Arquitetura de Computadores

PROF. ISAAC

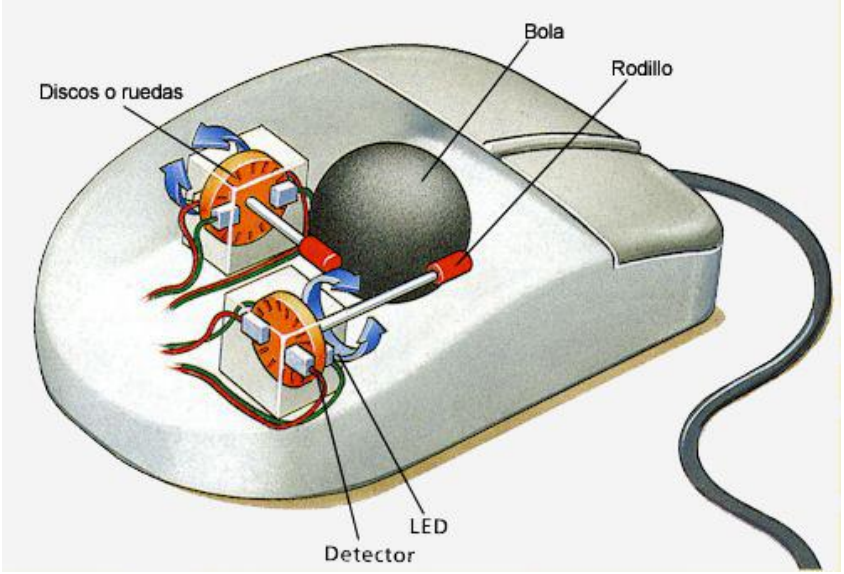
Dispositivos de Entrada

Mouse

A ideia de um dispositivo apontador como um mouse foi mostrada pela primeira vez por Doug Engelbart usando um protótipo em 1967.

A Macintosh incluiu um mouse como seu dispositivo apontador em 1973.

O mouse original era eletromecânico e usava uma grande esfera que, quando rolada sobre uma superfície, fazia com que um contador x e um y fossem incrementados.



Mouse

Atualmente os mouse são óticos, O mouse ótico é composto por um LED para fornecer iluminação, e um fotodetector (uma minúscula câmera).

O LED ilumina a superfície abaixo do mouse; e a câmera tira centenas/milhares de fotografias em cada segundo.

Os quadros sucessivos são enviados para um processador ótico simples, que compara as imagens e determina se e quanto o mouse foi movido.

Mouse

DPI - Dots per Inch

A sensibilidade do mouse é determinada pelo DPI. Ela representa o número de pontos nos quais o sensor do mouse divide a superfície que está lendo.

Quanto maior for o DPI, maior será a resolução da imagem captada pelo sensor fotodetector.

Quanto mais DPIs, mais sensível é o sensor do mouse, sendo capaz de detectar e reagir aos menores movimentos.

Um arranjo comum é enviar uma sequência de bytes ao computador toda vez que o mouse se movimenta, esse envio ocorre de forma serial.

Teclado

Em computadores pessoais, quando uma tecla é pressionada, uma interrupção é gerada e a rotina de interrupções do teclado (uma parte do software do sistema operacional) é executada.

A rotina de interrupções lê um registrador de hardware dentro do controlador de teclado para pegar o número da tecla (1 a 102) que acabou de ser pressionada.

Quando a tecla é solta, ocorre uma segunda interrupção. Assim, se um usuário pressionar SHIFT, e em seguida pressionar e soltar M, e depois soltar SHIFT, o sistema operacional pode ver que o usuário quer um “M”, e não um “m”.

O tratamento de sequências de várias teclas envolvendo SHIFT, CTRL e ALT é todo feito em software (incluindo a abominável sequência CTRL-ALT-DEL, que é usada para reiniciar PCs).

