

Arquitetura de Computadores

PROF. ISAAC

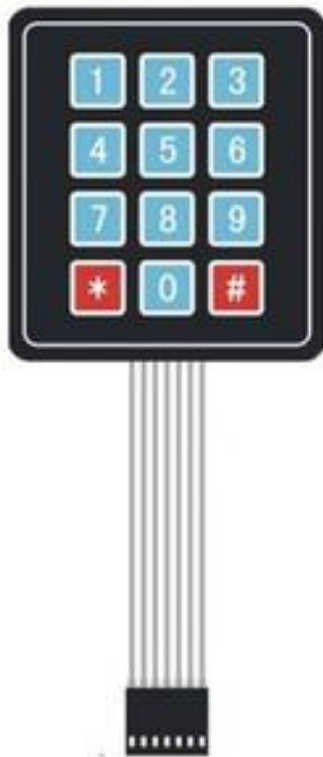
Teclado Matricial

Teclado Matricial



Teclado Matricial

4x3



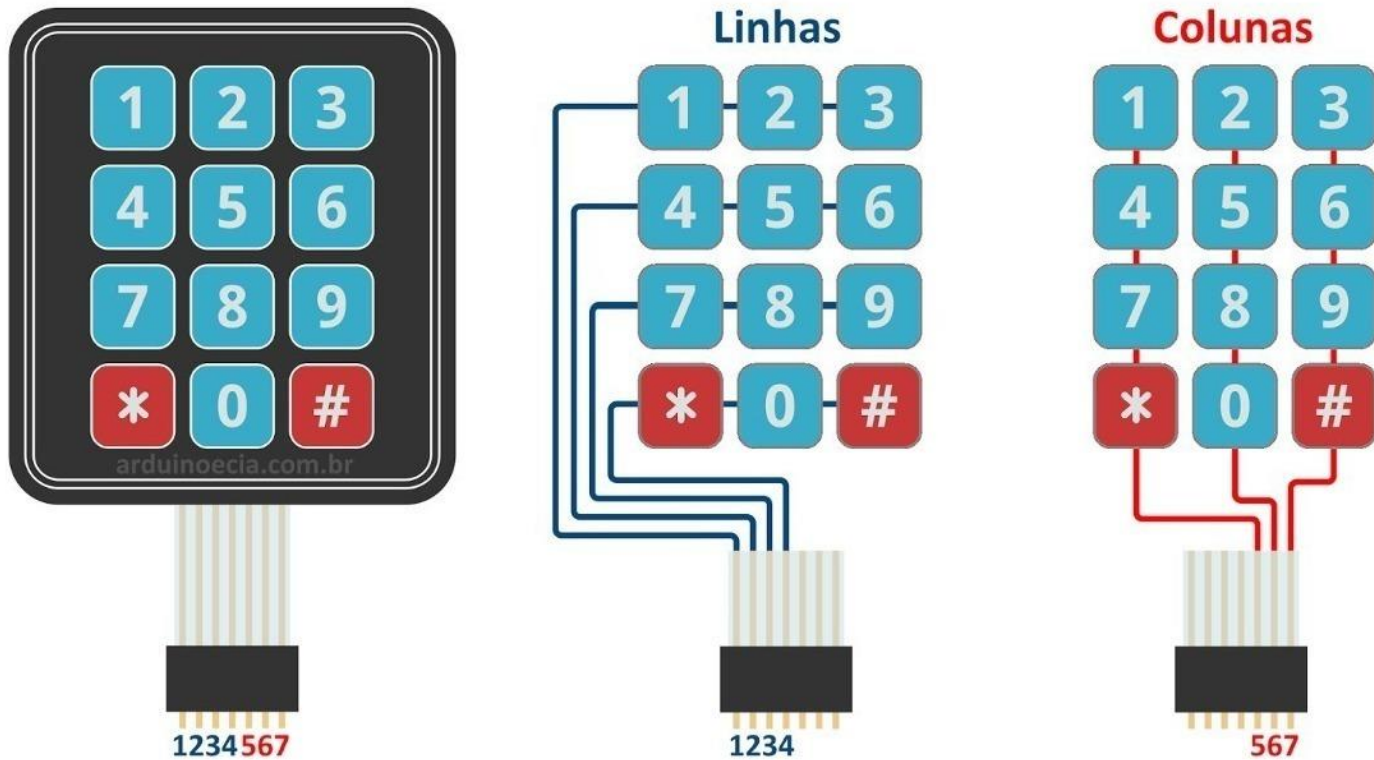
