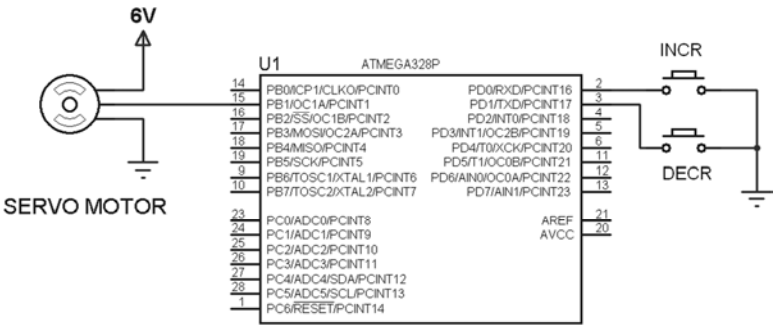


Fig. 9.20 – Circuito para o acionamento de um servo motor.

[3] Circuito esquemático, PCB e demais detalhes podem ser encontrados em www.borgescorporation.blogspot.com.br

9.15 – Elaborar um programa para controlar o motor de passo unipolar da fig. 9.25. Dois botões controlam a direção de rotação e outros dois, a velocidade. Primeiro, deve-se consultar o manual do fabricante do ULN2803 para entender suas características e funcionamento.

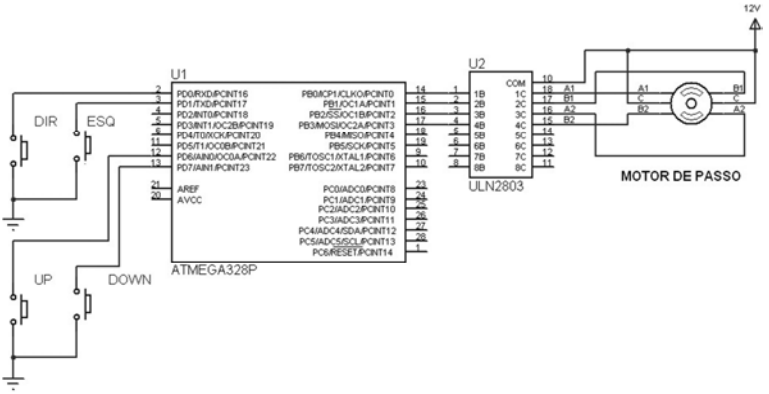


Fig. 9.25 – Controle de um motor de passo unipolar.

9.16 – Como funciona um motor de passos bipolar e como deve ser o seu circuito de controle?

9.17 – Qual a resolução temporal empregada para o TC1 do programa de trabalho com o módulo sonar HC-SR04? Qual sua relação com a resolução do módulo?
Por que o limite para a apresentação da distância foi de 431 cm?

10. GERANDO MÚSICAS COM O MICROCONTROLADOR

10.1 – Utilizando o modo de geração de frequência do ATmega328, faça um programa para gerar as notas musicais básicas de um teclado com 1 oitava (pode ser a 4ª). Empregue um circuito adequado, incluindo as teclas. Na fig. 10.3, é ilustrada a organização das notas musicais no teclado (para aumentar o teclado basta colocar outra oitava em série com a primeira). Obs.: só pode ser tocada uma nota por vez.

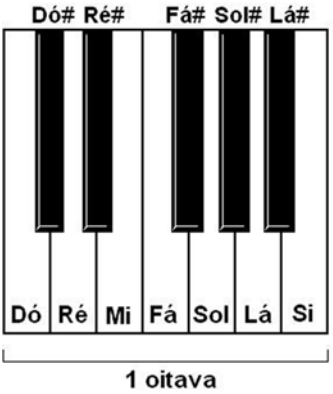


Fig. 10.3 – Organização de uma oitava em um teclado musical.

10.2 – Com base no programa apresentado neste capítulo, faça um programa para aumentar e diminuir a velocidade de uma pequena música (batida) no ATmega328, toda vez que um botão for pressionado. Gere uma função leitora de arquivos RTTTL e utilize uma opção para a seleção de algumas músicas. Pesquise por músicas no formato RTTTL na internet.

10.3 – Otimize o programa de leitura de um arquivo RTTTL para gerar a base de tempo para as notas musicais utilizando o TC2.

11. TÉCNICAS DE MULTIPLEXAÇÃO

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.

