

# **ARDUINO & TINKERCAD**

**Simulação de HW/SW do ARDUINO**  
**[www.tinkercad.com](http://www.tinkercad.com)**

---

Prof. Cláudio

2017

# ARDUINO + LCD NO TINKERCAD

---

## ✖ LCD

- + Descrição
- + LCD's Textuais

*to tinker* = brincar  
remendar, consertar

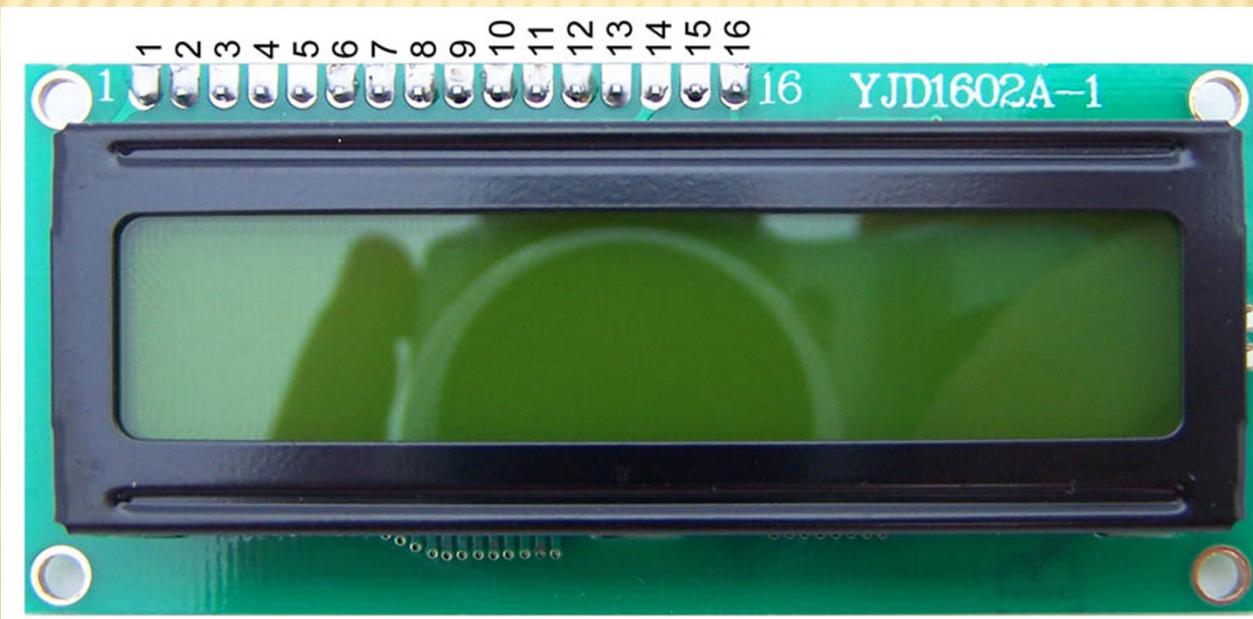
# LCD's Textuais

Compatíveis com o Controlador Hitachi HD44780



# LCD'S DE TEXTO

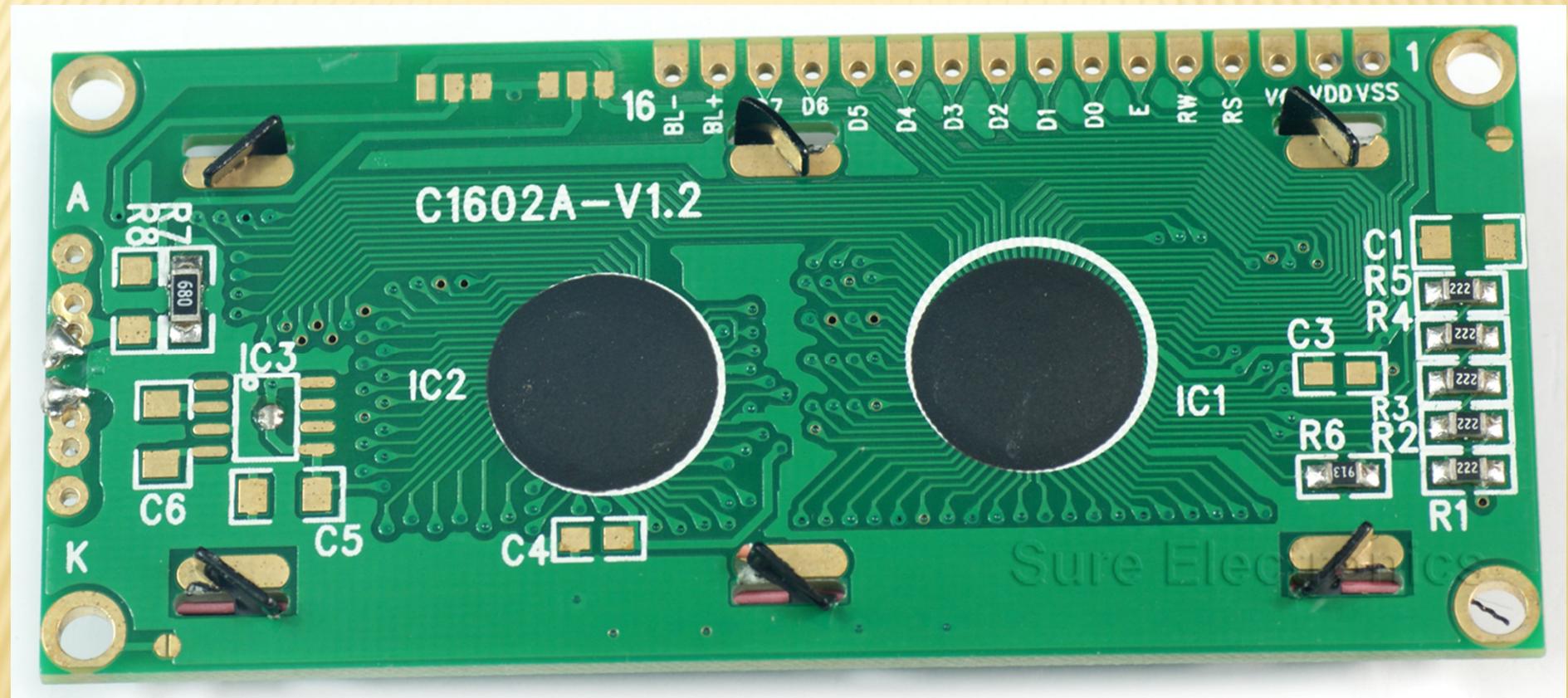
- A biblioteca *LiquidCrystal* permite controlar displays textuais de LCD compatíveis com o *driver* Hitachi HD44780.
- Há muitos desses dispositivos no mercado e podem ser reconhecidos pela interface de 16 pinos.