4x4



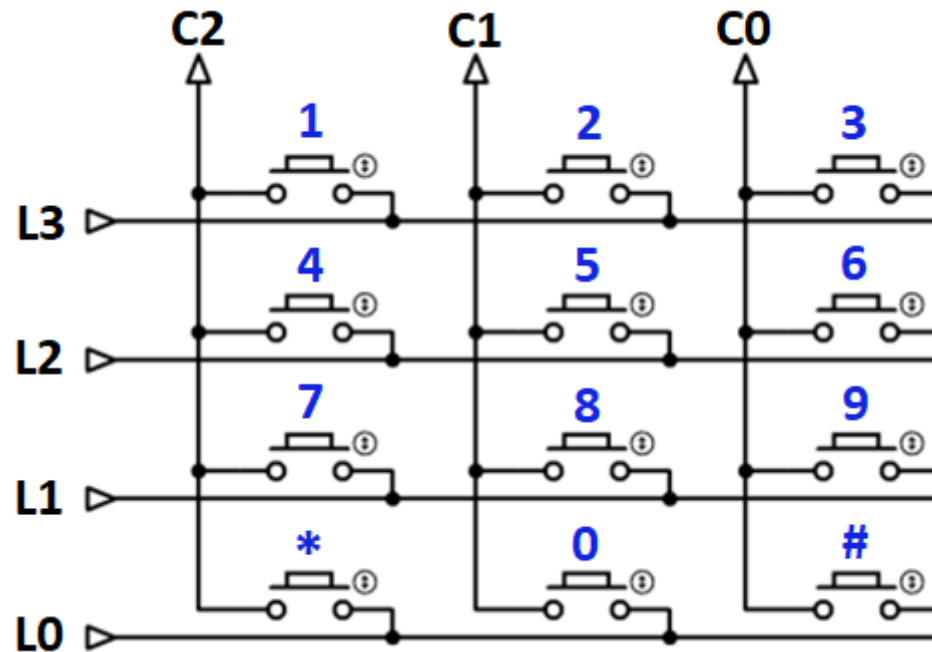
5x4



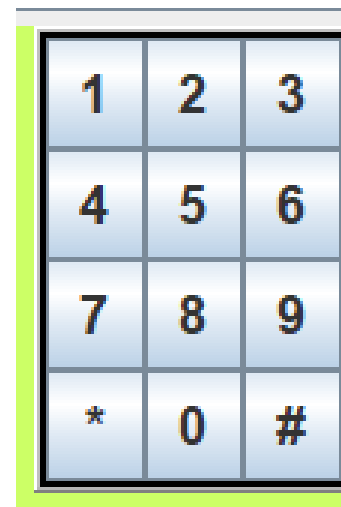
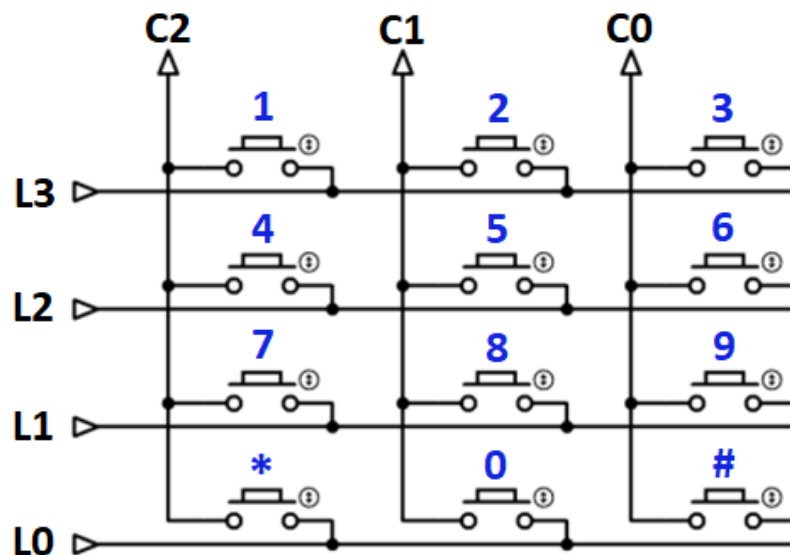
Teclado Matricial



