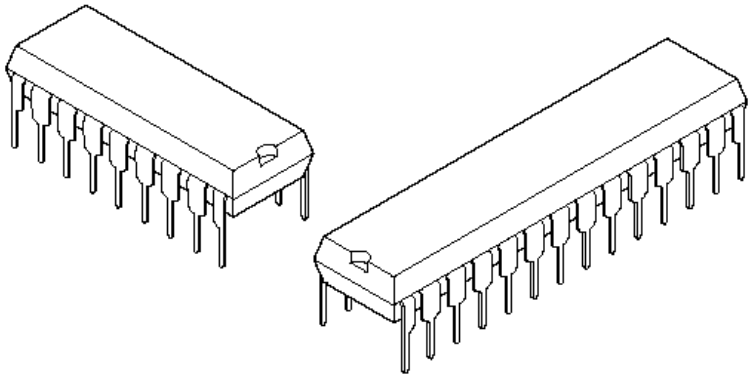
Circuitos de Interface para Microcontroladores



Adaptação livre de parte do documento AXE001_pic_electronics.pdf
da Revolution Education Ltd.

Índice

1. O que é um Microcontrolador PIC ?

Os equipamentos controlados por microprocessadores são normalmente constituídos por vários circuitos integrados ou chips, cada um com a sua função, a saber: o microprocessador (CPU), uma memória EPROM com o programa, uma memória RAM para armazenamento de dados e interfaces de entrada/saída (I/O input/output) para ligação ao exterior. Pelo contrário, os sistemas baseados em microcontroladores possuem um único chip – o microcontrolador. Um microcontrolador PIC é um circuito integrado de pequenas dimensões e que contém num único chip, a CPU, RAM, ROM e circuitos de interface. A grande variedade destes componentes possibilita que o mesmo fabricante ofereça modelos com mais ou menos RAM, com outros dispositivos como portos de comunicação, conversores analógico/digitais, etc.

Os microcontroladores PIC podem ser utilizados como "cérebro" para controlar uma enorme variedade de equipamentos, desde máquinas de lavar a telemóveis ou automóveis, etc. Existem portanto aos milhões por todo o lado, sendo o seu custo muito reduzido. São, por isso, excelentes companheiros para a tarefa a que nos propomos: conceber um robô móvel autónomo, dotado de vários sensores. Para isso as informações provenientes desses sensores, de toque, de detecção de obstáculos, de distância, etc, devem ser adaptadas de forma a serem interpretadas pelo nosso PIC. Por outro lado, o controlo de dispositivos como motores ou relés também exige adaptação de sinal. É desse assunto que iremos tratar nos próximos capítulos.

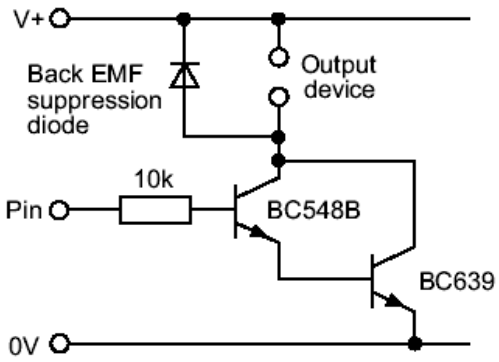
2. Interfaceamento do Microcontrolador

Neste capítulo iremos explicar como realizar as interfaces para que diferentes dispositivos de entrada e de saída possam ser ligados ao microcontrolador.
Iremos tratar dos seguintes dispositivos:

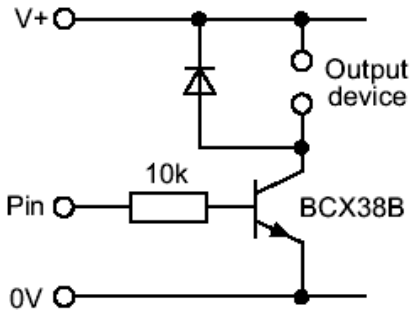
2.1. Circuitos de interface standard

2.1.1. Circuito de interface com transístor

Muitos dos dispositivos de saída irão necessitar de um circuito de comutação por transístor. Para a maioria dos casos um par Darlington formado por dois transístores é ideal.
Contudo, este circuito necessita de dois transístores separados. É possível adquirir um dispositivo contendo os dois transístores num único encapsulamento. Estes transístores designam-se Darlington e possuem elevado ganho. Um transístor como o BCX38B pode accionar correntes até 800 mA. Este será o transístor utilizado em todos os exemplos deste manual.

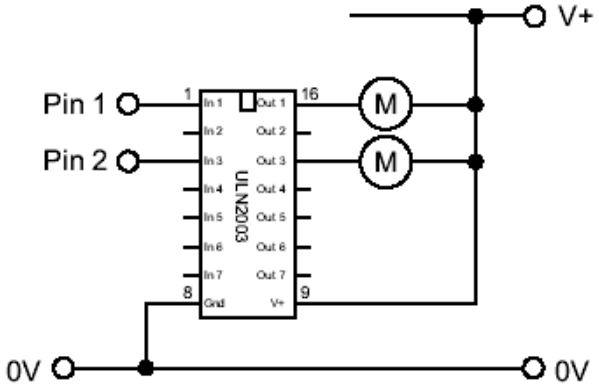


Note que é comum colocar um díodo invertido em paralelo com o dispositivo controlado (díodo de roda-livre). Isto é essencial para cargas indutivas, como são os casos de motores e relés. Sempre que se desliga um destes dispositivos cria-se uma corrente inversa que iria destruir o transístor. Poderá ser usado um vulgar díodo 1N4001.

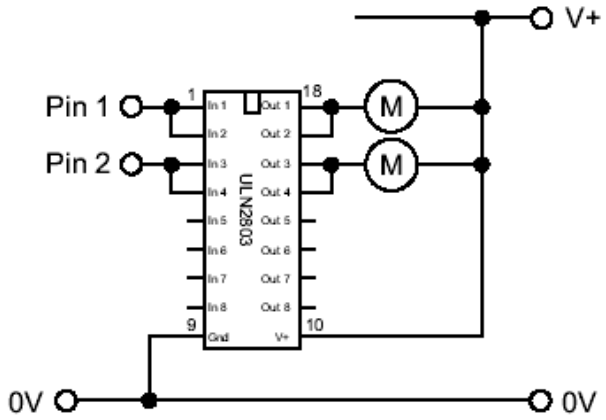


2.1.2. Circuito de interface com transístor Darlington

Se for necessário controlar mais do que um dispositivo, poderá ser aconselhável utilizar um integrado específico, o ULN2003 *Darlington driver IC*, que contém 7 transístores Darlington e ainda os díodos de roda-livre num único invólucro.

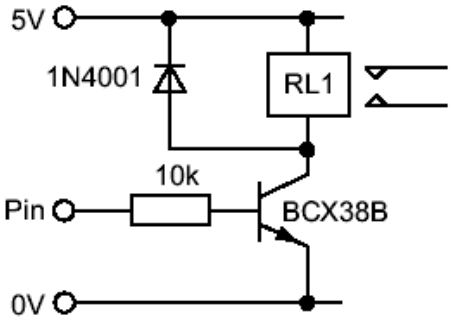


A versão ULN2803 contém 8 transístores.
Se for necessário controlar correntes mais elevadas poderão ser ligadas duas saídas como na figura.



2.1.3. Circuito de interface com relés

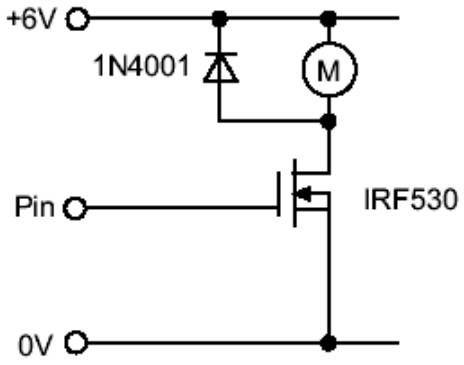
Os relés podem ser utilizados para accionar dispositivos de maior potência como motores e solenóides. Nesse caso, o relé pode ser alimentado por uma fonte de alimentação separada, por exemplo 12 V. Note o uso do díodo de roda-livre em paralelo com a bobina do relé.