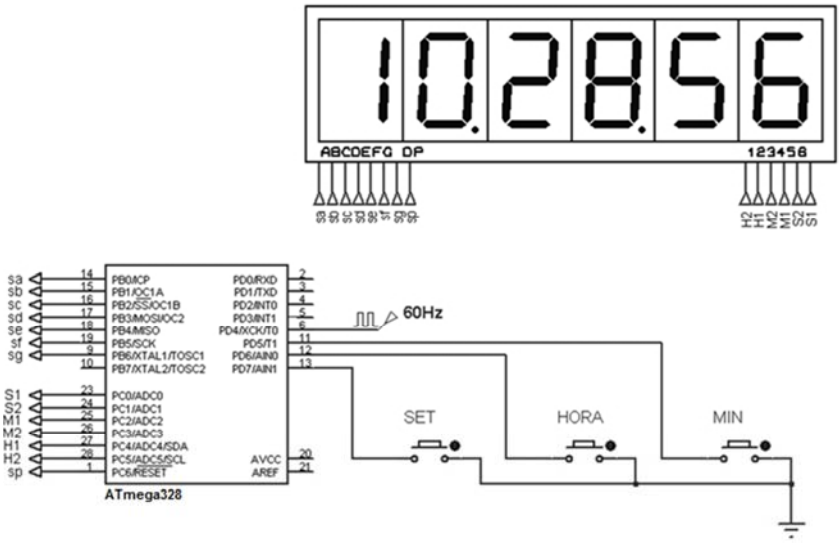


Fig. 11.10 – Relógio com contagem baseado na frequência da rede elétrica (circuito simplificado).

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

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?

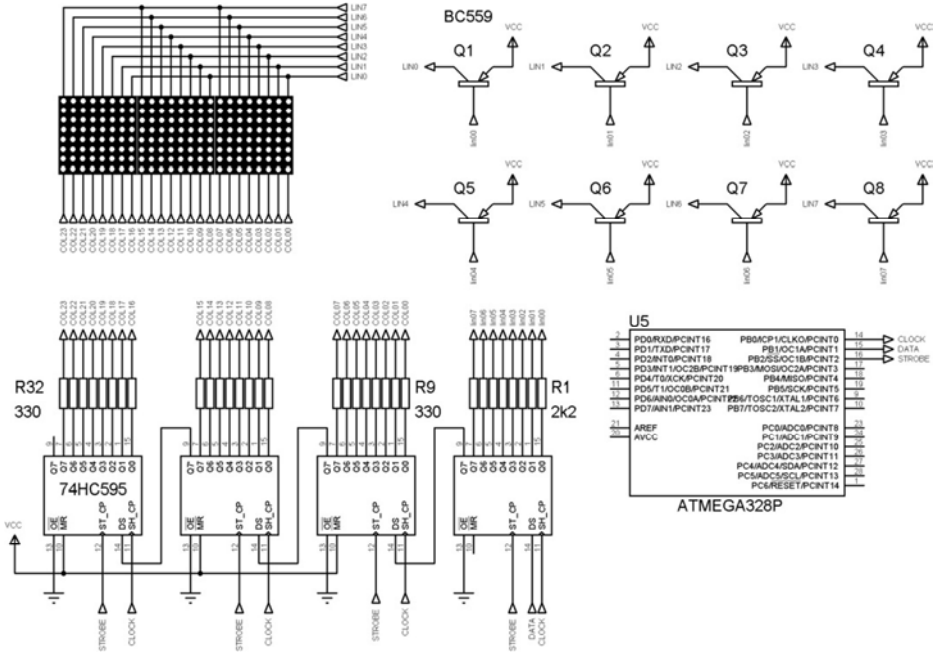


Fig. 11.18 – Matriz de LEDs e o 74HC595.

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 física 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.

12. DISPLAY GRÁFICO (128 x 64 pontos)

12.1 – Elaborar um programa para gerar uma animação gráfica com 5 quadros para o circuito da fig. 12.13.

12.2 – Alterar as funções apresentadas para o LCD 128 × 64 para que seja feita a leitura do seu bit de ocupado (*busy flag*).

12.3 – Alterar as funções apresentadas para a escrita de mensagens com pixels invertidos no LCD 128 × 64, ou seja, escrita clara com fundo escuro.

12.4 – Crie uma função para a leitura dos dados diretamente do LCD 128 × 64, permitindo a alteração de um único pixel se desejado.

13. FORMAS DE ONDA E SINAIS ANALÓGICOS

13.1 – Elaborar um programa para gerar uma onda senoidal de acordo com o circuito da fig. 13.3. A frequência do sinal pode ser alterada por dois botões.