Teclado Matricial

Teclado Matricial

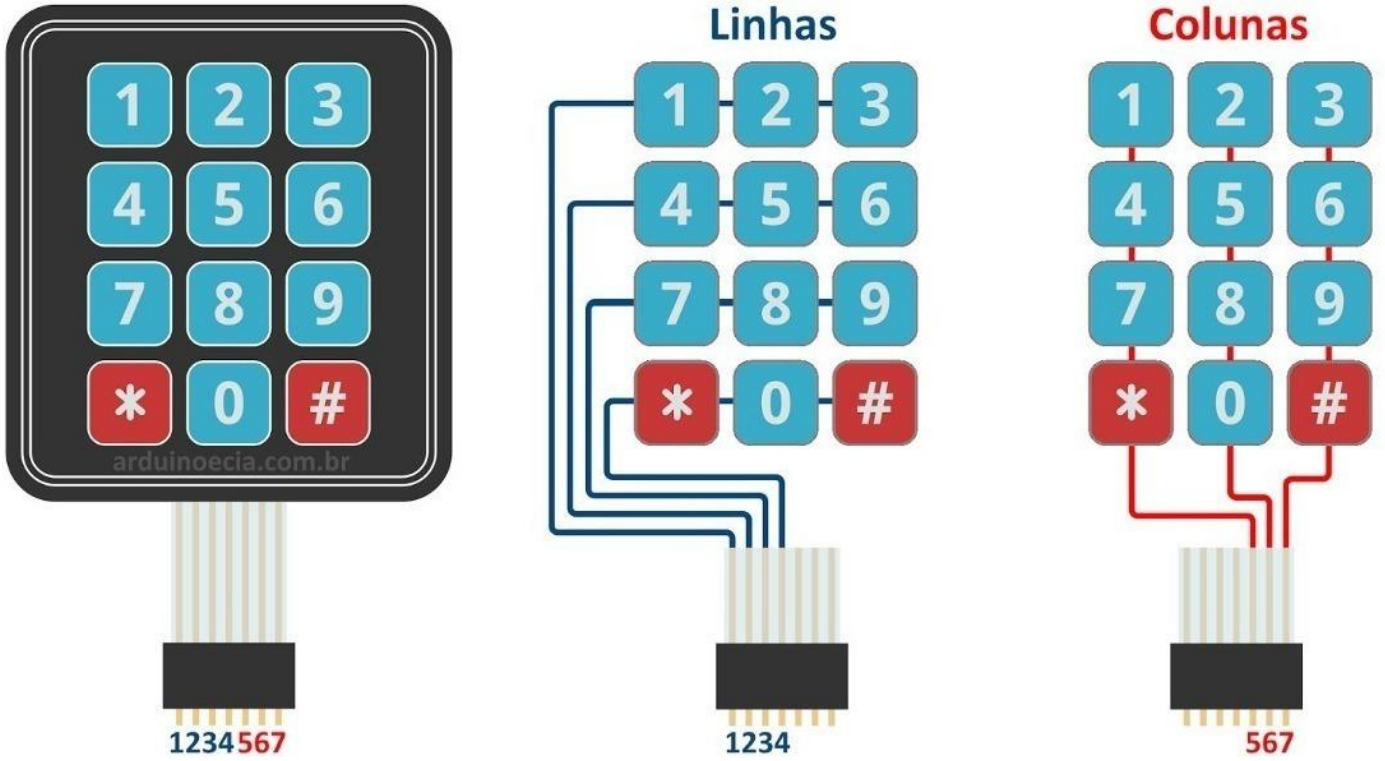
Teclado Matricial

4x3

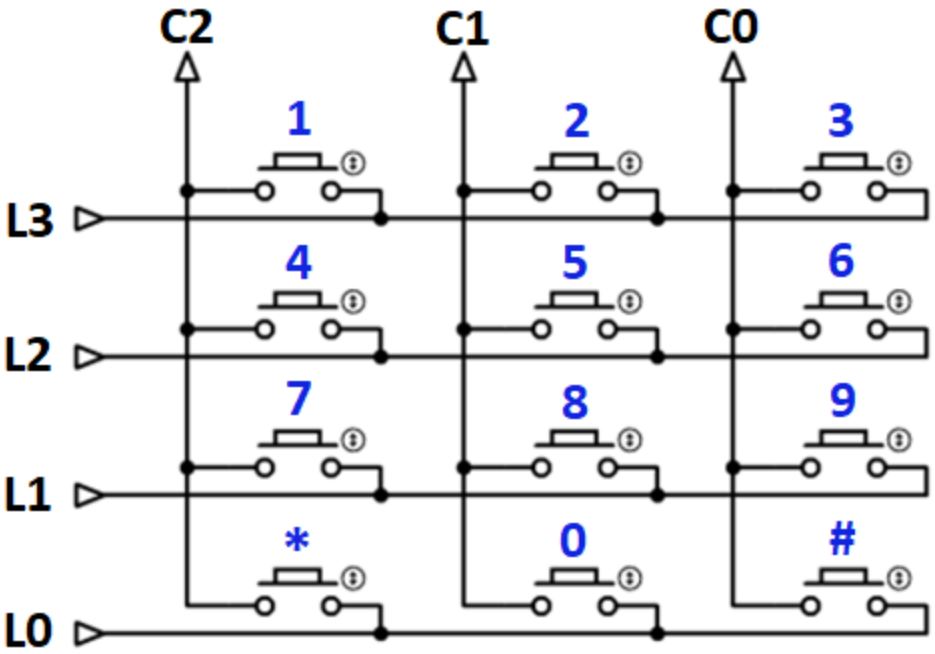
4x4

5x4

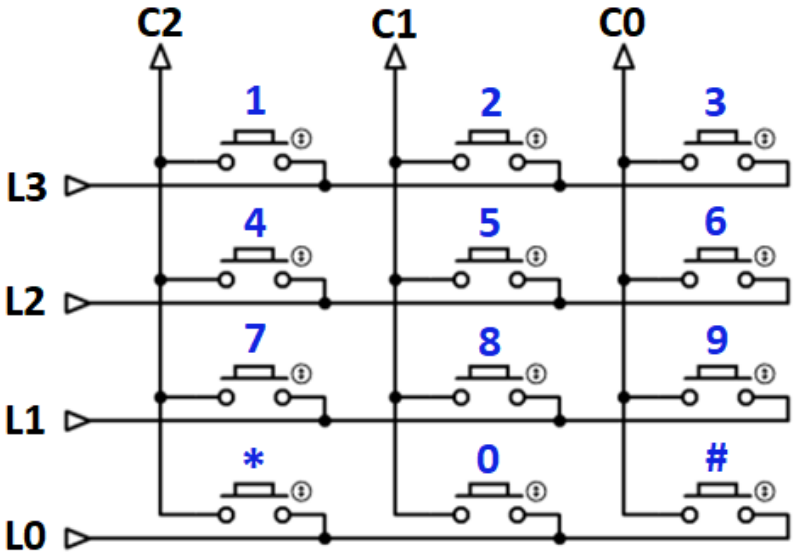
Teclado Matricial

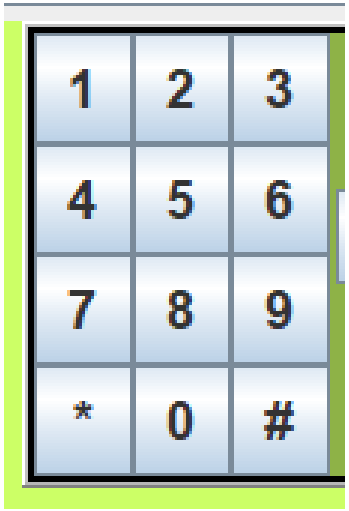


Teclado Matricial



Teclado Matricial no EdSim51

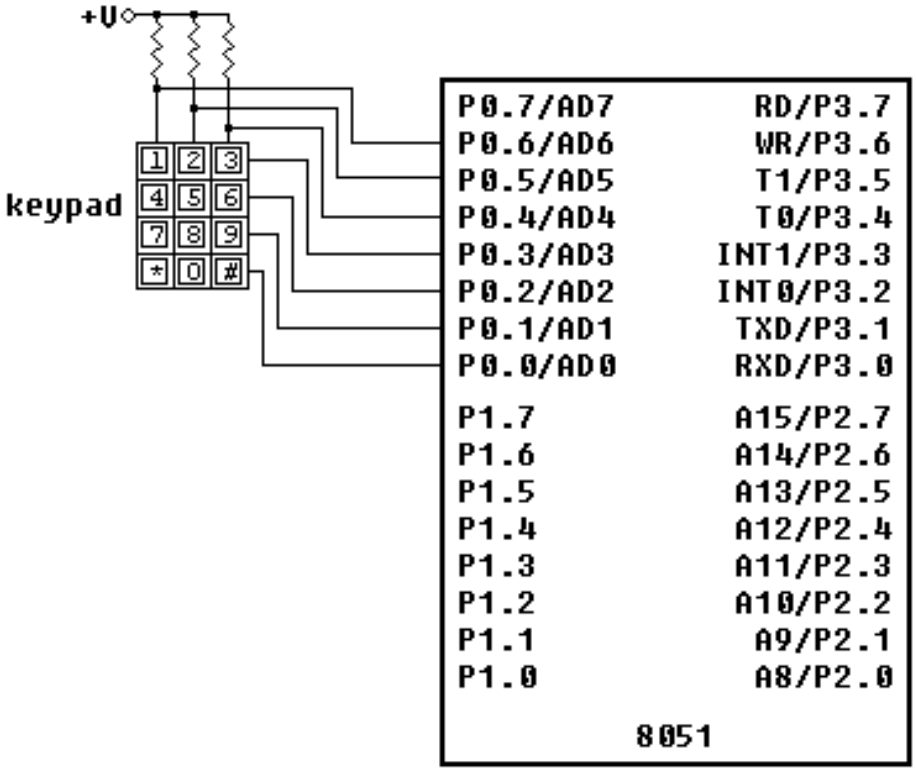




Ligação do Teclado Matricial no 8051 – edSim51

O teclado matricial está ligado nos pinos de P0.0 até P0.6, onde:

- As colunas estão ligadas nos pinos P0.6, P0.5 e P0.4.
- As linhas estão ligadas nos pinos P0.3, P0.2, P0.1 e P0.0.



Programação

Exemplo de leitura do Teclado Matricial

Este exemplo realiza a leitura do Teclado matricial, armazenando em R0 o valor numérico estabelecido pelo programa referente a tecla pressionada.

A figura abaixo apresenta a relação entre a tecla pressionada e o número que será armazenado em R0

```
+----+----+----+
| 11 | 1Ø |  9 | row3
+----+----+----+
|  8 |  7 |  6 | row2
+----+----|----+
|  5 |  4 |  3 | row1
+----+----+----+
|  2 |  1 |  Ø | rowØ
+----+----+----+
 col2 col1 colØ
```

Exemplo de leitura do Teclado Matricial

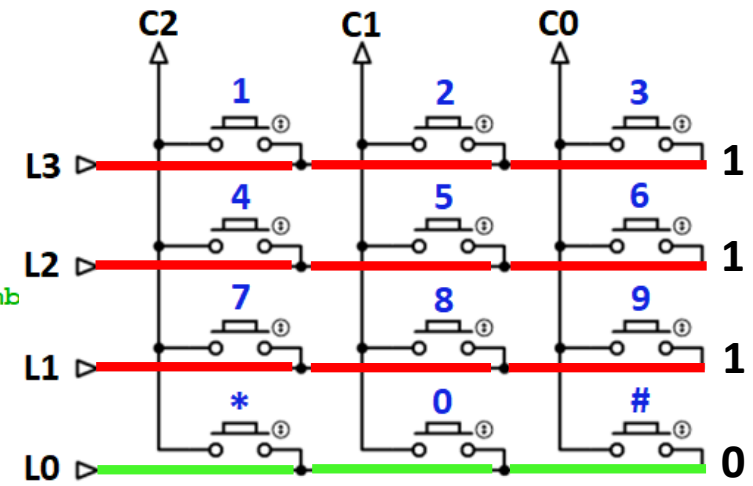
```
start:
   MOV RØ, #Ø          ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3          ; set row3
   CLR PØ.Ø           ; clear rowØ
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   ; scan row1
   SETB PØ.Ø          ; set rowØ
   CLR PØ.1           ; clear row1
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   ; scan row2
   SETB PØ.1          ; set row1
   CLR PØ.2           ; clear row2
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   ; scan row3
   SETB PØ.2          ; set row2
   CLR PØ.3           ; clear row3
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   JMP start          ; | go back to scan row Ø
                      ; | (this is why row3 is set at the start of the program
                      ; | - when the program jumps back to start, row3 has just been scanned)

finish:
   JMP $              ; program execution arrives here when key is found - do nothing