2.1.4. Circuito de interface com transistor FET

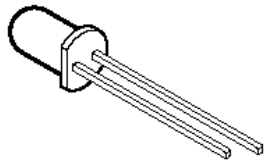
Os transístores de Potência MOSFET podem ser utilizados em vez dos pares Darlington para accionamento de dispositivos de média potência. O circuito a utilizar é o da figura. Repare-se no uso do díodo de roda-livre.

Os MOSFETs apresentam uma resistência interna muito baixa (décimas de ohm), pelo que a queda de tensão no transístor é muito reduzida.

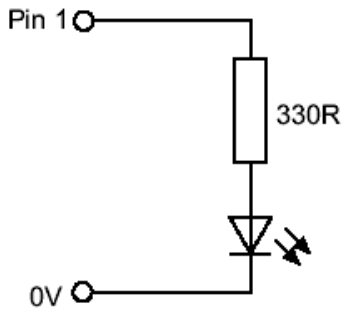


2.2. Interface com Dispositivos de saída

2.2.1. LED

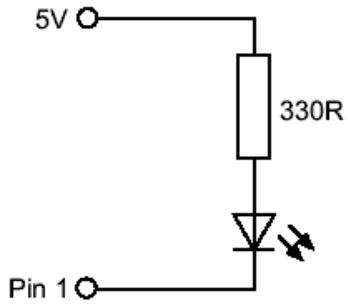


Os pinos do microcontrolador PIC podem deixar passar (*sink*) ou fornecer (*source*) correntes de 20 mA, o que significa que é possível ligar directamente um LED a um pino de saída. Lembre-se de que os pinos de um microcontrolador como o PICAXE vêm já configurados, uns como saídas e outros como entradas. Alguns são reprogramáveis. Para limitar a corrente utiliza-se em série uma resistência de 330 ohms.



LED ligado à massa.

Para ligar o LED (ON) - high 1
Para desligar o LED (OFF) - low 1



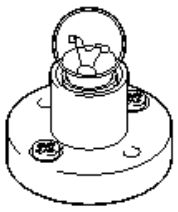
LED ligado à alimentação.

Para ligar o LED (ON) - low 1
Para desligar o LED (OFF) - high 1

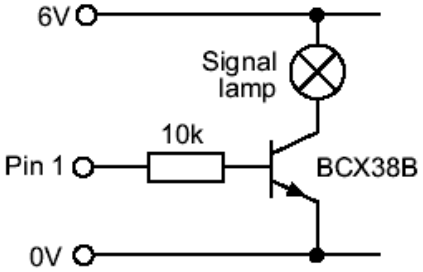
2.2.2. Lâmpada de sinalização

Para interfacear uma lâmpada de sinalização é necessário o circuito com transístor atrás referido. Isso resulta nomeadamente da elevada corrente utilizada pela lâmpada mas também pode ser frequente esta necessitar de uma alimentação diferente da do microcontrolador. Nesse caso é necessário garantir que a massa (0V, GND) é comum às duas alimentações. Importa referir que como o nosso circuito vai ser alimentado por baterias ou pilhas a melhor solução é usar um LED, pois consome menos corrente.

Para ligar a lâmpada - high 1
Para desligar a lâmpada - low 1



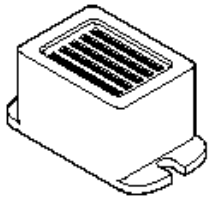
signal lamp



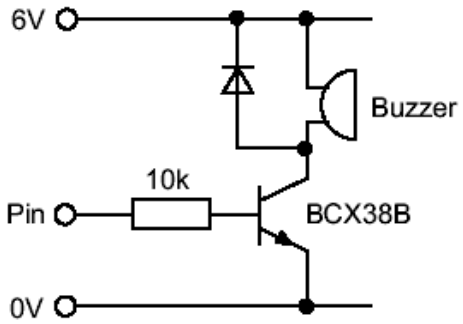
2.2.3. Bezouro ou Buzzer

Para interfacear um bezouro (buzzer) é necessário utilizar o circuito a transístor standard. No caso de se utilizar uma alimentação separada é necessário garantir que a massa (0V, GND) é comum às duas alimentações. No caso de se utilizarem baterias ou pilhas como fonte de alimentação, é de lembrar que os altifalantes piezo-eléctricos consomem menos corrente. Além disso os bezouros apenas produzem uma frequência, ao contrário dos altifalantes piezo-eléctricos que permitem reproduzir diferentes frequências.

Para ligar o buzzer -high 1
Para desligar o buzzer -low 1

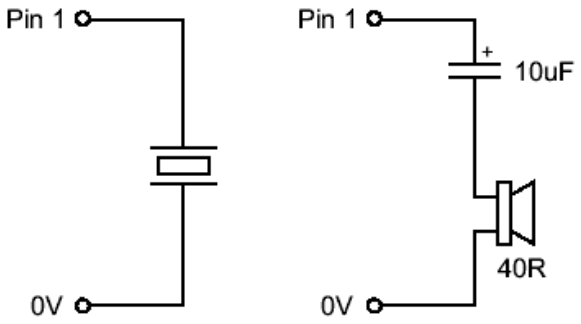


buzzer



2.2.4. Altifalantes piezo-eléctricos

Pode usar-se um altifalante piezo-eléctrico para produzir dieferentes sons. Os bezouros produzem som (ruído) quando são alimentados, enquanto que os altifalantes exigem um sinal pulsado (onda quadrada) para gerar o som. Felizmente isso é fácil de produzir no PICAXE, pois existe uma instrução *sound*.



Para produzir uma nota de tom 100, duração 50 no pino 1

```
sound 1, (100,50)
```

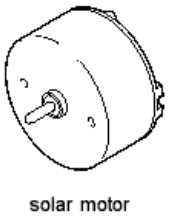
Para produzir um som variado usando a variável b1 -

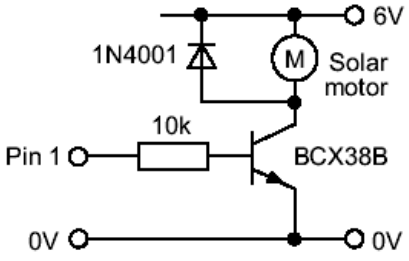
```
for b1 = 1 to 100      ; a instrução sound é repetida tomando os valores de tom
  sound 1, (b1,25)     ; de 1 a 100
next b1
```

2.2.5. Micromotores

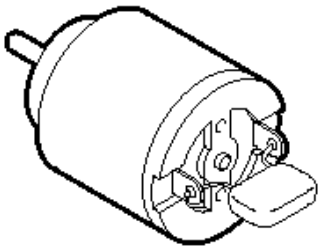
Muitos projectos requerem a utilização de motores de corrente contínua (CC) baratos para criar movimento circular. Existem várias possibilidades de interfacear motores ao microcontrolador.

O circuito da figura utiliza transistores Darlington para accionar o motor. Este circuito funciona bem com motores CC de brinquedos baratos. No entanto importa acautelar o facto de que os motores produzem muito ruído eléctrico nas linhas de alimentação do microcontrolador, podendo levá-lo a deixar de funcionar.

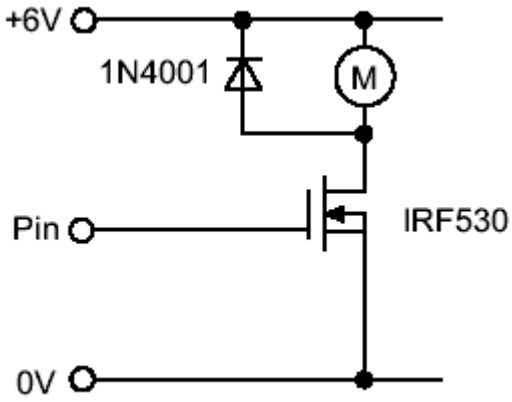

solar motor



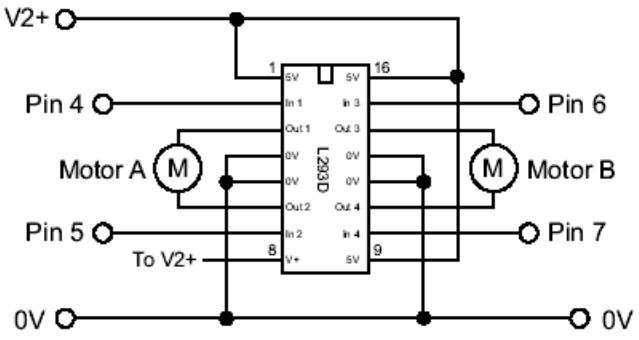
Uma solução para o ruído eléctrico consiste em colocar entre os terminais do motor um condensador de poliéster (não polarizado) de 220nF.