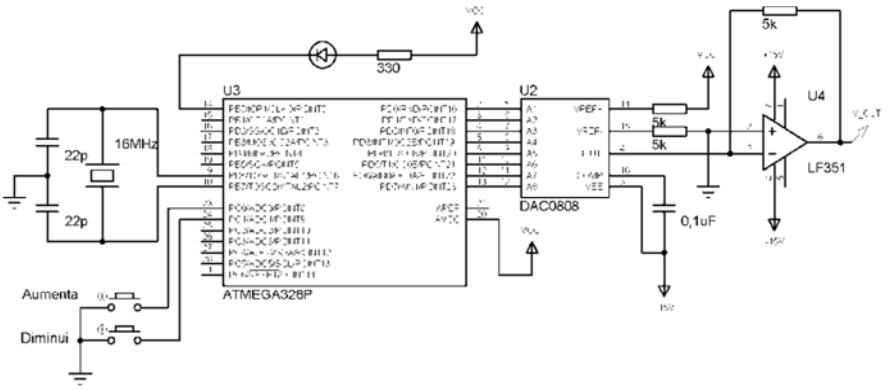


Fig. 13.3 – Circuito para gerar um sinal senoidal.

13.2 – Com base no exercício anterior, como empregar o ATmega para produzir as formas de onda triangular, dente-de-serra e quadrada?

13.3 – Empregando uma rede R/2R, faça um programa para gerar uma rampa de tensão com o ATmega328. O nível da tensão de saída pode ser controlado por dois botões e um *display* de 7 segmentos pode sinalizar o nível de tensão de saída, conforme fig. 13.6.

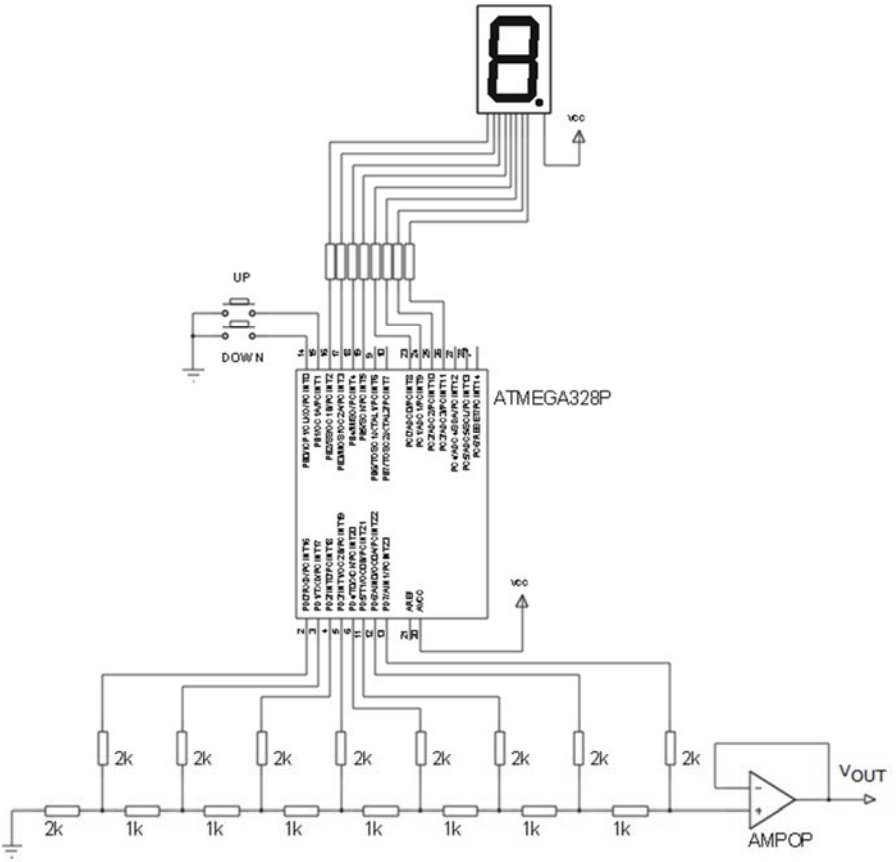


Fig. 13.6 – Circuito para o exercício 13.3.

13.4 – Elaborar um circuito que funcione como um conversor DA para um sinal PWM. Desenvolva um programa para o teste.

13.5 - Baseado no conceito de valor médio, empregue o PWM do ATmega para criar um sinal senoidal com frequência de 10 Hz. Obs.: para uma precisão adequada, a frequência do sinal PWM deve ser no mínimo 10 vezes maior que a frequência do sinal gerado.

Como alterar a frequência do sinal gerado?

13.6 – Refaça os cálculos do exemplo anterior com um *prescaler* igual a 1 para o TC0.

13.7 – Consulte referências bibliográficas adequadas para a revisão e melhor compreensão do equacionamento para um conversor Buck.

13.8 – Quais os tipos de conversores CC-CC utilizados para elevar a tensão?

13.9 – Como seria um circuito microcontrolado para o controle da tensão de saída de um conversor Buck? Considerando a leitura dessa tensão com o AD do microcontrolador, quais os tipos de controle que poderiam ser feitos?

13.10 – Como funciona a modulação PWM senoidal mais simples que pode ser empregada em um inversor?

13.11 – Como funciona a modulação PWM senoidal a 3 níveis?