Teclado Matricial



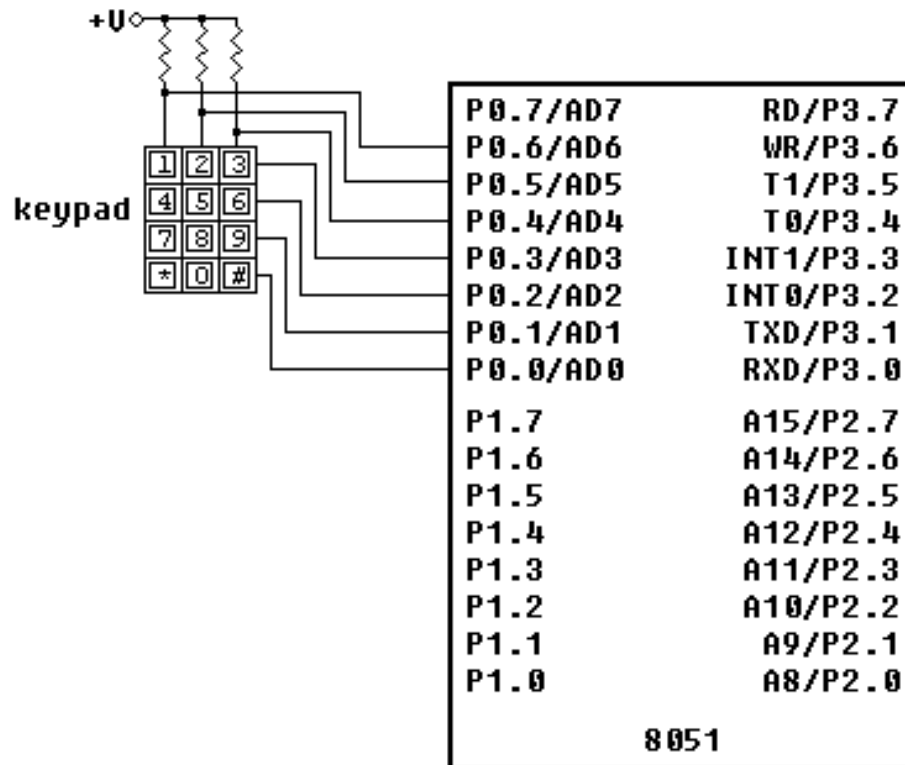
Teclado Matricial no EdSim51



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 | 10 | 9 | row3
+---+---+---+
| 8 | 7 | 6 | row2
+---+---+---+
| 5 | 4 | 3 | row1
+---+---+---+
| 2 | 1 | 0 | row0
+---+---+---+
      col2 col1 col0

```

1	2	3
4	5	6
7	8	9
*	0	#

Exemplo de leitura do Teclado Matricial

```
start:
    MOV R0, #0          ; clear R0 - the first key is key0

    ; scan row0
    SETB P0.3           ; set row3
    CLR P0.0            ; clear row0
    CALL colScan        ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

    ; scan row1
    SETB P0.0           ; set row0
    CLR P0.1            ; clear row1
    CALL colScan        ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

    ; scan row2
    SETB P0.1           ; set row1
    CLR P0.2            ; clear row2
    CALL colScan        ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

    ; scan row3
    SETB P0.2           ; set row2
    CLR P0.3            ; clear row3
    CALL colScan        ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

    JMP start           ; | go back to scan row 0
                        ; | (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 P0.4, gotKey ; if col0 is cleared - key found
    INC R0           ; otherwise move to next key
    JNB P0.5, gotKey ; if col1 is cleared - key found
    INC R0           ; otherwise move to next key
    JNB P0.6, gotKey ; if col2 is cleared - key found
    INC R0           ; otherwise move to next key
    RET             ; return from subroutine - key not found
gotKey:
    SETB F0         ; key found - set F0
    RET             ; and return from subroutine
```

Exemplo de leitura do Teclado Matricial

```

start:
    MOV R0, #0      ; clear R0 - the first key is key0

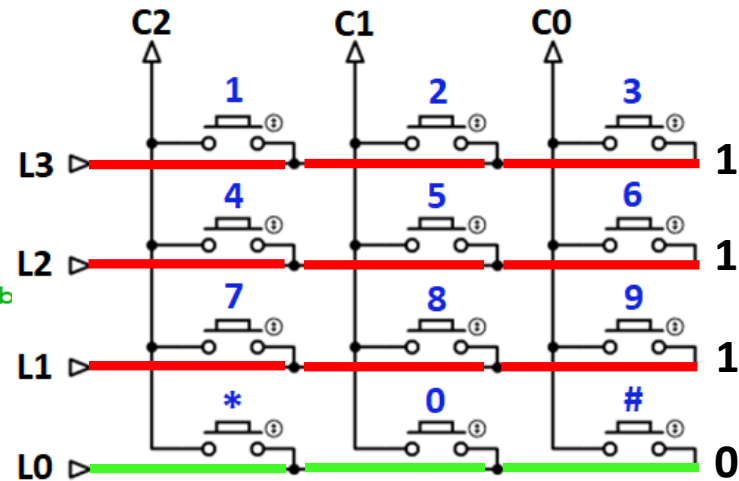
    ; scan row0
    SETB P0.3      ; set row3
    CLR P0.0       ; clear row0
    CALL colScan   ; call column-scan subroutine
    JB F0, finish  ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its numb
    ; scan row1
    SETB P0.0      ; set row0
    CLR P0.1       ; clear row1
    CALL colScan   ; call column-scan subroutine
    JB F0, finish  ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row2
    SETB P0.1      ; set row1
    CLR P0.2       ; clear row2
    CALL colScan   ; call column-scan subroutine
    JB F0, finish  ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row3
    SETB P0.2      ; set row2
    CLR P0.3       ; clear row3
    CALL colScan   ; call column-scan subroutine
    JB F0, finish  ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    JMP start      ; | go back to scan row 0
                    ; | (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