Para accionar motores de maior potência, pode usar-se o circuito a MOSFET anteriormente descrito.

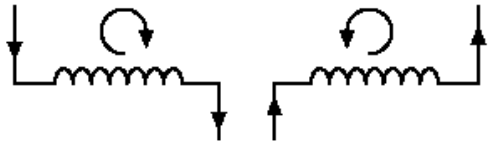


Em várias circunstâncias irá ser necessário accionar mais do que um motor, como é o caso dos robôs móveis. A solução mais simples e barata consiste em utilizar um integrado *motor driver* como o L293D. Este integrado possibilita o controlo de dois motores CC, utilizando quatro pinos de saída do microcontrolador. Naturalmente se tivermos um só motor bastam dois pinos. Esta solução irá ser utilizada nos nossos robôs pois além de permitir o accionamento nos dois sentidos dos dois motores, permite ainda o controlo de velocidade por PWM, como se verá mais à frente.



Ambas as entradas baixas	- o motor pára
Primeira entrada alta, segunda entrada baixa	- motor para frente
Primeira entrada baixa, segunda entrada alta	- motor para trás
Ambas as entradas altas	- motor pára

A mudança dos sinais nos pinos de entrada tem o efeito de produzir a alteração do sentido da corrente no enrolamento do motor, logo do seu sentido de rotação.



Current flow

Note que o L293D aquecerá com uso intenso. O uso de um dissipador colocado no topo do integrado pode ajudar ao arrefecimento. No caso de montagens em placas pré-perfuradas (*stripboard*) a soldadura às pistas dos quatro terminais de massa centrais pode ajudar a dissipar.

Uma maneira de reduzir os efeitos do ruído eléctrico sobre o microcontrolador, consiste em usar fontes de alimentação separadas para a parte de controlo (microcontrolador) e para os motores. No caso dos robôs que iremos construir, pode usar-se uma vulgar pilha de 9V com regulador de tensão para 5V para a electrónica e um *pack* de baterias de modelismo (NiCad ou NiMH de 9,6V) para os motores. Naturalmente continua a ser necessário manter um terminal de massa comum às duas alimentações. A separação total de circuitos pode ser realizada usando circuitos optoelectrónicos mas apenas se justifica, face à sua complexidade, no caso de ruído eléctrico excessivo, resultante de comutação de relés ou motores.

2.2.6. Motor passo-a-passo unipolar

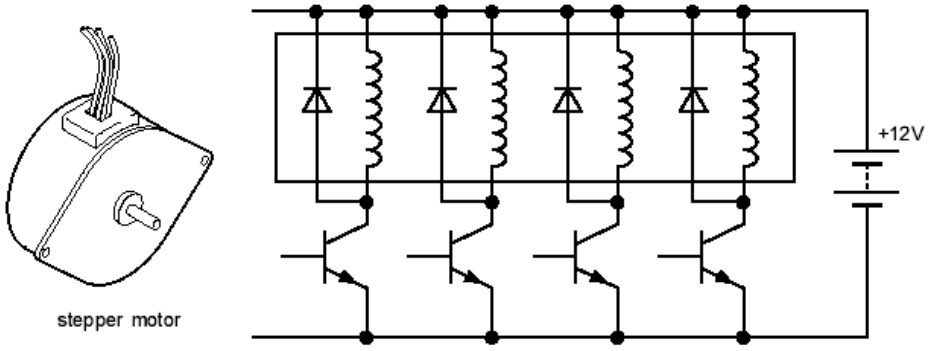
Os motores passo-a-passo são motores muito precisos com aplicação corrente em leitores de discos de computadores, impressoras e relógios. Ao contrário dos motores CC, que rodam livremente quando a alimentação é aplicada, os motores passo-a-passo requerem que a alimentação seja aplica segundo um determinado padrão. Para cada impulso, o motor passo-a-passo roda um passo (*step*), normalmente 7.5 graus (o que dá 48 passos para uma rotação completa).

Existem dois tipos principais de motores passo-a-passo - Unipolares e Bipolares.

Os motores unipolares possuem normalmente quatro enrolamentos que são ligados (on) e desligados (off) numa sequência determinada. Os motores bipolares possuem dois enrolamentos nos quais a corrente é invertida numa sequência idêntica. O uso de motores bipolares é tratado no capítulo seguinte.

Cada um dos quatro enrolamentos de um motor unipolar tem que ser comutado *on* e *off* numa certa ordem para que o motor rode. Muitos sistemas controlados por

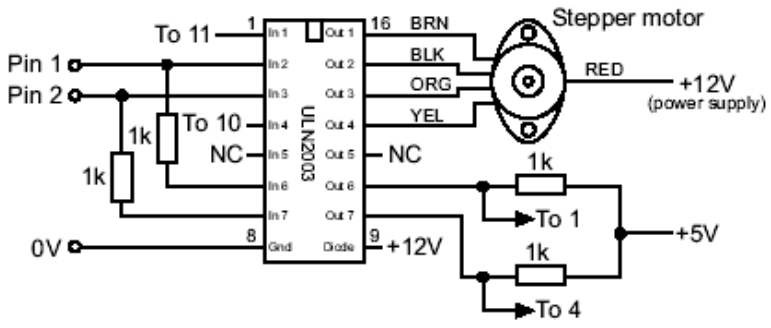
microprocessador utilizam quatro pinos de saída para controlar cada um dos quatro enrolamentos.



Como o motor passo-a-passo é alimentado a 12V, é necessário um circuito a transístor para cada enrolamento. Como os enrolamentos são indutivos, é necessário proteger cada transístor com um díodo de roda-livre. O uso do integrado ULN2003 (ou 2803) é neste caso de particular relevância, simplificando o número de componentes e as ligações.
A tabela abaixo apresenta os quatro passos necessários para que o motor rode.

Step	Coil 1	Coil 2	Coil 3	Coil 4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

Analisando com cuidado a tabela, reparamos que existe um padrão. O enrolamento 2 é sempre o oposto (operação lógica NOT) do enrolamento 1. O mesmo se aplica aos enrolamentos 3 e 4. É assim possível reduzir a metade o número de pinos do microcontrolador necessários para controlar o motor, desde que se acrescentem duas portas lógicas NOT. Na figura seguinte, este sistema é implementado usando os transístores livres como inversores (portas NOT).



Nota: existem várias normas quanto às cores dos condutores do motor.

Antes de começar a programar, existe um outro padrão a verificar na sequência de passos.

Analise a tabela, que mostra os enrolamentos 1 e 3.