13.12 – Com funcionam e para que servem os IGBTs?

13.13 – Pesquise um circuito inversor completo para um inversor monofásico.

14. SPI

14.1 – Altere o programa para o TC72 (pág. 329), apresentando a temperatura com uma resolução de 0,25 ºC (fig. 14.7).

14.2 – Elaborar um programa para a leitura do sensor de temperatura MAX6675 do circuito da fig. 14.11.

O MAX6675 digitaliza o sinal proveniente de um termopar tipo K[4] e sua saída de dados é de 12 bits com compatibilidade SPI. Sua resolução é de 0,25°C, com medição de temperatura entre 0 e 1024°C. Seu protocolo de comunicação é apresentado na fig. 14.12. Nela se observa que os bits mais significativos do dado são enviados primeiro. O formato do dado é apresentado na tab. 14.7.

[4] O termopar tipo K é um sensor de temperatura composto por Chromel (90% de Níquel e 10% de Cromo) e Alumel (95% de Níquel, 2% de Manganês, 2% de Alumínio e 1% de Si1ício). Os dois materiais são conectados e na sua junção é gerada uma pequena diferença de potencial dependendo da temperatura a que são submetidos. Fonte http://pt.wikipedia.org/wiki/Termopar.

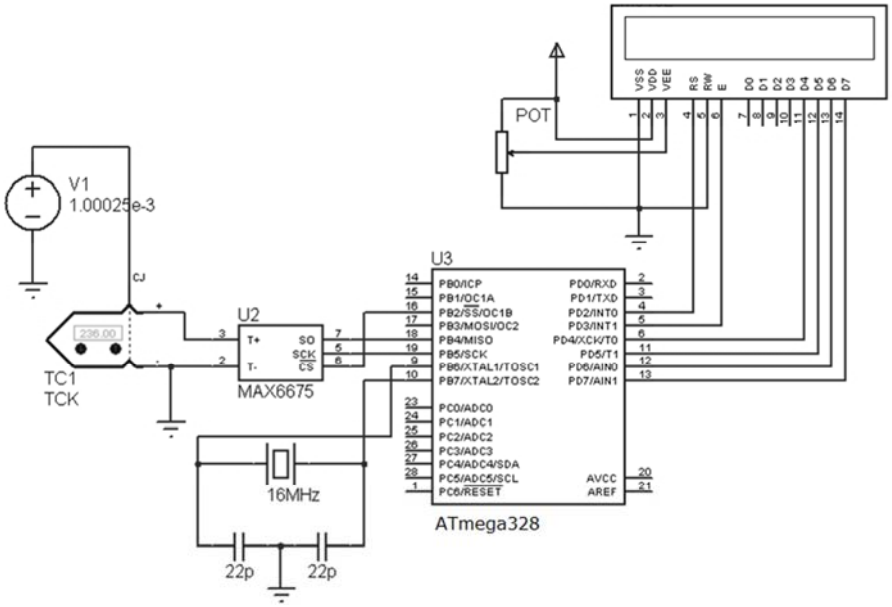


Fig. 14.11 – Usando o sensor MAX6675 (a fonte V1 é para a compensação da tensão gerada pelo termopar, utilizada para a simulação).

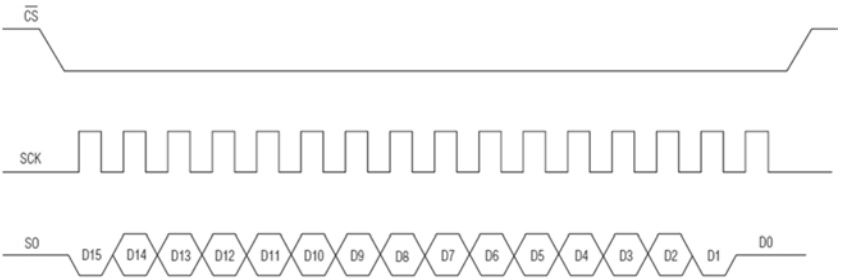


Fig. 14.12 – Protocolo de comunicação do MAX6675.

Tab. 14.7 - Formato de dados enviado pelo MAX6675.