```

Exemplo de leitura do Teclado Matricial

```
; column-scan subroutine
colScan:
   JNB PØ.4, gotKey  ; if colØ is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.5, gotKey  ; if col1 is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.6, gotKey  ; if col2 is cleared - key found
   INC RØ            ; otherwise move to next key
   RET               ; return from subroutine - key not found
gotKey:
   SETB FØ           ; key found - set FØ
   RET               ; and return from subroutine
```

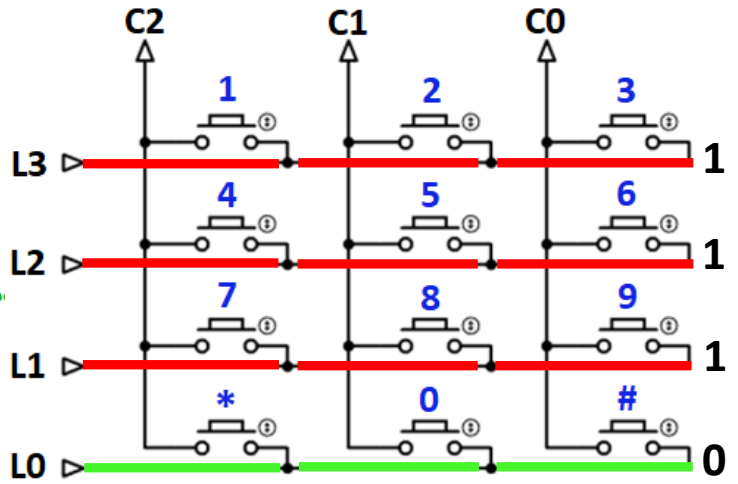
Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø          ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3           ; set row3
   CLR PØ.Ø            ; clear rowØ
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its numb
   ; scan row1
   SETB PØ.Ø           ; set rowØ
   CLR PØ.1            ; clear row1
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row2
   SETB PØ.1           ; set row1
   CLR PØ.2            ; clear row2
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row3
   SETB PØ.2           ; set row2
   CLR PØ.3            ; clear row3
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   JMP start           ; | go back to scan row Ø
                       ; | (this is why row3 is set at the start of the program
                       ; | - when the program jumps back to start, row3 has just been scanned)


finish:
   JMP $               ; program execution arrives here when key is found - do nothing
```



Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø        ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3       ; set row3
   CLR PØ.Ø        ; clear rowØ
   CALL colScan    ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of program
                   ; | (because the pressed key was found and its numb
   ; scan row1
   SETB PØ.Ø       ; set rowØ
   CLR PØ.1        ; clear row1
   CALL colScan    ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of program
                   ; | (because the pressed key was found and its number is in  RØ)
```