Step	Coil 1	Coil 3	Change
1	1	1	
			coil 3
2	1	0	
			coil 1
3	0	0	
			coil 3
4	0	1	
			coil 1
1	1	1	

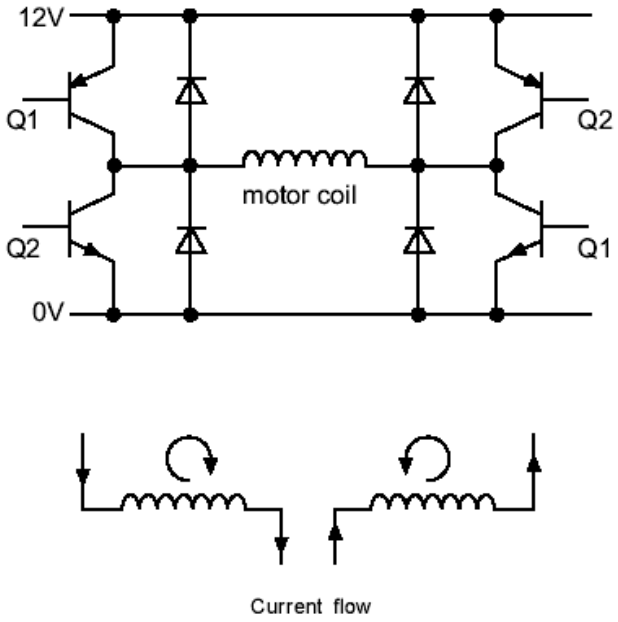
Repare que na passagem do passo 1 para o passo 2, apenas muda o enrolamento 3. Repare agora no passo seguinte – só o enrolamento 1 muda. De facto os dois enrolamentos mudam "à vez" de alto para baixo sucessivamente. Esta passagem alto-baixo-alto pode ser descrita como um estado de "toggling". Isso torna a programação ainda mais simples – basta usar a instrução *toggle*.

```
steps:
toggle 1      ; Toggle pino 1
pause 200     ; Espera 200 ms
toggle 2      ; Toggle pino 2
pause 200     ; Espera 200ms
goto steps    ; Ciclo
```

Nota: Se o motor 'trepidar', verifique a ordem de ligações dos condutores (cores).

2.2.7. Motor passo-a-passo bipolar

Os motores bipolares possuem dois enrolamentos que têm que ser accionados de modo a que a corrente circule nos enrolamentos pela sequência adequada. A mudança de campo magnético que esses enrolamentos produzem faz com que o motor (rotor) rode por passos. O circuito normalmente utilizado para cada enrolamento é o apresentado na figura seguinte. Repare que existem quatro transístores de controlo, que são activados "aos pares". Assim, para dois enrolamentos existem quatro pares de transístores de controlo (Q1-Q4) que têm que ser activados *on* e *off* numa determinada sequência.



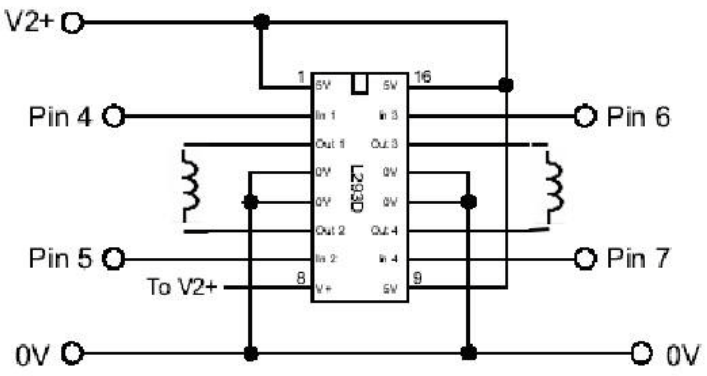
Note que como os enrolamentos são indutivos, para proteger os transístores da extra-corrente de rotura, são necessários díodos de roda-livre em paralelo com cada um deles.

A tabela abaixo mostra os quatro passos necessários para fazer rodar o motor.

Step	Q1	Q2	Q3	Q4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

Afortunadamente, o *motor driver L293D* foi desenhado especificamente para esta função de comutação. Cada L293D possui 8 transístores e os respectivos díodos num encapsulamento de 16 pinos.

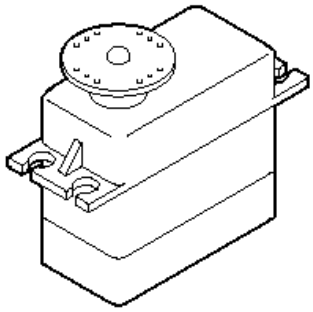
Quatro pinos do microcontrolador são ligados aos quatro pares de transístores (pinos 2, 7, 10 e 15).



O programa junto faz o motor rodar 100 passos para a esquerda e a seguir 100 passos para a direita, utilizando duas subrotinas. A subrotina *lstep* faz o motor rodar um passo para a esquerda, a subrotina *rstep* faz o motor rodar um passo para a direita. A variável b1 é utilizada para guardar a posição do passo e deverá ser utilizada em qualquer outra parte do programa.

```
main:
for b3 = 0 to 99                          ; inicia um ciclo for...next
  gosub lstep                             ; chama subrotina left step
next b3                                   ; ciclo seguinte

for b3 = 0 to 99                          ; inicia um ciclo for...next
gosub rstep                               ; chama subrotina  right step
next b3                                   ; ciclo seguinte

lstep: let b1 = b1 + 1                    ; soma 1 à variável b1
goto step                                 ; salta para tabela lookup
rstep: let b1 = b1 - 1                    ; subtrai 1 da variável b1
step: let b1 = b1 & 2                     ; mascara os dois bits menos pesados de b1
lookup b1,(%1010,%1001,%0101,%0110),b2    ; guarda o código lookup em b2
let pins = b2                             ; faz sair o valor de b2 para os pinos de controlo
return
```

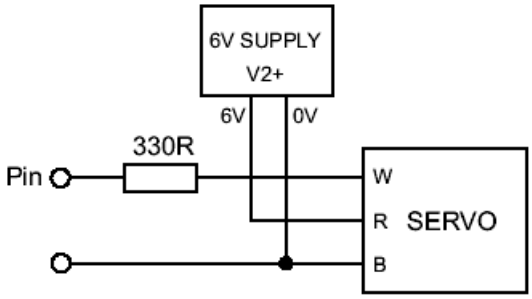
2.2.8. Servo radio-controlado



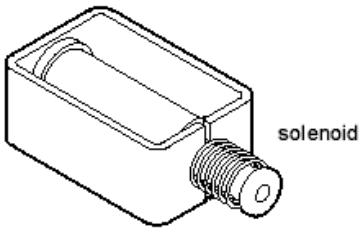
Os servos são vulgarmente utilizados nos carros e aviões radiocontrolados para controlar mecanismos de direcção. Trata-se de dispositivos muito precisos que rodam sempre o mesmo ângulo para um dado sinal, pelo que são ideais para utilização em automatismos. Um servo típico possui três condutores de ligação, normalmente preto, encarnado e branco (ou amarelo). O condutor preto é a referência de massa da alimentação (0 volts), o condutor encarnado é a alimentação de 5V (de 4,8V a 6V) e o condutor branco (ou amarelo) é o sinal de posicionamento. Este sinal de posição é um impulso de 0,75 a 2,25 milisegundos (ms), repetido cada 18 ms aproximadamente (portanto, temos 50 impulsos por segundo). Com o impulso de 0,75 ms o servo move-se (roda) para uma das extremidades e com o impulso de 2,25 ms para a extremidade oposta. Desse modo, com um impulso de 1,5 ms, o servo roda para a posição central. Se os impulsos terminarem o servo move-se ao acaso. Lamentavelmente, os servos consomem correntes elevadas (de 200 mA a 1 A) e introduzem ruído eléctrico nos condutores de alimentação. Deverão, portanto, ser alimentados por fonte de alimentação

separada, como se mostra na figura abaixo. Lembre-se que o condutor de massa (referência) deve ser comum às duas alimentações.

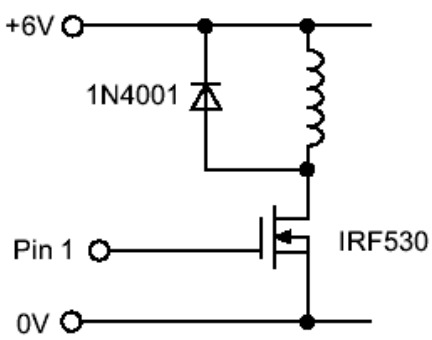
```
loop:
        servo 4,75              ; move servo para um dos lados
        pause 2000              ; espera 2 segundos
        servo 4,150             ; move servo para posição central
        pause 2000              ; espera 2 segundos
        servo 4,225             ; move servo para o outro lado
        pause 2000              ; espera 2 segundos
        goto loop               ; volta para o início
```



2.2.9. Solenóides e válvulas solenóides

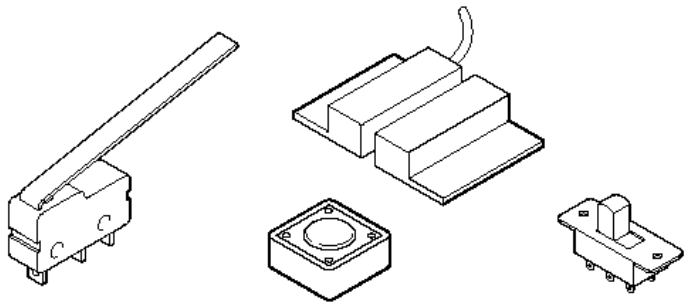


Um solenóide consiste num êmbolo de ferro colocado no interior de uma bobina (indutância) eléctrica, enrolada em torno de um tubo. Quando se alimenta electricamente a bobina, cria-se um campo magnético que atrai o êmbolo para dentro do tubo. Quando se retira a alimentação, uma mola empurra o êmbolo para fora do tubo. O controlo de um solenóide pode ser efectuado com o circuito a MOSFET atrás descrito.