	Bit de Sinal	Leitura de 12 bits da temperatura												Entrada do termopar	ID do dispositivo	Estado
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	MSB											LSB		0	*Tri-State*

Para o emprego da SPI do ATmega, basta escrever um dado qualquer no registrador de dados da SPI (SPDR). O controle do pino $\overline{CS}$ deve ser feito por software e os dados serão recebidos no pino MISO. O trecho de programa abaixo ilustra o processo.

```
//-------------------------------------------------------------------------------
//MAX6675
     ...                  //configurações para uso da SPI
     clr_bit(PORTB,SS);   //pino SS em zero
     temp_MSB =le_MAX();  //lê o byte + significativo.(temp_MSB é um unsigned char)
     temp_LSB =le_MAX();  //lê o byte - significativo.(temp_LSB é um unsigned char)
     set_bit(PORTB,SS);   //pino SS em 1

     valor_int = (temp_LSB>>3) | (temp_MSB<<5);         /*são 12 bits de dados, os
                                                          demais não são analisados*/
     valor_int = valor_int/4;  /*divide o valor por 4 para uma resolução de 1 grau,
                                 ou valor_int=(temp_LSB>>5)|(temp_MSB<<3) – já divide.
                                                    valor_int é um unsigned int*/
     ...                  //aqui vai a rotina para mostrar o valor no display
//-------------------------------------------------------------------------------
unsigned char  le_MAX()
{
     SPDR = 0x00;                    //envia um dado qualquer para recebe um byte
     while(!(SPSR & (1<<SPIF)));     //aguarda o envio do dado
     return SPDR;                    //retorna o valor recebido
}
//-------------------------------------------------------------------------------
```

14.3 – Como o MAX6675 poderia ser utilizado para o projeto de um forno para solda de componentes SMD?

14.4 – Pesquise o sistema de arquivos FAT16/FAT32 e desenvolva um programa para escrever e ler dados nesses formatos em cartões SD/MMC.

15. USART

15.1 – Elaborar um programa para realizar uma comunicação serial (RS232) entre o AVR e um computador, conforme exemplo da fig. 15.7. O programa deve escrever "TESTE SERIAL" na primeira linha do LCD e mandar uma mensagem ao terminal de recepção. Os caracteres digitados nesse terminal devem ser escritos na segunda linha do LCD. Quando a escrita estiver completa, deve ser apagada e reiniciada. Devem ser empregados 2400 bps, 8 bits de dados, sem paridade e 1 bit de parada.

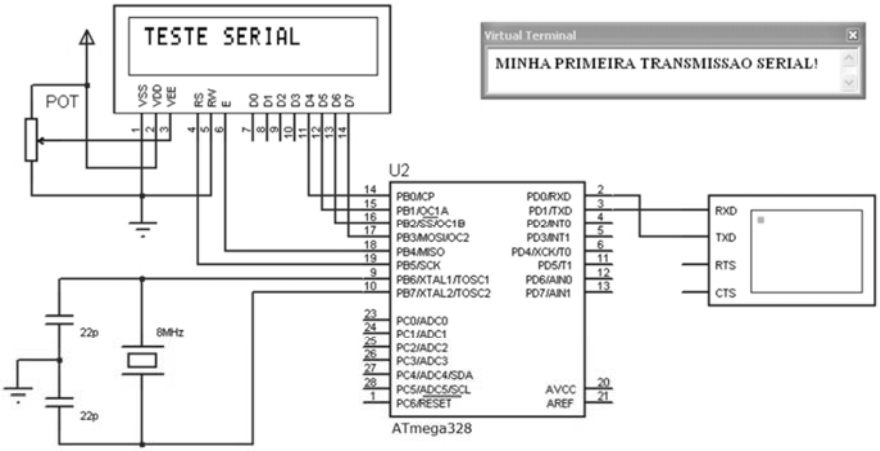


Fig. 15.7 – Comunicação serial: possível simulação no Proteus.

15.2 – Faça um programa para controlar, através do computador, 4 LEDs ligados aos pinos PB0:4 do ATmega328. Imprima na tela do hiperterminal uma mensagem semelhante a da fig. 15.8, crie sua própria imagem.

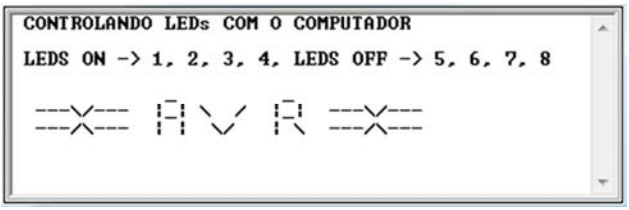


Fig. 15.8 – Mensagem recebida no computador.

A ideia é: quando o número 1 for pressionado, o LED ligado ao pino PB0 é acionado, quando o número for 5 ele será desligado (comando similares para os demais LEDs).

Para gerar a mensagem da fig. 15.8 é necessário escrever cada linha individualmente na programação, tal como:

```
const char msg1 [] PROGMEM = " CONTROLANDO LEDs COM O COMPUTADOR\0";
const char msg2 [] PROGMEM = " LEDS ON -> 1, 2, 3, 4, LEDS OFF -> 5, 6, 7, 8\0";
const char msg3 [] PROGMEM = "                 _           _ \0";
const char msg4 [] PROGMEM = "  ---\\/---  |_| \\  /  |_|  ---\\/---\0";
const char msg5 [] PROGMEM = "  ---/\\---  | |  \\/   | \\  ---/\\---\0";
```

Obs.: caso seja feita uma simulação no Proteus, para o deslocamento do cursor para uma nova linha é necessário enviar o valor 0x0D para o seu terminal virtual.