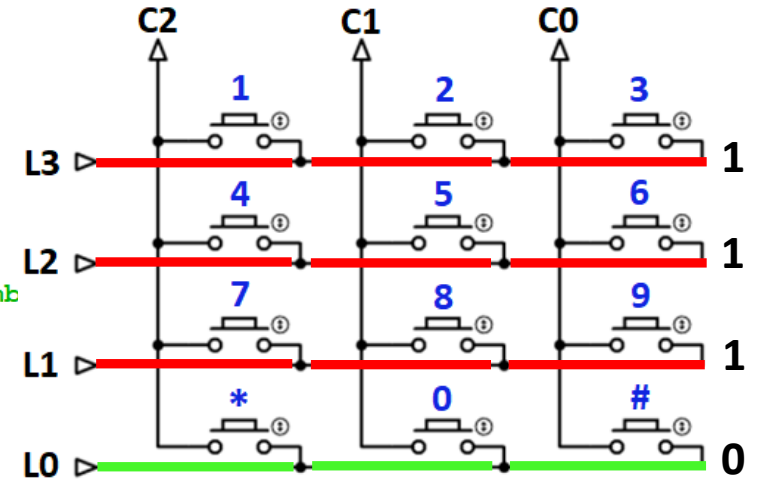


Verifica se alguma coluna possui o valor 0

```
; column-scan subroutine
colScan:
   JNB PØ.4, gotKey  ; if colØ is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.5, gotKey  ; if col1 is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.6, gotKey  ; if col2 is cleared - key found
   INC RØ            ; otherwise move to next key
   RET               ; return from subroutine - key not found
gotKey:
   SETB FØ           ; key found - set FØ
   RET               ; and return from subroutine
```

Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø        ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3       ; set row3
   CLR PØ.Ø        ; clear rowØ
   CALL colScan    ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of program
                   ; | (because the pressed key was found and its numb
   ; scan row1
   SETB PØ.Ø       ; set rowØ
   CLR PØ.1        ; clear row1
   CALL colScan    ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of program
                   ;   (because the pressed key was found and its number is in  RØ)
```




```
; column-scan subroutine
colScan:
   JNB PØ.4, gotKey  ; if colØ is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.5, gotKey  ; if col1 is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.6, gotKey  ; if col2 is cleared - key found
   INC RØ            ; otherwise move to next key
   RET               ; return from subroutine - key not found
gotKey:
   SETB FØ           ; key found - set FØ
   RET               ; and return from subroutine
```

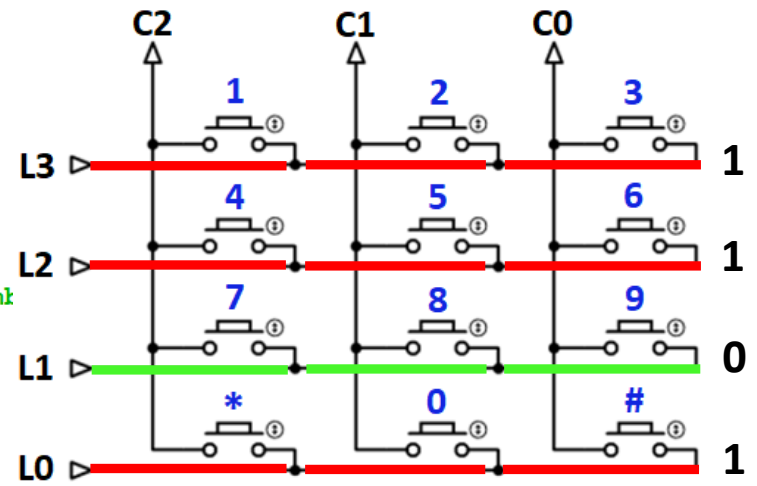
Verifica se alguma coluna possui o valor 0

Se encontrar 0 em alguma coluna, *set* F0

```
+----+----+----+
| 11 | 1Ø |  9 | row3
+----+----+----+
|  8 |  7 |  6 | row2
+----+----|----+
|  5 |  4 |  3 | row1
+----+----+----+
|  2 |  1 |  Ø | rowØ
+----+----+----+
 col2 col1 colØ
```

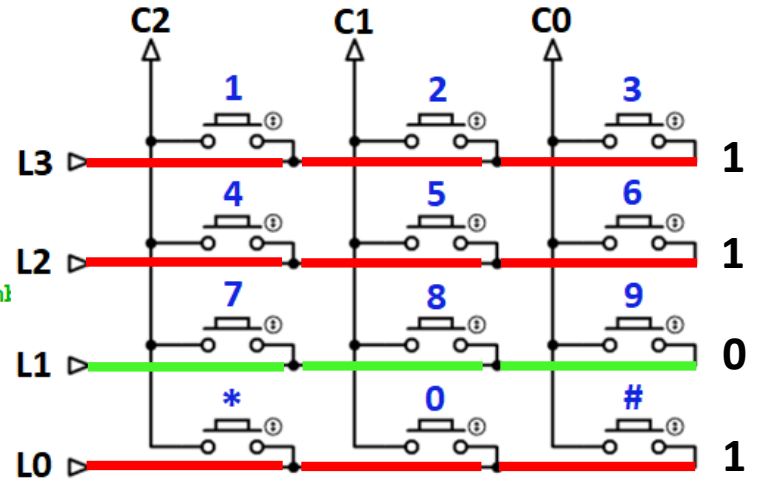
Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø          ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3           ; set row3
   CLR PØ.Ø            ; clear rowØ
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its numb
   ; scan row1
   SETB PØ.Ø           ; set rowØ
   CLR PØ.1            ; clear row1
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row2
   SETB PØ.1           ; set row1
   CLR PØ.2            ; clear row2
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row3
   SETB PØ.2           ; set row2
   CLR PØ.3            ; clear row3
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   JMP start           ; | go back to scan row Ø
                       ; | (this is why row3 is set at the start of the program
                       ; | - when the program jumps back to start, row3 has just been scanned)

finish:
   JMP $               ; program execution arrives here when key is found - do nothing
```



Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø          ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3         ; set row3
   CLR PØ.Ø          ; clear rowØ
   CALL colScan      ; call column-scan subroutine
   JB FØ, finish      ; | if FØ is set, jump to end of program
                     ; | (because the pressed key was found and its numb
   ; scan row1
   SETB PØ.Ø         ; set rowØ
   CLR PØ.1          ; clear row1
   CALL colScan      ; call column-scan subroutine
   JB FØ, finish      ; | if FØ is set, jump to end of program
                     ; | (because the pressed key was found and its number is in  RØ)
```




```
; column-scan subroutine
colScan:
   JNB PØ.4, gotKey  ; if colØ is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.5, gotKey  ; if col1 is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.6, gotKey  ; if col2 is cleared - key found
   INC RØ            ; otherwise move to next key
   RET               ; return from subroutine - key not found
gotKey:
   SETB FØ           ; key found - set FØ
   RET               ; and return from subroutine
```

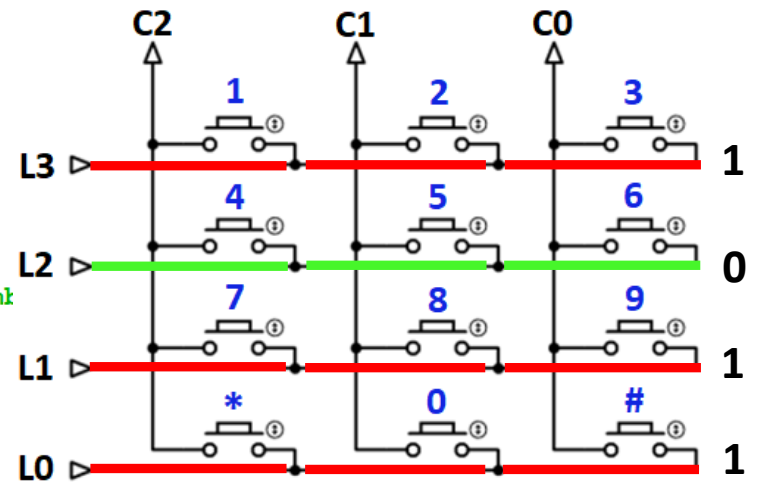
Verifica se alguma coluna possui o valor 0

Se encontrar 0 em alguma coluna, *set* F0

```
+----+----+----+
| 11 | 1Ø |  9 | row3
+----+----+----+
|  8 |  7 |  6 | row2
+----+----+----+
|  5 |  4 |  3 | row1
+----+----+----+
|  2 |  1 |  Ø | rowØ
+----+----+----+
 col2 col1 colØ
```

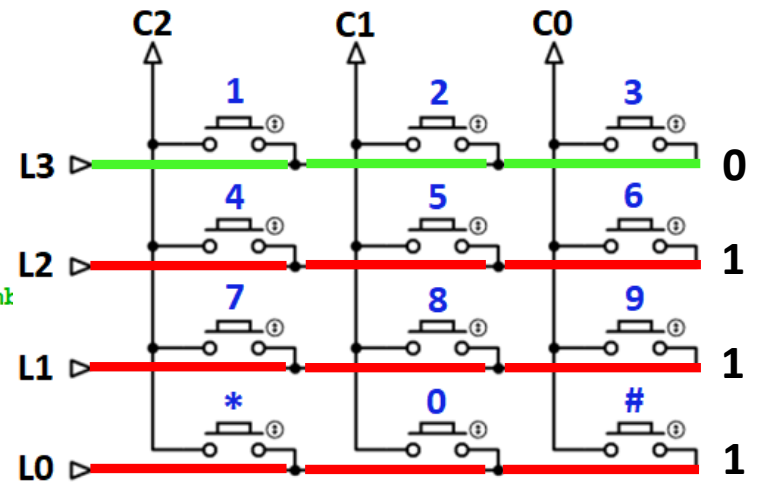
Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø          ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3           ; set row3
   CLR PØ.Ø            ; clear rowØ
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its numb
   ; scan row1
   SETB PØ.Ø           ; set rowØ
   CLR PØ.1            ; clear row1
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row2
   SETB PØ.1           ; set row1
   CLR PØ.2            ; clear row2
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row3
   SETB PØ.2           ; set row2
   CLR PØ.3            ; clear row3
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   JMP start           ; | go back to scan row Ø
                       ; | (this is why row3 is set at the start of the program
                       ; | - when the program jumps back to start, row3 has just been scanned)


finish:
   JMP $               ; program execution arrives here when key is found - do nothing
```



Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø          ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3           ; set row3
   CLR PØ.Ø            ; clear rowØ
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its numb
   ; scan row1
   SETB PØ.Ø           ; set rowØ
   CLR PØ.1            ; clear row1
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row2
   SETB PØ.1           ; set row1
   CLR PØ.2            ; clear row2
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   ; scan row3
   SETB PØ.2           ; set row2
   CLR PØ.3            ; clear row3
   CALL colScan        ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                       ; | (because the pressed key was found and its number is in  RØ)
   JMP start           ; | go back to scan row Ø
                       ; | (this is why row3 is set at the start of the program
                       ; | - when the program jumps back to start, row3 has just been scanned)

finish:
   JMP $               ; program execution arrives here when key is found - do nothing
```



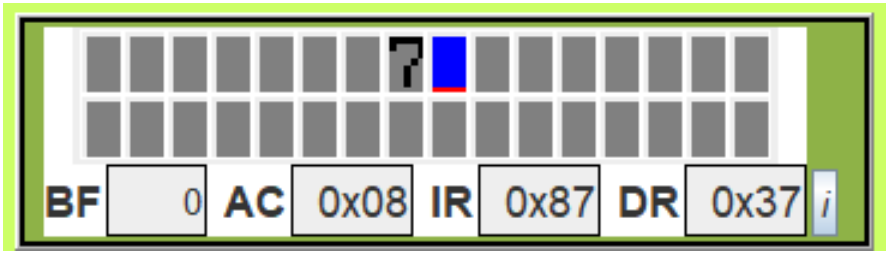
Exemplo de leitura do Teclado Matricial