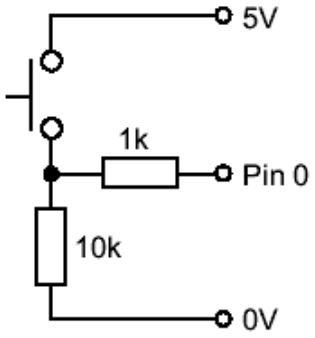
2.3. Interface com Dispositivos de Entrada

2.3.1. Interruptores



Existe disponível uma enorme variedade de interruptores, mas a maior parte possui dois contactos, que ou estão "abertos" (*off*) ou "fechados" (*on*). Os dois circuitos apresentados abaixo podem ser usados com a maioria dos interruptores.

Com este primeiro circuito o pino de entrada (Pin 0) fica baixo (0) quando o interruptor está aberto e alto (1), quando o interruptor é fechado.

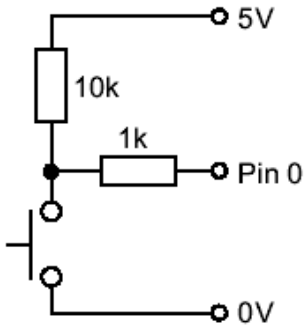


Um programa possível

Vai para 'salta' quando o interruptor **<u>está aberto</u>**:
if pin0 = 0 then salta

Vai para 'salta' quando o interruptor **<u>é fechado</u>**:
if pin0 = 1 then salta

Com o circuito seguinte o pino de entrada (Pin 0) fica alto (1) quando o interruptor está aberto e baixo (0), quando o interruptor é fechado.



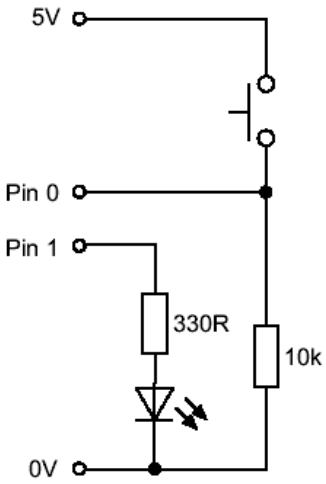
Um programa possível

Vai para 'salta' quando o interruptor **<u>está aberto</u>**:
if pin 0 = 1 then salta

Vai para 'salta' quando o interruptor **<u>é fechado</u>**:
if pin 0 = 0 then salta

Interruptores e ressaltos (debouncer)

Todos os interruptores mecânicos "ressaltam" quando o interruptor abre ou fecha. Isso significa que os contactos mecânicos "ressaltam" um sobre o outro antes de estabilizarem. Como o microcontrolador é muito rápido, pode acontecer em certos programas que registe dois ou três contactos em vez de registar apenas um. O processo mais simples de ultrapassar este problema é simplesmente acrescentando um tempo de espera (pause 100) depois de cada instrução **if**…. Se a secção de código a seguir à pressão do botão for longa, este atraso ocorre naturalmente, já que cada instrução leva certo tempo a ser executada. No entanto, se o código não for longo, deve incluir-se um atraso. Outra solução consiste em utilizar a instrução **button**.

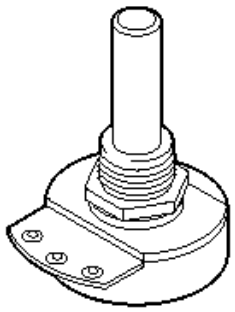


Os dois programas seguintes mostram o efeito dos "ressaltos". O programa deveria acender o LED no pino 1 quando o interruptor ligado ao pino 0 fosse pressionado mais do que cinco vezes. Contudo o primeiro programa não funciona correctamente, pois o microcontrolador irá contar ressaltos e não os impulsos reais, pelo que o LED acenderá prematuramente.

```
init: let b0 = 0
main: if pin 0 = 1 then add
goto main
add: let b0 = b0 + 1
if b0 < 5 then main
high 1
goto main
```

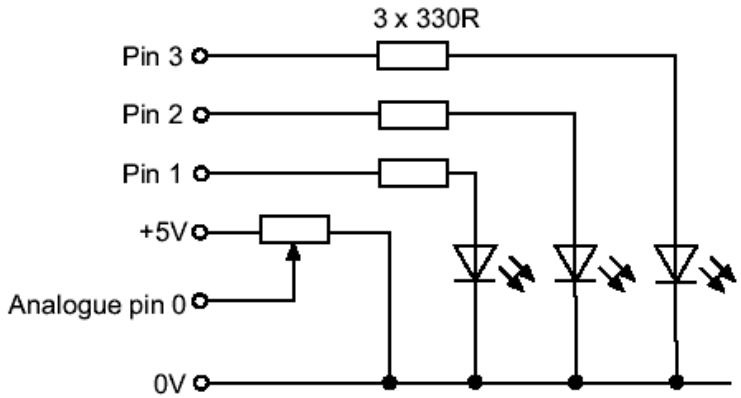
```
init: let b0 = 0
main: if pin 0 = 1 then add
goto main
add: pause 100                ; pequeno atraso aqui
let b0 = b0 + 1
if b0 < 5 then main
high 1
goto main
```

2.3.2. Potenciómetro



Um potenciómetro (ou resistência variável) possui um eixo que pode ser rodado para variar o valor da resistência do potenciómetro. Podemos usar potenciómetros para medir movimentos circulares ou ângulos. Para serem utilizados com o microcontrolador, os potenciómetros são colocados como divisores de tensão e o microcontrolador lê efectivamente a tensão que converte para um valor numérico (digital). Para isso usa-se a

instrução readADC que activa o Conversor Analógico-Digital. O valor da resistência vem assim transformado num valor entre 0 e 255, que é guardado numa variável. A instrução **if**…poderá agora ser utilizada para definir as acções a desenvolver consoante o valor lido.

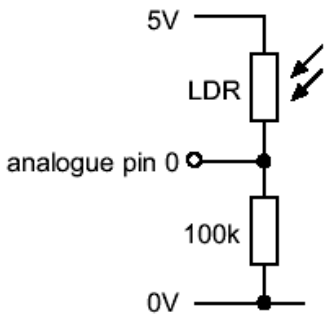


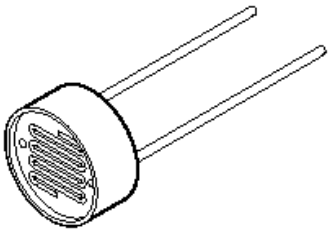
O programa abaixo acende três LEDs diferentes (ligados aos pinos 1,2 e 3), dependendo da leitura do sensor analógico (potenciómetro)

```
main:   readadc 0,b1            ; lê o valor no  pin0 para a variável b1
        if b1<75 then light1    ; se b1 é menor do que 75 então salta para light 1
        if b1<175 then light2   ; se b1 é menor do que 175 então salta para light 2
        goto light3             ; se b1 é maior do que 175 então salta para light 3

light1: high 1                  ; LED 1 on
        low 2                   ; LED 2 off
        low 3                   ; LED 3 off
        goto main               ; loop

light2: low 1                   ; LED 1 off
        high 2                  ; LED 2 on
        low 3                   ; LED 3 off
        goto main               ; loop

light3: low 1                   ; LED 1 off
        low 2                   ; LED 2 off
        high 3                  ; LED 3 on
        goto main               ; loop
```

2.3.3. LDR (Light Dependent Resistor)

Uma Light Dependent Resistor (LDR) é uma resistência cujo valor depende da luz recebida. Um dispositivo muito frequente, o ORP-12, possui uma resistência muito elevada no escuro, e uma resistência baixa à luz. A ligação de uma LDR ao microcontrolador é muito fácil, exigindo no entanto calibração.