*A programação exigida por este exercício é muito utilizada para a automação de sistemas, onde é necessário enviar e receber dados ou comandos de um microcontrolador.

15.3 – Altere o programa exemplo apresentado anteriormente de tal forma que ele realize a comunicação com o computador empregando as interrupções da USART.

15.4 – Faça um programa para ligar 5 LEDs em um circuito com um módulo RF RX de acordo com os botões ligados a um módulo RF TX, conforme apresentado na fig. 15.12. Isto é, faça um controle remoto onde cada botão aciona um único LED.

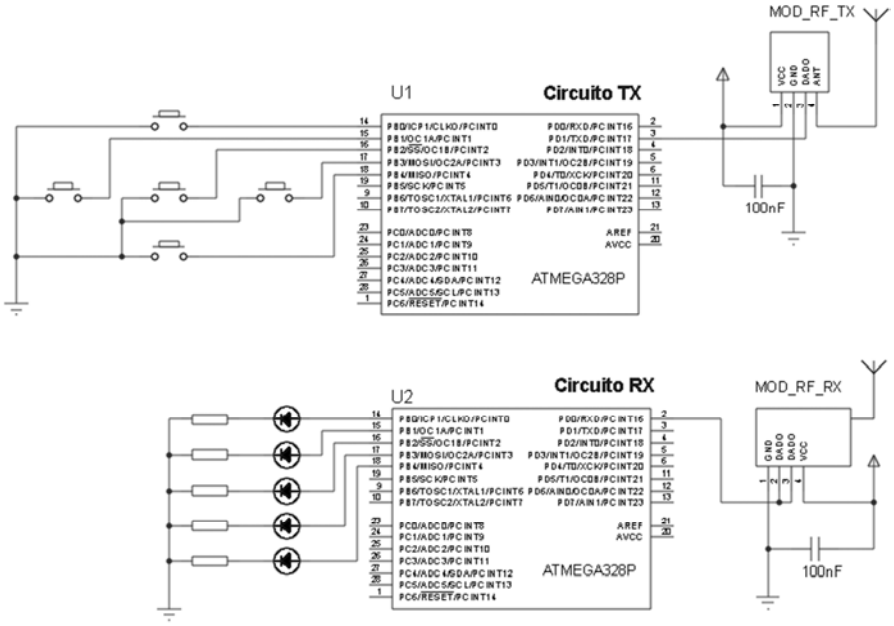


Fig. 15.12 – Controle remoto por RF.

Obs.: os capacitores de desacoplamento de 100 nF são importantes para o correto funcionamento do circuito.

15.5 – Pesquise as aplicações realizadas com *Smart Phones* e *tablets* que utilizam bluetooth para interagir com sistemas microcontrolados. Por exemplo, um osciloscópio enviando dados via bluetooth para um celular.

15.6 – Pesquise sobre a tecnologia ZigBee. Como a USART poderia ser empregada para o trabalho com essa tecnologia?

16. TWI (TWO WIRE SERIAL INTERFACE) – I2C

16.1 – Elaborar um programa para ler e escrever em um DS1307 (*Real Time Clock*) empregando o módulo TWI do ATmega328, conforme circuito da fig. 16.16. Altere o programa apresentado para que ele não utilize a interrupção do TWI e faça a atualização do LCD a cada 1 segundo utilizando o sinal proveniente do pino SOUT do DS1307.

16.2 – Baseado no programa apresentado anteriormente, altere as rotinas de trabalho com o DS1307 para a leitura e escrita sequencial de vários bytes.

17. COMUNICAÇÃO 1 FIO POR SOFTWARE

17.1 – Consulte o manual do DS18S20 para saber como o CRC é gerado e como deve ser calculado. Utilize essa informação para resolver o exercício 17.2.

17.2 – Elaborar um programa para trabalhar com dois sensores de temperatura DS18S20, conforme fig. 17.6. Antes de colocá-los juntos no barramento 1 fio é necessário ler individualmente o código único de 64 bits de cada sensor. Apresente o valor do CRC de cada DS18S20 em uma linha do LCD 16×2 em conjunto com sua temperatura.