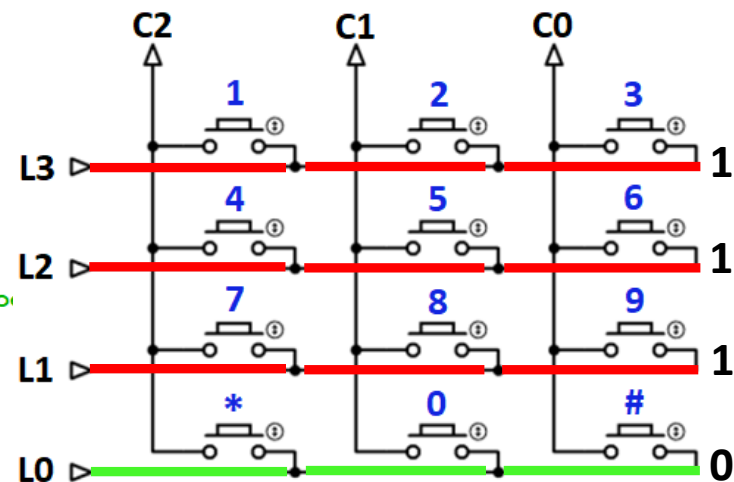
start:

```

MOV R0, #0          ; clear R0 - the first key is key0

; scan row0
SETB P0.3           ; set row3
CLR P0.0            ; clear row0
CALL colScan        ; call column-scan subroutine
JB F0, finish       ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number
                    ; | is in R0)

; scan row1
SETB P0.0           ; set row0
CLR P0.1            ; clear row1
CALL colScan        ; call column-scan subroutine
JB F0, finish       ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)
    
```



Verifica se
alguma coluna
possui o valor 0

; column-scan subroutine

colScan:

```

JNB P0.4, gotKey ; if col0 is cleared - key found
INC R0           ; otherwise move to next key
JNB P0.5, gotKey ; if col1 is cleared - key found
INC R0           ; otherwise move to next key
JNB P0.6, gotKey ; if col2 is cleared - key found
INC R0           ; otherwise move to next key
RET              ; return from subroutine - key not found
    
```

gotKey:

```

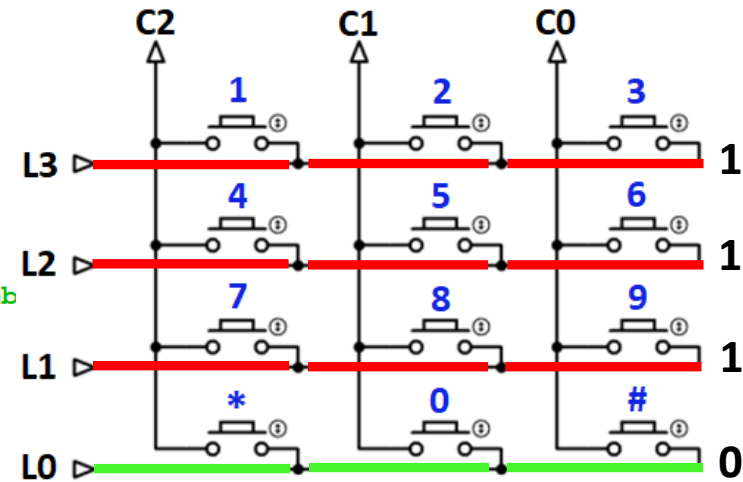
SETB F0          ; key found - set F0
RET              ; and return from subroutine
    
```

Exemplo de leitura do Teclado Matricial

```

start:
  MOV R0, #0      ; clear R0 - the first key is key0

  ; scan row0
  SETB P0.3      ; set row3
  CLR P0.0       ; clear row0
  CALL colScan   ; call column-scan subroutine
  JB F0, finish  ; | if F0 is set, jump to end of program
                  ; | (because the pressed key was found and its numb
                  ; |
  ; scan row1
  SETB P0.0      ; set row0
  CLR P0.1       ; clear row1
  CALL colScan   ; call column-scan subroutine
  JB F0, finish  ; | if F0 is set, jump to end of program
                  ; | (because the pressed key was found and its number is in R0)
  
```



; column-scan subroutine

colScan:

JNB P0.4, gotKey ; if col0 is cleared - key found

INC R0 ; otherwise move to next key

JNB P0.5, gotKey ; if col1 is cleared - key found

INC R0 ; otherwise move to next key

JNB P0.6, gotKey ; if col2 is cleared - key found

INC R0 ; otherwise move to next key

RET ; return from subroutine - key not found

gotKey:

SETB F0 ; key found - set F0

RET ; and return from subroutine

+-----+-----+-----+				
11 10 9	row3			
+-----+-----+-----+				
8 7 6	row2			
+-----+-----+-----+				
5 4 3	row1			
+-----+-----+-----+				
2 1 0	row0			
+-----+-----+-----+				
col2 col1 col0				