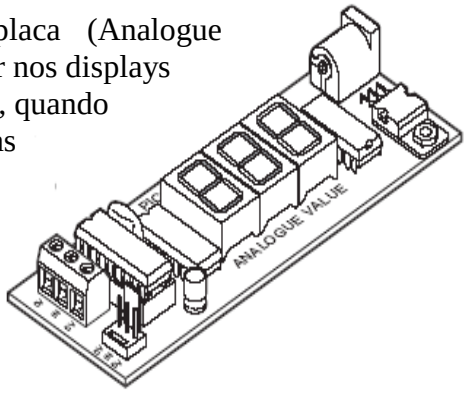




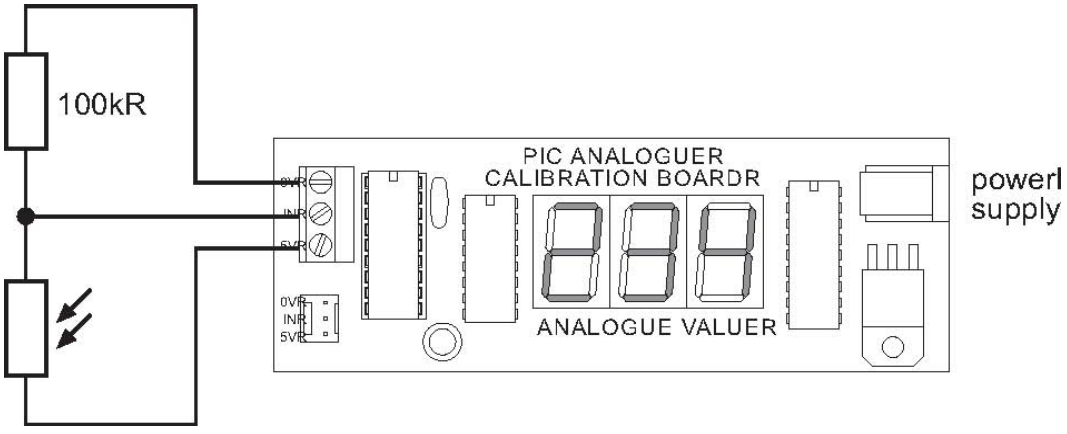
Convém lembrar que a resposta de uma LDR não é linear, pelo que as leituras não variam como num potenciómetro. Em geral, existe uma grande variação de resistência para níveis de iluminação elevados. Isso pode ser compensado por software, utilizando gamas de variação menores para os níveis escuros. Experimente até encontrar os ajustes adequados ao circuito e às condições de iluminação.

```
main:
        readadc 0,b1            ; leitura do valor
        if b1<50 then light1    ; gama 0-50 = 50
        if b1<100 then light2   ; gama 50-100 = 50
        if b1<145 then light3   ; gama 100-145 = 45
        if b1<175 then light4   ; gama 145-175 = 30
        goto main
```

NOTA: A PICAXE dispõe de uma placa (Analogue Calibration Board – **BAS810**) que permite ver nos displays de 7 segmentos o valor exacto, entre 0 e 255, quando um sensor está lá ligado. Permite tirar leituras nas mais diversas condições de teste de modo a obtermos a melhor gama de valores para um programa.
Se quiser consultar mais informações:
www.techsupplies.co.uk/cgi-bin/techsupplies.storefront/43afe871050f0efa2740c2c98abb0710/Product/View/BAS810

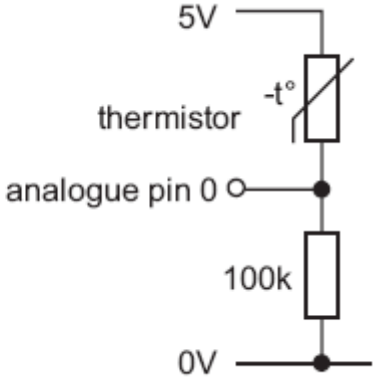


Exemplo de ligação:



2.3.4. Termistor

Um termistor é uma resistência variável com a temperatura. Na realidade todas as resistências variam com a temperatura, só que os termistores são feitos para terem uma grande variação com a temperatura. A ligação do termistor ao microcontolador é muito simples, mas convém também calibrá-lo (com o **BAS810**, por exemplo).



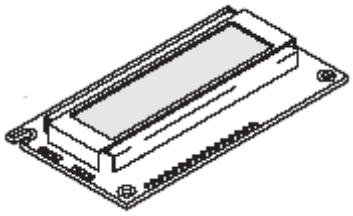
Convém lembrar que a resposta de um termistor não é linear, pelo que as leituras não variam como num potenciómetro. Em geral, existe uma grande variação de resistência para temperaturas baixas. Isso pode ser compensado por software, utilizando gamas de variação menores para os níveis de temperatura mais altos. Experimente até encontrar os ajustes adequados ao circuito e às condições de temperatura.

```
main:
        readadc 0,b1            ; leitura do valor
        if b1<50 then light1    ; gama 0-50 = 50
        if b1<100 then light2   ; gama 50-100 = 50
        if b1<145 then light3   ; gama 100-145 = 45
        if b1<175 then light4   ; gama 145-175 = 30
        goto main
```

2.4. Interface com Componentes Avançados

2.4.1. Display LCD (Liquid Crystal Display)

Um Display de Cristal Liquido é um dispositivo electrónico que permite mostrar caracteres como numeros, texto e quaisquer outros, incluindo pontuação. Existem dois tipos principais de displays LCD, os displays numericos (tipo dos usados nos relógios e calculadoras) e os displays alfanuméricos (como os usados nos telemóveis).



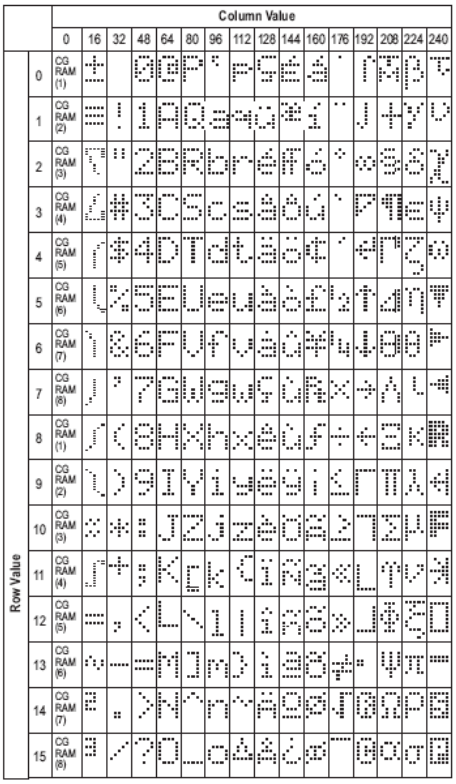
Nos displays numéricos o conjunto de cristais está distribuído em segmentos, enquanto que nos alfanunéricos estão sob a forma de pontos.Cada crystal tem uma ligação eléctrica individual por forma a ser controlado independentemente. Quando o cristal está 'off' (i.e. quando não há tensão aplicada ao cristal) o cristal não se vê porque reflecte a mesma quantidade de luz que o material do fundo onde está. Contudo, quando o cristal tem uma tensão aplicada, fica polarizado, absorvendo luz e portanto aparecendo um segmento, ou ponto, mais escuro que pode ser visualizado contra o material do fundo.

É importante perceber a diferença entre os displays LCD e LED. Os LED fornecem luz (podendo ser vistos na escuridão) enquanto o LCD apenas reflecte a luz que recebe. Em

contrapartida há consumo de energia nos LED enquanto que nos LCD praticamente não há consumos.

2.4.1.1. Caracteres LCD

A tabela seguinte mostra os caracteres disponíveis num display LCD típico. O código do character é obtido adicionando o numero da coluna ao número da linha.