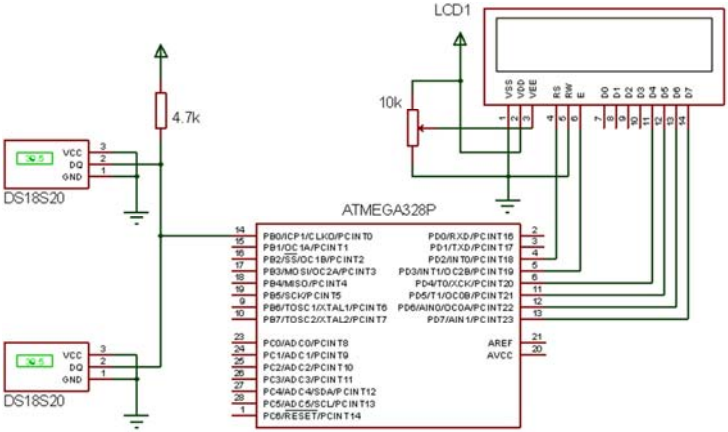


Fig. 17.6 – Circuito com dois DS18S20.

18. COMPARADOR ANALÓGICO

18.1 – Elaborar um programa para executar uma determinada função quando uma tecla for pressionada. Use o comparador analógico para gerar a interrupção para o botão**.**

18.2 – Elaborar um programa para medir uma resistência desconhecida empregando o circuito da fig. 18.3.

18.3 – Como os circuitos apresentados para a medição de resistência e capacitância podem ser utilizados para se medir indutância? Quais as técnicas que podem ser empregadas para aumentar a resolução desses circuitos e como melhorá-los?

19. CONVERSOR ANALÓGICO-DIGITAL - ADC

19.1 – Com base no *application note* AVR122 faça um programa para ler com a maior precisão possível o sensor de temperatura interno do ATmega328. Utilize uma média de leituras para apresentar o resultado.

19.2 – Determine valores para os resistores da fig. 19.5. Depois repita o cálculo para um teclado matricial de 12 e 9 teclas.

19.3 – Empregando o ADC do ATmega328, elaborar um circuito e o respectivo programa para medir com precisão uma resistência desconhecida.

19.4 – Faça um programa para a leitura do teclado da fig. 19.6.

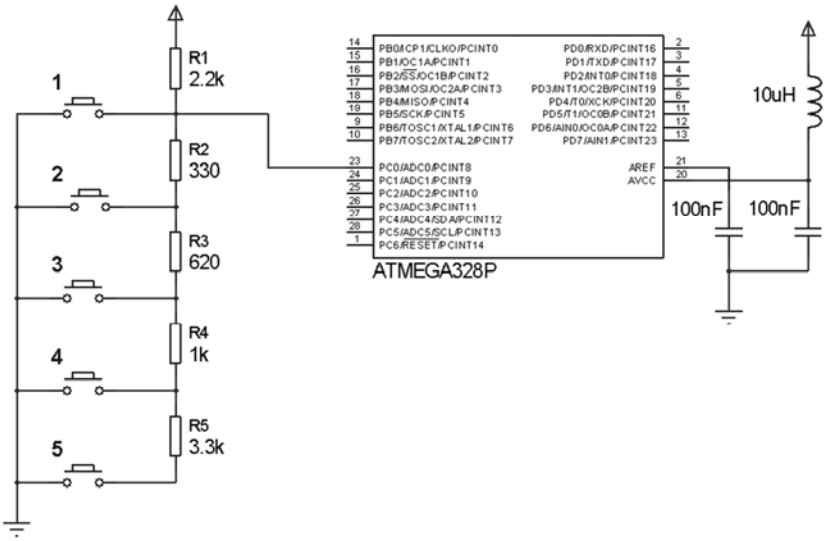


Fig. 19.6 – Leitura de 5 teclas com o ADC do ATmega328.

19.5 – Elaborar um programa para ler os sensores de temperatura LM35 conforme o circuito da fig. 19.8. Escreva a temperatura de cada um deles em uma linha do LCD 16×2.

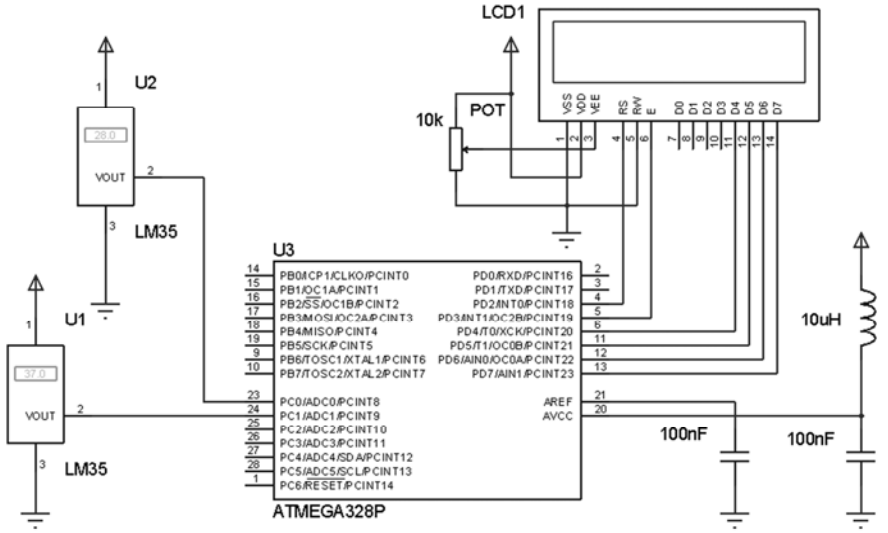


Fig. 19.8 – Empregando o ATmega328 para ler dois sensores de temperatura LM35.