Verifica se
alguma coluna
possui o valor 0

Se encontrar 0
em alguma
coluna, set F0

Exemplo de leitura do Teclado Matricial

```

start:
    MOV R0, #0      ; clear R0 - the first key is key0

    ; scan row0
    SETB P0.3       ; set row3
    CLR P0.0        ; clear row0
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its numk

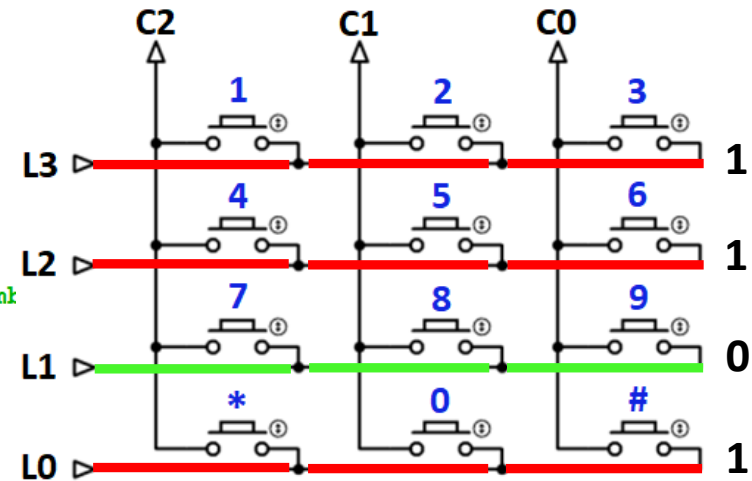
    ; scan row1
    SETB P0.0       ; set row0
    CLR P0.1        ; clear row1
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row2
    SETB P0.1       ; set row1
    CLR P0.2        ; clear row2
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row3
    SETB P0.2       ; set row2
    CLR P0.3        ; clear row3
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    JMP start       ; | go back to scan row 0
                    ; | (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
    
```



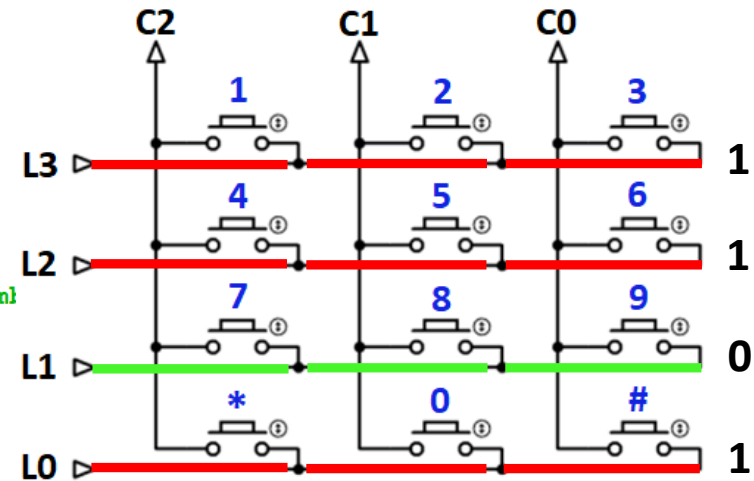
Exemplo de leitura do Teclado Matricial

```

start:
  MOV R0, #0      ; clear R0 - the first key is key0

  ; scan row0
  SETB P0.3      ; set row3
  CLR P0.0       ; clear row0
  CALL colScan   ; call column-scan subroutine
  JB F0, finish  ; | if F0 is set, jump to end of program
                  ; | (because the pressed key was found and its num)

  ; scan row1
  SETB P0.0      ; set row0
  CLR P0.1       ; clear row1
  CALL colScan   ; call column-scan subroutine
  JB F0, finish  ; | if F0 is set, jump to end of program
                  ; | (because the pressed key was found and its number is in R0)
  
```



; column-scan subroutine

colScan:

JNB P0.4, gotKey ; if col0 is cleared - key found

INC R0 ; otherwise move to next key

JNB P0.5, gotKey ; if col1 is cleared - key found

INC R0 ; otherwise move to next key

JNB P0.6, gotKey ; if col2 is cleared - key found

INC R0 ; otherwise move to next key

RET

; return from subroutine - key not found

gotKey:

SETB F0 ; key found - set F0

RET ; and return from subroutine

+-----+-----+-----+				
11 10 9	row3			
+-----+-----+-----+				
8 7 6	row2			
+-----+-----+-----+				
5 4 3	row1			
+-----+-----+-----+				
2 1 0	row0			
+-----+-----+-----+				
col2 col1 col0				