Os caracteres, do 32 ao 127 são sempre iguais independentemente do LCD, enquanto os characters do 16 ao 31 e do 128 ao 255 diferem conforme o fabricante do LCD. Claro que, sendo assim, alguns LCD poderão mostrar caracteres diferentes dos da tabela ao lado.

Do caracter 0 ao 15 são normalmente designados como caracteres 'user-defined' e assim, devem ser definidos antecipadamente, de acordo com os manuais de cada fabricante, para evitar que, para esses códigos, apareçam caracteres aleatórios.

O funcionamento de um display LCD é relativamente complexo dado que cada display pode mesmo armazenar mais caracteres do que aqueles que ele tem capacidade de mostrar na sua janela. E, na sua janela, de uma maneira geral, o display só consegue mostrar 16 caracteres de cada vez.

Só que na maioria dos displays LCD há uma memória que permite armazenar 40 caracteres por cada linha que o display tenha. Cada espaço na memória RAM é uma "caixa" para um character e tem um endereço que o descreve. A primeira linha tem os endereços 128 a 191. A segunda linha, os endereços de 192 a 255.

Os displays de 16x2 têm uma janela com duas linhas. Quer dizer, podemos ver 16 caracteres em cada uma das duas linhas. O caracter que é para aparecer na segunda linha, obriga-nos a mover o cursor para o início dessa linha.

Para mover o cursor, para para qualquer posição do display, basta enviar o endereço da RAM como instrução. Isto quer dizer que para enviar para a primeira posição da primeira linha (endereço '128'), é enviar este endereço como instrução. Ora, para mover o cursor para o início da linha 2 basta enviar o endereço ('192') da RAM como instrução para o módulo LCD. Para mover o cursor para a quinta posição da segunda linha tem que se enviar a instrução '197' (=192+5).

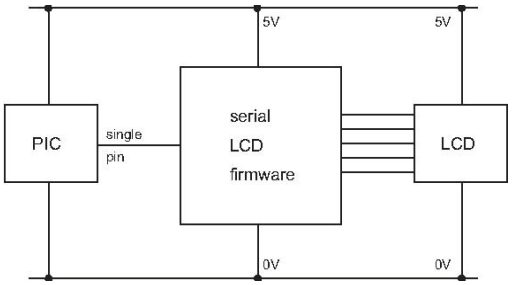
2.4.1.2. Interface de LCD a microcontoladores

Há três formas principais de fazer o interface de LCD a microcontroladores.

1) Com um integrado com firmware Série para LCD
2) Com um módulo Série LCD (tem uma PCI já com o integrado com firmware)
3) Ligando Directamente

Ligando o LCD usando O FRM010 (Serial LCD Firmware Chip)

O FRM010 é usado para permitir o controlo série de um LCD alfanumerico. Assim permite aos micocontroladores (e sistemas baseados em microcontroladorescomo os PICAXE e Basic Stamp) utilizar um ecrã onde seja visualizado o texto que o utilizador queira e de uma forma relativamente simples. Pois, todos os comandos do LCD são transmitidos a partir do microcontrolador através de uma única linha série, como se pode ver pela figura ao lado e usando a instrução *serout*.

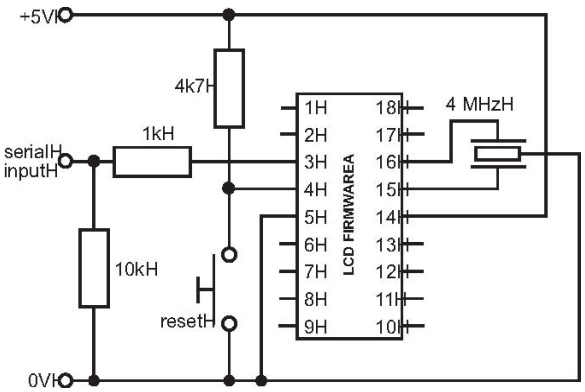


Por exemplo, para que apareça escrito o texto 'Hello' a instrução é simplesmente:

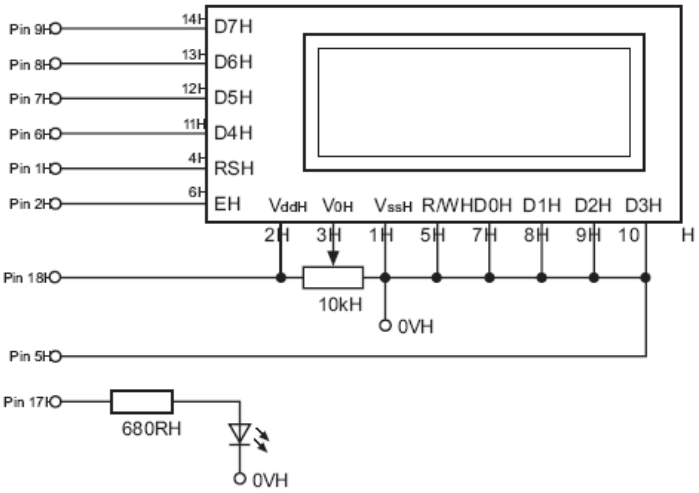
```
serout 7,T2400,("Hello")
```

O preço a pagar é que se têm que fazer as ligações de acordo com os dois esquemas abaixo:

A) LIGAÇÕES DO FIRMWARE



B) LIGAÇÕES DO MÓDULO LCD

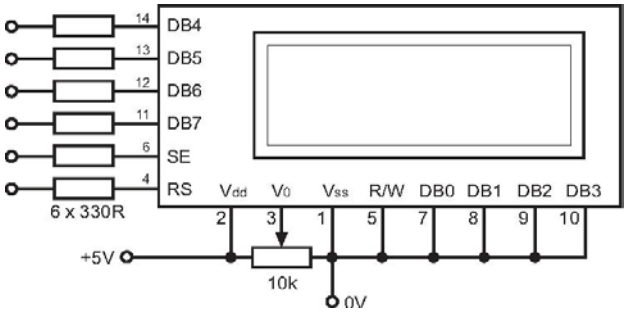


Ligando o LCD usando o AXE033 (Serial LCD Module)

O AXE033 é um módulo série em PCI (é vendido em kit) já com todas as ligações feitas do LCD para o integrado do firmware.
Portanto, o módulo vem-nos ajudar a resolver as ligações necessárias na opção anterior, além de que este módulo permite o uso do protocolo de comunicação i2c.

Ligando o LCD directamente

O LCD tem 6 linhas que podem ser ligadas directamente aos pinos do microcontrolador. Contudo é uma boa prática de projecto ligar resistências de 330R às linhas para proteger de descargas estáticas.
O potenciómetro de 10k ligado ao pino 3 tem como missão ajustar o contraste do display. Todas os pinos do LCD não usados, como se vê na figura, devem ficar ligados à massa.



Um Programa Simples para escrever no LCD

O Programa que se segue escreve a frase 'Hello there!' em duas linhas do display LCD. Temos que usar três subrotinas standard chamadas ***init, wrins*** e ***wrchr***. Elas resolvem todos os nossos problemas não só de inicialização como de envio para o LCD tanto das instruções como das mensagens.
Essas rotinas standard fazem exactament o seguinte:

init Faz a inicialização do LCD, deixando-o pronto a aceitar as instruções
wrins envia as instruções armazenadas na variável b1 para o módulo LCD
wrchr envia um caracter armazenado na variável b1 para ser mostrado no ecrã LCD

Já à frente serão explicadas mais detalhadamente estas três subrotinas.