```
start:
   MOV RØ, #Ø          ; clear RØ - the first key is keyØ

   ; scan rowØ
   SETB PØ.3          ; set row3
   CLR PØ.Ø           ; clear rowØ
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   ; scan row1
   SETB PØ.Ø          ; set rowØ
   CLR PØ.1           ; clear row1
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   ; scan row2
   SETB PØ.1          ; set row1
   CLR PØ.2           ; clear row2
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   ; scan row3
   SETB PØ.2          ; set row2
   CLR PØ.3           ; clear row3
   CALL colScan       ; call column-scan subroutine
   JB FØ, finish       ; | if FØ is set, jump to end of program
                      ; | (because the pressed key was found and its number is in  RØ)
   JMP start          ; | go back to scan row Ø
                      ; | (this is why row3 is set at the start of the program
                      ; | - when the program jumps back to start, row3 has just been scanned)

finish:
   JMP $              ; program execution arrives here when key is found - do nothing
```

Exemplo 02

Exemplo de leitura do Teclado Matricial e escrita no LCD

Este exemplo realiza a leitura do Teclado matricial e escreve no Display LCD a respectiva tecla que foi pressionada no teclado matricial.

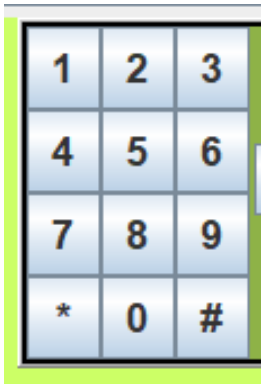


Exemplo de leitura do Teclado Matricial e escrita no LCD

Para facilitar a escrita dos caracteres no display LCD, devemos armazenar na memória os caracteres do Teclado, na sequência dos números que serão escritos no R0, conforme exemplo anterior.

```
org ØØØØh
   LJMP START

org ØØ3Øh
START:
; put data in RAM
   MOV 4ØH, #'#'
   MOV 41H, #'Ø'
   MOV 42H, #'*'
   MOV 43H, #'9'
   MOV 44H, #'8'
   MOV 45H, #'7'
   MOV 46H, #'6'
   MOV 47H, #'5'
   MOV 48H, #'4'
   MOV 49H, #'3'
   MOV 4AH, #'2'
   MOV 4BH, #'1'
```

```
+----+----+----+
| 11 | 1Ø |  9 | row3
+----+----+----+
|  8 |  7 |  6 | row2
+----+----|----+
|  5 |  4 |  3 | row1
+----+----+----+
|  2 |  1 |  Ø | rowØ
+----+----+----+
 col2 col1 colØ
```



Exemplo de leitura do Teclado Matricial e escrita no LCD

Para verificar se uma tecla do Teclado matricial foi pressionada, criaremos duas sub-rotinas (**leituraTeclado**, **colScan**), conforme exemplo anterior.

```
leituraTeclado:
   MOV RØ, #Ø        ; clear RØ - the first key is keyØ

   ; scan rowØ
   MOV PØ, #ØFFh
   CLR PØ.Ø          ; clear rowØ
   CALL colScan      ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of progr
                     ; | (because the pressed key was found and
   ; scan row1
   SETB PØ.Ø         ; set rowØ
   CLR PØ.1          ; clear row1
   CALL colScan      ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of progr
                     ; | (because the pressed key was found and
   ; scan row2
   SETB PØ.1         ; set row1
   CLR PØ.2          ; clear row2
   CALL colScan      ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of progr
                     ; | (because the pressed key was found and
   ; scan row3
   SETB PØ.2         ; set row2
   CLR PØ.3          ; clear row3
   CALL colScan      ; call column-scan subroutine
   JB FØ, finish     ; | if FØ is set, jump to end of progr
                     ; | (because the pressed key was found and
finish:
   RET
```

```
; column-scan subroutine
colScan:
   JNB PØ.4, gotKey  ; if colØ is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.5, gotKey  ; if col1 is cleared - key found
   INC RØ            ; otherwise move to next key
   JNB PØ.6, gotKey  ; if col2 is cleared - key found
   INC RØ            ; otherwise move to next key
   RET               ; return from subroutine - key not found
gotKey:
   SETB FØ           ; key found - set FØ
   RET               ; and return from subroutine
```

Exemplo de leitura do Teclado Matricial e escrita no LCD

Agora, nosso programa principal chamado MAIN, que chama as sub-rotinas de leitura do teclado e escrita no display LCD.

```
MAIN:
   ACALL lcd_init
ROTINA:
   ACALL leituraTeclado
   JNB FØ, ROTINA    ;if FØ is clear, jump to ROTINA
   MOV A, #Ø7h
   ACALL posicionaCursor
   MOV A, #4Øh
   ADD A, RØ
   MOV RØ, A
   MOV A, @RØ
   ACALL sendCharacter
   CLR FØ
   JMP ROTINA
```

Exemplo de leitura do Teclado Matricial e escrita no LCD

Primeiramente, devemos chamar a sub-rotina lcd_init para configurar nosso display LCD.

```
MAIN:
   ACALL lcd_init
ROTINA:
   ACALL leituraTeclado
   JNB FØ, ROTINA    ;if FØ is clear, jump to ROTINA
   MOV A, #Ø7h
   ACALL posicionaCursor
   MOV A, #4Øh
   ADD A, RØ
   MOV RØ, A
   MOV A, @RØ
   ACALL sendCharacter
   CLR FØ
   JMP ROTINA
```

Exemplo de leitura do Teclado Matricial e escrita no LCD

Agora, executamos a sub-rotina leituraTeclado para verificar se alguma tecla foi pressionada.

$$F0 = \begin{cases} \text{Uma tecla foi pressionada} & \text{if } F0 \text{ is } 1 \\ \text{Nenhuma tecla foi pressionada} & \text{if } F0 \text{ is } 0 \end{cases}$$

```
MAIN:
   ACALL lcd_init
ROTINA:
   ACALL leituraTeclado
   JNB FØ, ROTINA    ;if FØ is clear, jump to ROTINA
   MOV A, #Ø7h
   ACALL posicionaCursor
   MOV A, #4Øh
   ADD A, RØ
   MOV RØ, A
   MOV A, @RØ
   ACALL sendCharacter
   CLR FØ
   JMP ROTINA
```

Exemplo de leitura do Teclado Matricial e escrita no LCD

Se uma tecla foi pressionada, o programa abaixo será executado para escrever no Display LCD a tecla que foi pressionada.