Verifica se
alguma coluna
possui o valor 0

Se encontrar 0
em alguma
coluna, set F0

Exemplo de leitura do Teclado Matricial

```

start:
    MOV R0, #0      ; clear R0 - the first key is key0

    ; scan row0
    SETB P0.3       ; set row3
    CLR P0.0        ; clear row0
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its numk

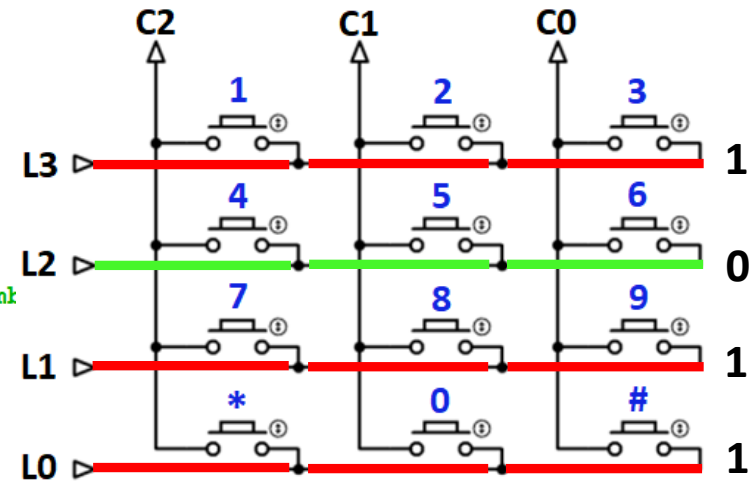
    ; scan row1
    SETB P0.0       ; set row0
    CLR P0.1        ; clear row1
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row2
    SETB P0.1       ; set row1
    CLR P0.2        ; clear row2
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row3
    SETB P0.2       ; set row2
    CLR P0.3        ; clear row3
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    JMP start       ; | go back to scan row 0
                    ; | (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 R0, #0      ; clear R0 - the first key is key0

    ; scan row0
    SETB P0.3       ; set row3
    CLR P0.0        ; clear row0
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its numk

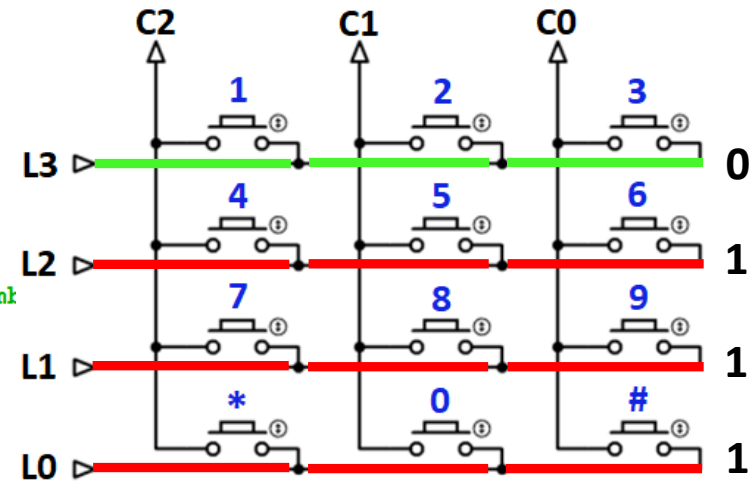
    ; scan row1
    SETB P0.0       ; set row0
    CLR P0.1        ; clear row1
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row2
    SETB P0.1       ; set row1
    CLR P0.2        ; clear row2
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    ; scan row3
    SETB P0.2       ; set row2
    CLR P0.3        ; clear row3
    CALL colScan    ; call column-scan subroutine
    JB F0, finish   ; | if F0 is set, jump to end of program
                    ; | (because the pressed key was found and its number is in R0)

    JMP start       ; | go back to scan row 0
                    ; | (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 R0, #0          ; clear R0 - the first key is key0

    ; scan row0
    SETB P0.3           ; set row3
    CLR P0.0            ; clear row0
    CALL colScan         ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

    ; scan row1
    SETB P0.0           ; set row0
    CLR P0.1            ; clear row1
    CALL colScan         ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

    ; scan row2
    SETB P0.1           ; set row1
    CLR P0.2            ; clear row2
    CALL colScan         ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

    ; scan row3
    SETB P0.2           ; set row2
    CLR P0.3            ; clear row3
    CALL colScan         ; call column-scan subroutine
    JB F0, finish       ; | if F0 is set, jump to end of program
                        ; | (because the pressed key was found and its number is in R0)

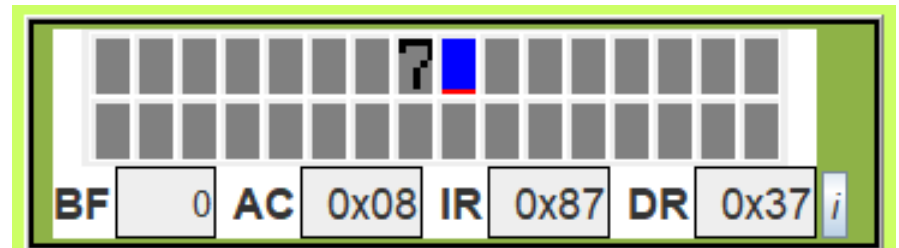
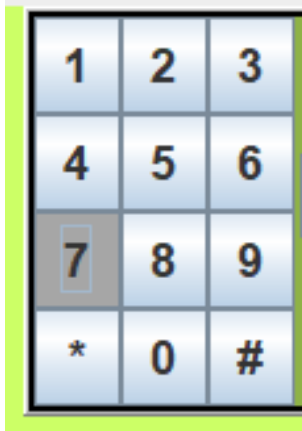
    JMP start           ; | go back to scan row 0
                        ; | (this is why row3 is set at the start of the program
                        ; | - when the program jumps back to start, row3 has just been scanned)

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.



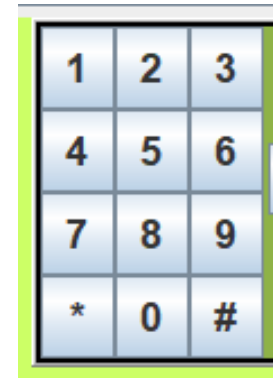
Exemplo de leitura do Teclado Matricial e escrita no LCD

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 0000h
    LJMP START

org 0030h
START:
; put data in RAM
    MOV 40H, #'#'
    MOV 41H, #'0'
    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	10	9	row3
8	7	6	row2
5	4	3	row1
2	1	0	row0
col2	col1	col0	



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 R0, #0          ; clear R0 - the first key is key0

; scan row0
MOV P0, #0FFh
CLR P0.0            ; clear row0
CALL colScan        ; call column-scan subroutine
JB F0, finish       ; | if F0 is set, jump to end of progr
                    ; | (because the pressed key was found and

; scan row1
SETB P0.0           ; set row0
CLR P0.1            ; clear row1
CALL colScan        ; call column-scan subroutine
JB F0, finish       ; | if F0 is set, jump to end of progr
                    ; | (because the pressed key was found and

; scan row2
SETB P0.1           ; set row1
CLR P0.2            ; clear row2
CALL colScan        ; call column-scan subroutine
JB F0, finish       ; | if F0 is set, jump to end of progr
                    ; | (because the pressed key was found and

; scan row3
SETB P0.2           ; set row2
CLR P0.3            ; clear row3
CALL colScan        ; call column-scan subroutine
JB F0, finish       ; | if F0 is set, jump to end of progr
                    ; | (because the pressed key was found and
```