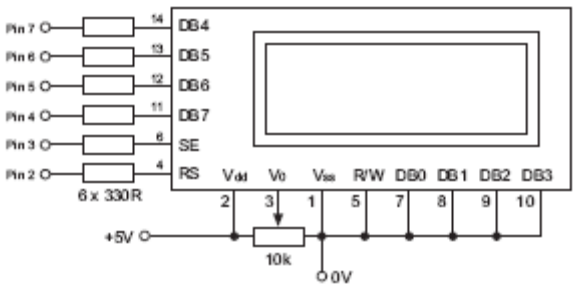
```
EEPROM 0,("Hello there!")        ' armazena o texto na memória EEPROM

        gosub init               'initializa o LCD

main:   let b1 = 1               'b1=1 => instrução 'clear display'
        gosub wrins              ' envia instrução para o LCD

        for b3 = 0 to 4          'ciclo for...next ("Hello"  4 caracteres)
            read b3, b1          ' b1=EEPROM[b3]
            gosub wrchr          ' envia character para LCD
        next b3                  ' continua

        let b1 = 192             'b1=192 => instrução 'start of second line' position
        gosub wrins              ' envia instrução para o LCD

        for b3 = 5 to 11         'ciclo for...next ("there!"-posições 5 a 11)
            read b3, b1          ' b1=EEPROM[b3]
            gosub wrchr          ' envia character para LCD
        next b3                  'continua
```

Um Programa mais Avançado

O programa seguinte faz rodar na janela do LCD a mensagem 'Hello there everybody!'. Isto porque o texto tem mais do que as 16 letras que um ecrã suporta. A mensagem é primeiro armazenada na memória do LCD, indo depois rodando na janela do display repetidamente para que seja mostrada toda a mensagem.

```
EEPROM 0,("Hello there everybody!")        ' armazena o texto na memória EEPROM

        gosub init                  ' initializa o LCD

start:  let b1 = 1                  'b1=1 => instrução 'clear display'
        gosub wrins                 ' envia instrução para o LCD

        for b3 = 0 to 22            ' ciclo for...next ("Hello there everybody!" 22 caracteres)
            read b3, b1             ' b1=EEPROM[b3]
            gosub wrchr             ' envia character para LCD
        next b3                     ' continua

        let b1 = 12                 'b1=12 => instrução 'hide cursor'
        gosub wrins                 ' envia instrução para o LCD
main:    let b1 = 24                'b1=24 => instrução 'scroll display left'
        gosub wrins                 ' envia instrução para o LCD
        pause 250                   ' pausa  0.25s
        goto main                   ' roda a mensagem ("Hello there everybody!")
```

Subrotinas Standard para ligar o LCD directamente.



Nos programas que vimos fazendo, vemos que o LCD tanto tem que receber instruções como caracteres. É o pino 4 (RS) do LCD – controlado pela saída 2 do microcontrolador – que define se é instrução ou caracter o que está à entrada. Pondo o pin 2 "baixo" o LCD está em modo de instrução enquanto que quando o pin 2 está "alto" fica em modo de caracter.

Cada instrução ou caracter é enviada como um número binário de 4 bits para as linhas de dados (pinos 7-4). Depois o LCD, quando recebe um impulso de ENABLE, no seu pino 6 - SE (e que é enviado pela saída 3 do microcontrolador) lê os dados e imprime o caracter (ou cumpre a instrução) correspondente ao número nas linhas de dados.

Todos os códigos de caracteres e instruções (comandos) têm 8 bit. Como só há 4 linhas de dados, os 8 bit são mandados em duas metades, uma a seguir à outra. Cada um dos quatro bit recebe o nome de "nibble" ('nibble alto' e 'nibble baixo').

Portanto, por cada caracter (ou comando) são transmitidos 2 nibbles.

1011 0101 = 10110101

nibble alto + nibble baixo = byte

A função das três subrotinas ‘standard’ (***init, wrins*** e ***wrchr***) é levar a cabo toda esta tarefa “complicada” quando trabalhamos com displays LCD.

```
init:  let pins = 0              ‘ Todas saídas=0
       let b4 = 0                ‘ Reset variavel
       let dirs = 252            ‘ Pins 2-7 como saída (%11111100).
       pause 200                 ‘ 200 ms para reset ao LCD.
       let pins = 48             ‘ Funcionamento LCD a 8-bit.
       pulsout 3,1               ‘ Envia comando (enable)
       pause 10                  ‘ Espera 10 ms
       pulsout 3,1               ‘ Envia comando outra vez(enable)
       pulsout 3,1               ‘ Envia comando outra vez(enable)
       let pins = 32             ‘ Funcionamento LCD a 4-bit.
       pulsout 3,1               ‘ Envia comando(enable)
       pulsout 3,1               ‘ Envia comando outra vez(enable)
       let pins = 128            ‘ Funcionar em duas linhas
       pulsout 3,1               ‘ Envia comando (enable).
       let b1 = 14               ‘ LCD ON e com cursor.
       gosub wrins               ‘ Ecrever instrucção no LCD
       return

wrchr: let pins = b1 & 240       ‘ nibble alto de b1=>fora(atenção RS=0).
       high 2                    ‘ Enviar Caracter=> RS alto
       pulsout 3,1               ‘ Pulsa enable (envia nibble alto).
       let b2 = b1 * 16          ‘ nibble baixo de b1=>b2.
       let pins = b2 & 240       ‘ nibble baixo de b1=>fora(atenção RS=0).
       high 2                    ‘ Caracter=> RS alto
       pulsout 3,1               ‘ Pulsa enable pin (envia nibble baixo).
       Return

wrins: let pins = b1 & 240       ‘ nibble alto de b1=>fora(RS=0->instruç).
       pulsout 3,1               ‘ Pulsa enable (envia nibble alto).
       Let b2 = b1 * 16          ‘ b2=b1 rodado 4x p/ esquerda.
       let pins = b2 & 240       ‘ nibble baixo de b1=>fora(+RS=0-instruç).
       pulsout 3,1               ‘ Pulsa enable pin (envia nibble baixo).
       high 2                    ‘ Modo character
       return
```

*Nota: na subrotina **init**, o comando let dirs =252(%11111100) vai afectar os 8 pinos e não apenas os 6 usados pelo LCD. Como se pode ver, nem **wrins** nem **wrchr** necessitam dos pinos 0 e 1. Se o nosso programa usar estas rotinas deve ter em conta que, quando retorna da chamada, o valor de b2 vem alterado (assim como w0).*

Usando o conjunto de instruções (comandos) do LCD.

Os códigos das instruções de comando para o LCD, como vimos, devem “ir” na variável b1 quando se chama a subrotina ***wrins***, que se encarrega de modificar o estado do LCD.

Códigos das instruções (comandos) para o LCD.

1 Limpa o display e move para a primeira linha
2 Move o cursor e a janela de display para o início da primeira linha
4 Modo “escrever da direita para a esquerda”
5 Modo ‘scroll para a esquerda’
6 Modo “escrever da esquerda para a direita”
7 Modo ‘scroll para a direita’

10 Visualização do LCD off
12 Sem cursor
13 Cursor a piscar
14 Visualização do LCD (e cursor) on
16 Move cursor para a esquerda uma posição
20 Move cursor para a direita uma posição
24 Faz o scroll da janela uma posição para a esquerda
28 Faz o scroll da janela uma posição para a direita
128 Move o cursor para o início da primeira linha
192 Move o cursor para o início da segunda linha

Exemplos

Limpar o display

```
clear: let b1 = 1 ‘ b1=instrução clear
       call wrins ‘ Envia para o LCD
```

Mover o cursor para a segunda linha

```
clear: let b1 = 192 ‘ b1=início da segunda linha
       call wrins    ‘ Envia para o LCD
```

Piscar uma menssagem 10 vezes

```
flash:for b3 = 1 to 10          ‘ for...next usando variable b3(nunca b1!!)
             let b1 = 10        ‘ b1= Visualização do LCD off
             gosub wrins        ‘ Envia para o LCD
             pause 200          ‘ Pausa de 0.2 segundos
             let b1 = 14        ‘ b1= Visualização do LCD on
             gosub wrins        ‘ Envia para o LCD
             pause 200          ‘ Pausa de 0.2 segundos
       next b3                  ‘ Fim do loop for...next
```

Fazer scroll a uma mensagem longa (30 caracteres)

```
scroll: for b3 = 1 to 30        ‘ for...next usando variable b3(nunca b1!!)
             let b1 = 28        ‘ b1= Scroll da janela uma posição p/ direita
             gosub wrins        ‘ Envia para o LCD
             pause 200          ‘ Pausa de 0.2 segundos
        next b3                 ‘ Fim do loop for...next
        let b1 = 1              ‘ b1=instrução clear
        gosub wrins             ‘ Envia para o LCD
        pause 200               ‘ Pausa de 0.2 segundos
        goto scroll             ‘ Loop
```