19.6 – Elaborar um programa para ler a tensão proveniente de um potenciômetro com o ADC do ATmega328 (ver a fig. 19.10). Apresente o valor binário no computador (ver o capítulo 15). Depois aumente a resolução do ADC para 11 bits e 12 bits, respectivamente. Verifique os resultados.

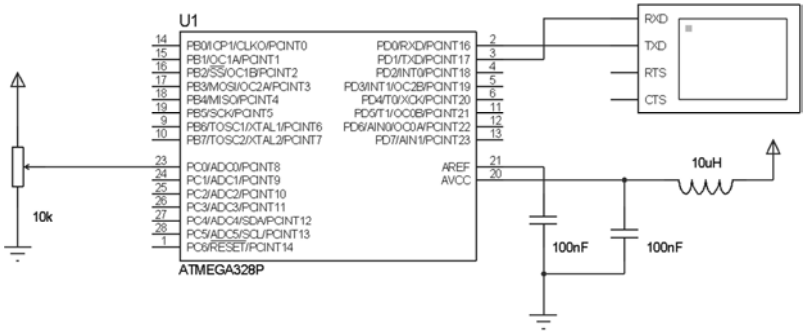


Fig. 19.10 – Lendo a tensão proveniente de um potenciômetro.

Obs.: potenciômetros são comuns em dispositivos de controle de posição, como por exemplo, *joysticks*.

19.7 – Utilize um filtro de média móvel de 5 amostras para o exercício 19.6.

20. MESCLANDO PROGRAMAÇÃO C COM ASSEMBLY

20.1 – Criar um programa em *assembly* que chame uma função em C.

20.2 – Criar um programa em C que chame uma função em *assembly*. Como colocar um trecho de código *assembly* dentro de um código em C?

20.3 – Quais são as técnicas utilizadas para a otimização de um programa em C?

20.4 – Abra um arquivo *.map de um projeto que você já fez e o analise.

21. RTOS

21.1 – Pesquise quais são os RTOSs disponíveis atualmente para os sistemas microcontrolados. Por que não se costuma utilizar um RTOS com microcontroladores de 8 bits?

21.2 – Repita o exercício 11.3 (capítulo 11), desta vez empregando o BRTOS.

21.3 – Por que os sistemas embarcados modernos exigem o uso de um RTOS?

23. FUSÍVEIS E A GRAVAÇÃO DO ATMEGA328

23.1 – Empregando o WDT do ATmega328 faça um programa utilizando o modo de interrupção para piscar um LED a cada 0,5 s.

23.2 – Consulte o manual do ATmega328 e descubra como funciona o *DebugWire*.

D. CIRCUITOS PARA O ACIONAMENTO DE CARGAS

D.1 - Deseja-se acionar o relé do circuito da fig. D.6[5] com um sinal de 5 V. O relé deve operar com uma corrente de 150 mA. Será empregado o transistor NPN BC548 que têm um ganho mínimo de 110. Determinar R_B.

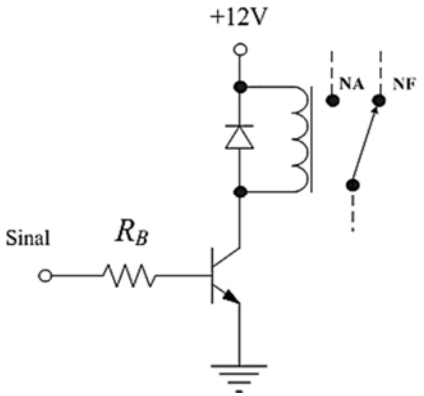


Fig. D.6 – Chave transistorizada para acionamento de um relé.

[5] Deve ser empregado um diodo, denominado diodo de roda livre, paralelo à bobina do relé, para evitar dano ao transistor durante o seu desligamento. Isso é necessário porque no corte da corrente de uma bobina, aparece uma força contra eletromotriz que induz uma tensão na bobina. Essa tensão possui polaridade oposta à de alimentação e sua magnitude pode ser elevada. Desta forma, o diodo de roda livre garante a desmagnetização gradual do indutor (dissipação da energia acumulada), grampeando a tensão induzida em cerca de 0,7 V até que a corrente no indutor se anule.