```
MAIN:
   ACALL lcd_init
ROTINA:
   ACALL leituraTeclado
   JNB FØ, ROTINA    ;if FØ is clear, jump to ROTINA
   MOV A, #Ø7h
   ACALL posicionaCursor
   MOV A, #4Øh
   ADD A, RØ
   MOV RØ, A
   MOV A, @RØ
   ACALL sendCharacter
   CLR FØ
   JMP ROTINA
```

Exemplo de leitura do Teclado Matricial e escrita no LCD

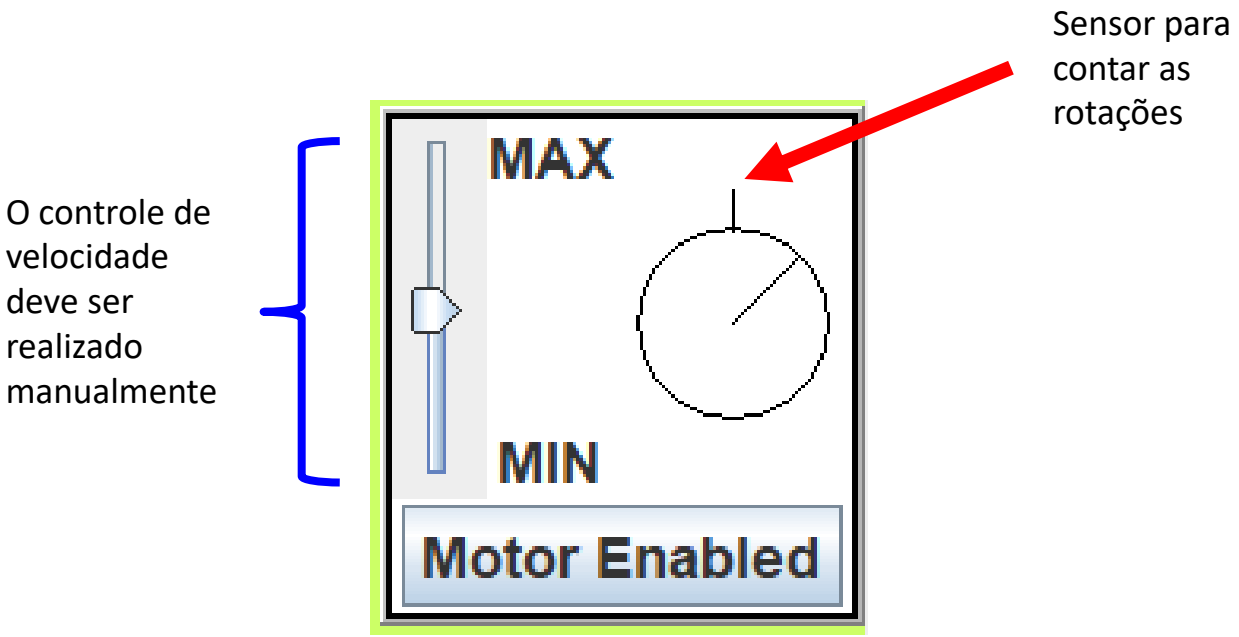
No final, o programa volta para o *Label* ROTINA para executar novamente a leitura do teclado.

```
MAIN:
   ACALL lcd_init
ROTINA:
   ACALL leituraTeclado
   JNB FØ, ROTINA    ;if FØ is clear, jump to ROTINA
   MOV A, #Ø7h
   ACALL posicionaCursor
   MOV A, #4Øh
   ADD A, RØ
   MOV RØ, A
   MOV A, @RØ
   ACALL sendCharacter
   CLR FØ
   JMP ROTINA
```

Motor no Edsim51

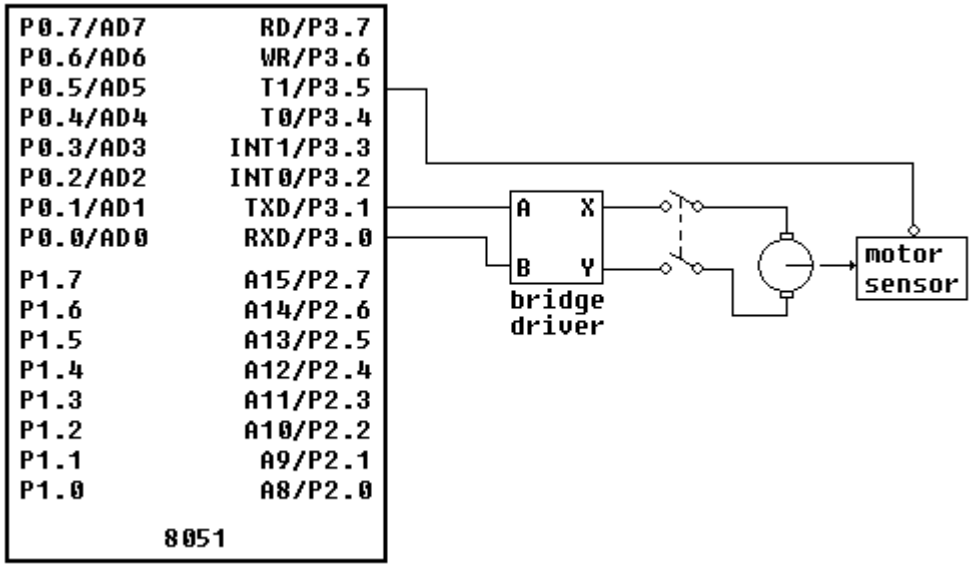
Motor no edSim51

No Edsim51 o motor possui um funcionamento simples, podendo apenas controlar a direção do motor (sentido horário ou anti-horário), e realizar a leitura em um sensor para contar as rotações.



Ligação do Motor no 8051 – edSim51

No edSim51 o motor está ligado nas portas P3.0 e P3.1 e P3.5, onde as portas P3.0 e P3.1 são para controlar o motor no sentido horário, anti-horário e parado. E no pino P3.5 podemos realizar a leitura do sensor.



A	B	motor
0	0	stop
0	1	forward
1	0	reverse
1	1	stop

Programa para controlar o Motor– edSim51

Para ligar o motor, precisa somente colocar P3.0=1 e P3.1=0 ou P3.0=0 e P3.1=1.

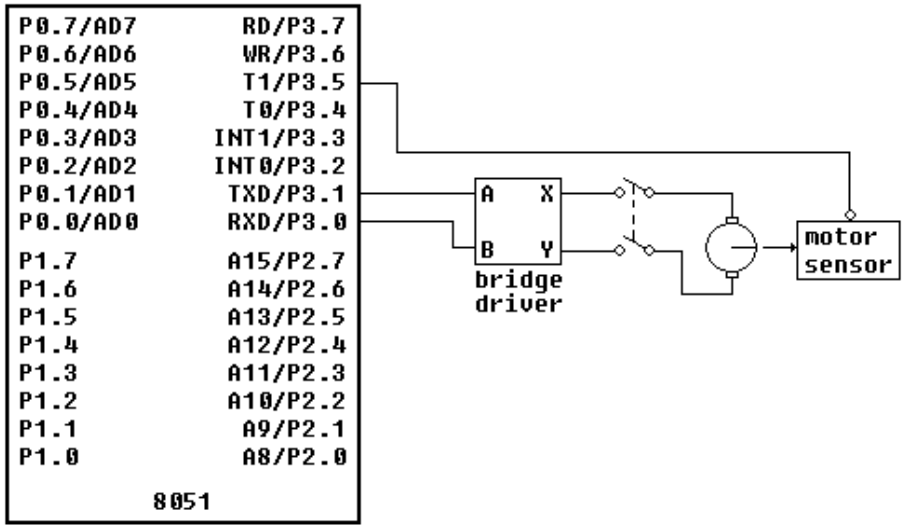
```
;gira o motor no sentido horário
SETB P3.Ø
CLR P3.1
```