finish:

```
RET
```

; column-scan subroutine

colScan:

```
JNB P0.4, gotKey    ; if col0 is cleared - key found
INC R0              ; otherwise move to next key
JNB P0.5, gotKey    ; if col1 is cleared - key found
INC R0              ; otherwise move to next key
JNB P0.6, gotKey    ; if col2 is cleared - key found
INC R0              ; otherwise move to next key
RET                 ; return from subroutine - key not found
```

gotKey:

```
SETB F0             ; key found - set F0
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 F0, ROTINA    ;if F0 is clear, jump to ROTINA
    MOV A, #07h
    ACALL posicionaCursor
    MOV A, #40h
    ADD A, R0
    MOV R0, A
    MOV A, @R0
    ACALL sendCharacter
    CLR F0
    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 F0, ROTINA ;if F0 is clear, jump to ROTINA
```

```
MOV A, #07h
```

```
ACALL posicionaCursor
```

```
MOV A, #40h
```

```
ADD A, R0
```

```
MOV R0, A
```

```
MOV A, @R0
```

```
ACALL sendCharacter
```

```
CLR F0
```

```
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 F0, ROTINA ;if F0 is clear, jump to ROTINA

MOV A, #07h

ACALL posicionaCursor

MOV A, #40h

ADD A, R0

MOV R0, A

MOV A, @R0

ACALL sendCharacter

CLR F0

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 F0, ROTINA    ;if F0 is clear, jump to ROTINA
    MOV A, #07h
    ACALL posicionaCursor
    MOV A, #40h
    ADD A, R0
    MOV R0, A
    MOV A, @R0
    ACALL sendCharacter
    CLR F0
    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.

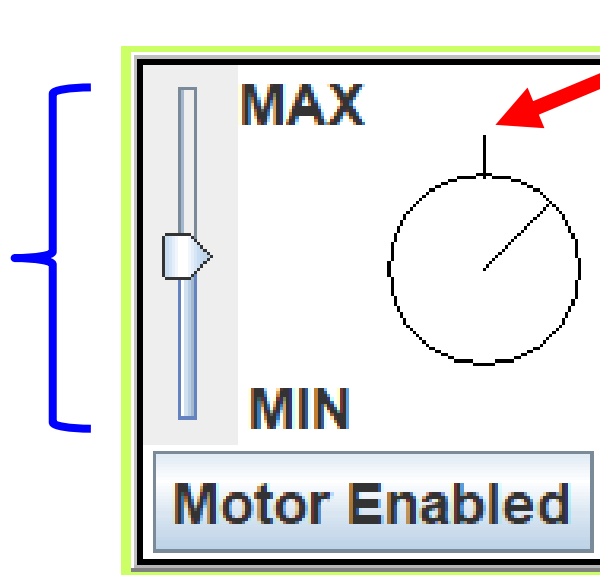
```
MAIN:
    ACALL lcd_init
ROTINA:
    ACALL leituraTeclado
    JNB F0, ROTINA    ;if F0 is clear, jump to ROTINA
    MOV A, #07h
    ACALL posicionaCursor
    MOV A, #40h
    ADD A, R0
    MOV R0, A
    MOV A, @R0
    ACALL sendCharacter
    CLR F0
    JMP ROTINA
```

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.

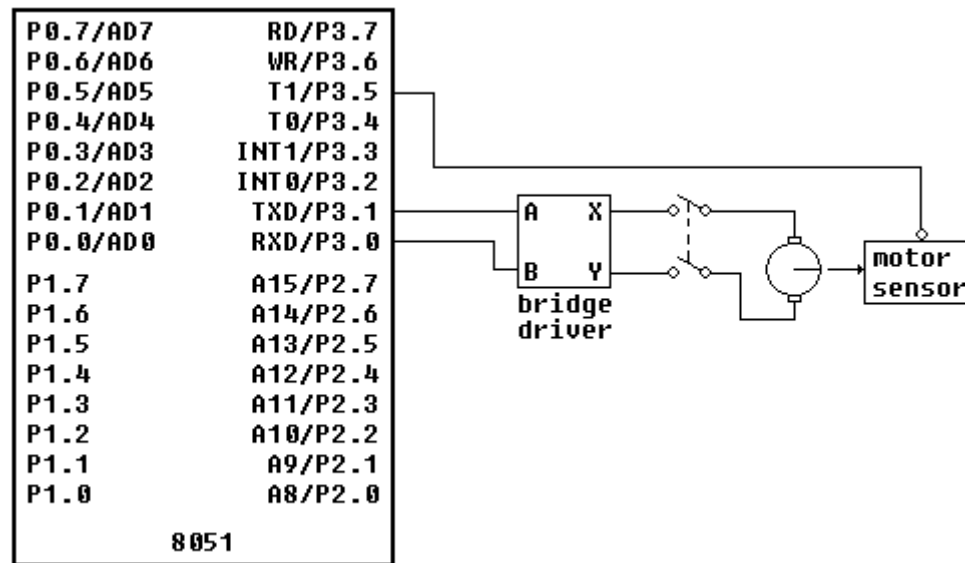
O controle de velocidade deve ser realizado manualmente



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.



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**.

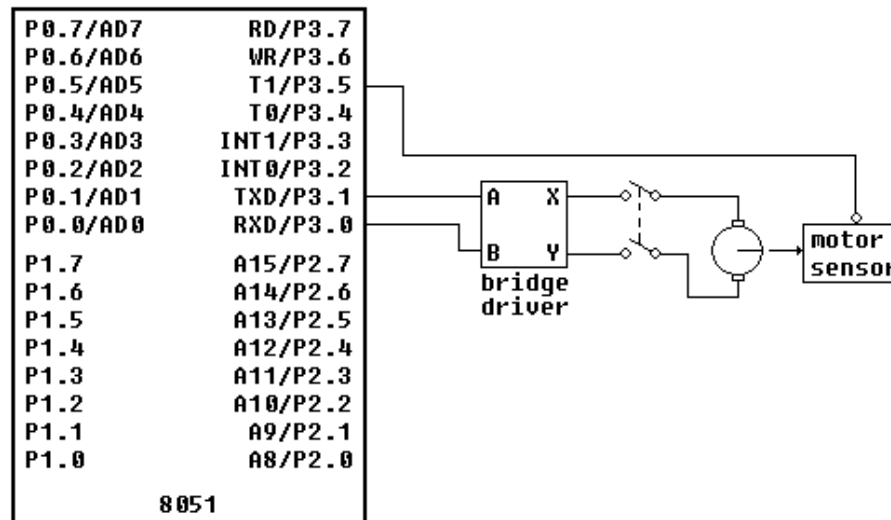
```
;gira o motor no sentido horário  
SETB P3.0  
CLR P3.1
```

```
;gira o motor no sentido anti-horário  
CLR P3.0  
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.

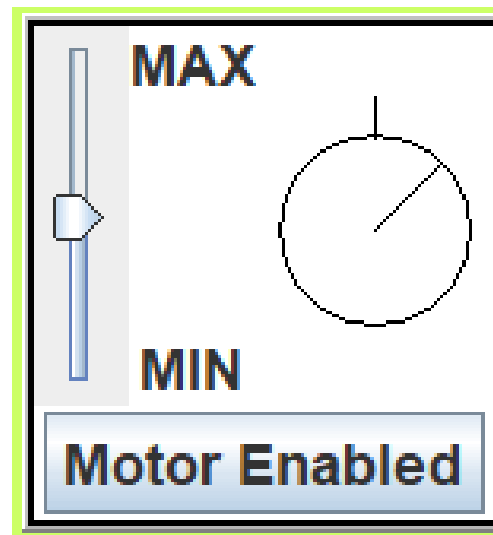
```
MOV TMOD, #01010000b ;Habilita o contador
SETB TR1               ;liga o CT1
```



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 0000h
    LJMP MAIN

org 001Bh
int01:
    CPL P3.0    ;|
    CPL P3.1    ;| Inverte o sentido de rotação do motor
    RETI

org 0030h
MAIN:
    MOV TMOD, #01100000b ;Contador CT1 no modo 2
    SETB EA          ;Habilita as interrupções
    SETB ET1         ;Habilita a interrupção CT1
    SETB TR1         ;Liga o contador 1
    SETB P3.0        ;|
    CLR P3.1         ;| Liga motor no sentido horário
    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.