```
;gira o motor no sentido anti-horário
CLR P3.Ø
SETB P3.1
```

Programa para controlar o Motor– edSim51

Para realizar a leitura das rotações do motor, podemos utilizar o contador CT1, realizando a contagem no pino P3.5, que é o pino T1 da contagem externa do contador 1.

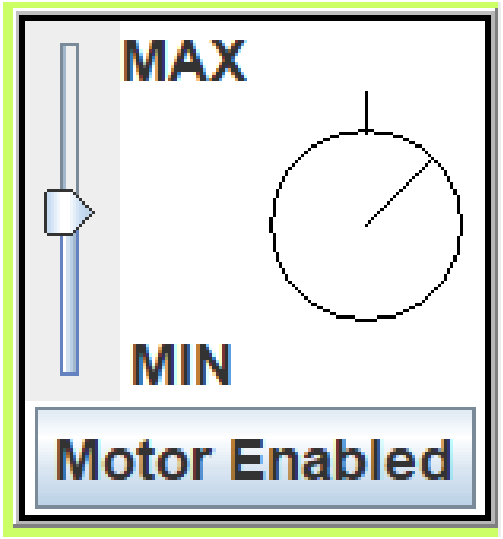
```
MOV TMOD, #Ø1Ø1ØØØØb    ;Habilita o contador
SETB TR1                ;liga o CT1
```



Exemplo

Exemplo com o Motor

Crie um programa que faça que ligue o motor, e a cada 255 voltas o motor deve inverter a rotação.



Exemplo com o Motor

Solução:

```
org ØØØØh
   LJMP MAIN

org ØØ1Bh
intØ1:
   CPL P3.Ø    ;|
   CPL P3.1    ;| Inverte o sentido de rotação do motor
   RETI

org ØØ3Øh
MAIN:
   MOV TMOD, #Ø11ØØØØØb   ;Contador CT1 no modo 2
   SETB EA                ;Habilita as interrupções
   SETB ET1               ;Habilita a interrupção CT1
   SETB TR1               ;Liga o contador 1
   SETB P3.Ø              ;|
   CLR  P3.1              ;| Liga motor no sentido horário
   JMP $
```

Exemplo LCD

Exemplo

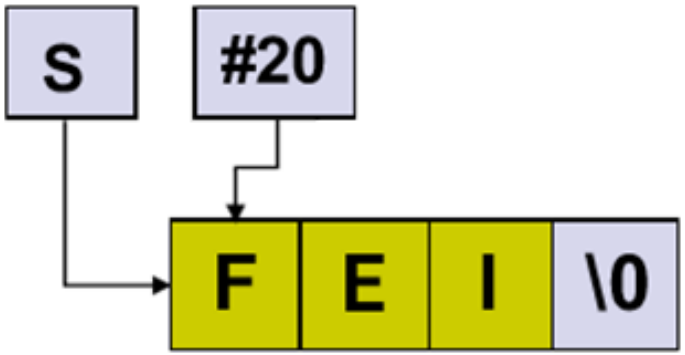
Escreva uma rotina que faça o 8051 escrever no Display LCD a palavra FEI centralizada na primeira linha e a palavra Display LCD centralizada na segunda linha.

Observação: Use as sub-rotinas **lcd_init, sendCharacter, posicionaCursor, clearDisplay**.

Exemplo

Solução:

Abaixo temos a String FEI escrita na memória do Programa.




```
; Escrevendo na memória de programa
FEI:
   DB "FEI"
   DB 0                    ;caracter null indica fim da String
Display:
   DB "Display LCD"
   DB 0                    ;caracter null indica fim da String
```

Exemplo

Solução:

Agora criaremos uma subrotina para escrever a String no LCD.

```
escreveString:
   MOV R2, #0
rot:
   MOV A, R2
   MOVC A,@A+DPTR          ;lê a tabela da memória de programa
   ACALL sendCharacter     ; send data in A to LCD module
   INC R2
   JNZ rot              ; if A is 0, then end of data has been reached - jump out of loop
   RET
```

Exemplo

Solução:

Agora no main usaremos as sub-rotinas para escrever as Strings no LCD.

```
main:
   ACALL lcd_init
   MOV  A,  #06h
   ACALL  posicionaCursor
   MOV  DPTR,#FEI                ;DPTR = início da palavra FEI
   ACALL  escreveString
   MOV  A,  #42h
   ACALL  posicionaCursor
   MOV  DPTR,#Display            ;DPTR = início da palavra Display
   ACALL  escreveString
   JMP   $
```

Bibliografia

ZELENOVSKY, R.; MENDONÇA, A. Microcontroladores Programação e Projeto com a Família 8051. MZ Editora, RJ, 2005.

Gimenez, Salvador P. Microcontroladores 8051 - Teoria e Prática, Editora Érica, 2010.