Vista Frontal

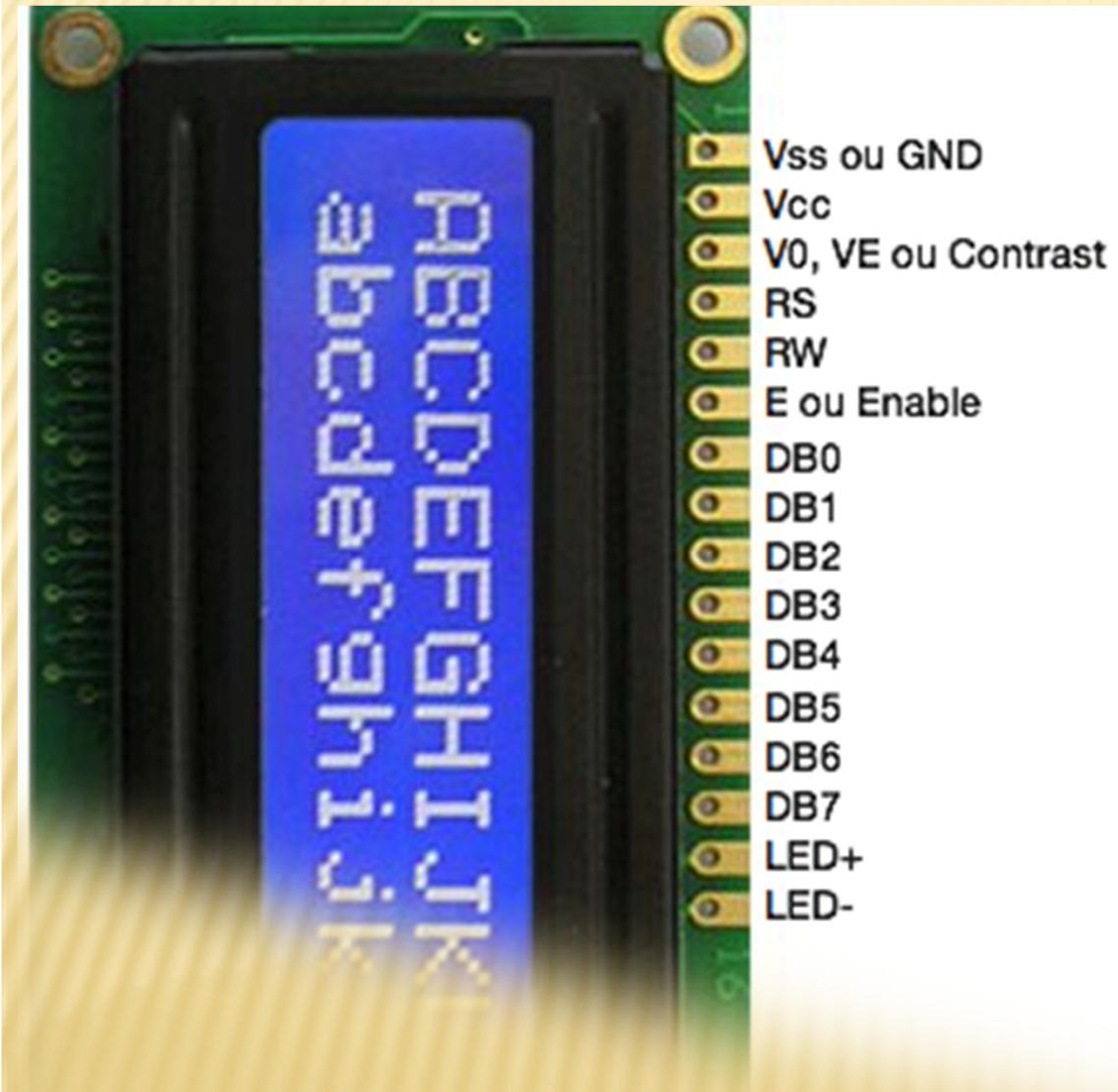
1	VSS (Ground)
2	VDD (+ve)
3	VE (Contrast Voltage)
4	Register Select
5	Read/Write
6	Enable
7	Data 0
8	Data 1
9	Data 2
10	Data 3
11	Data 4
12	Data 5
13	Data 6
14	Data 7
15	Backlight Anode (+ve)
16	Backlight Cathode (Ground)

# INTERFACE DE 16 PINOS



Vista Posterior

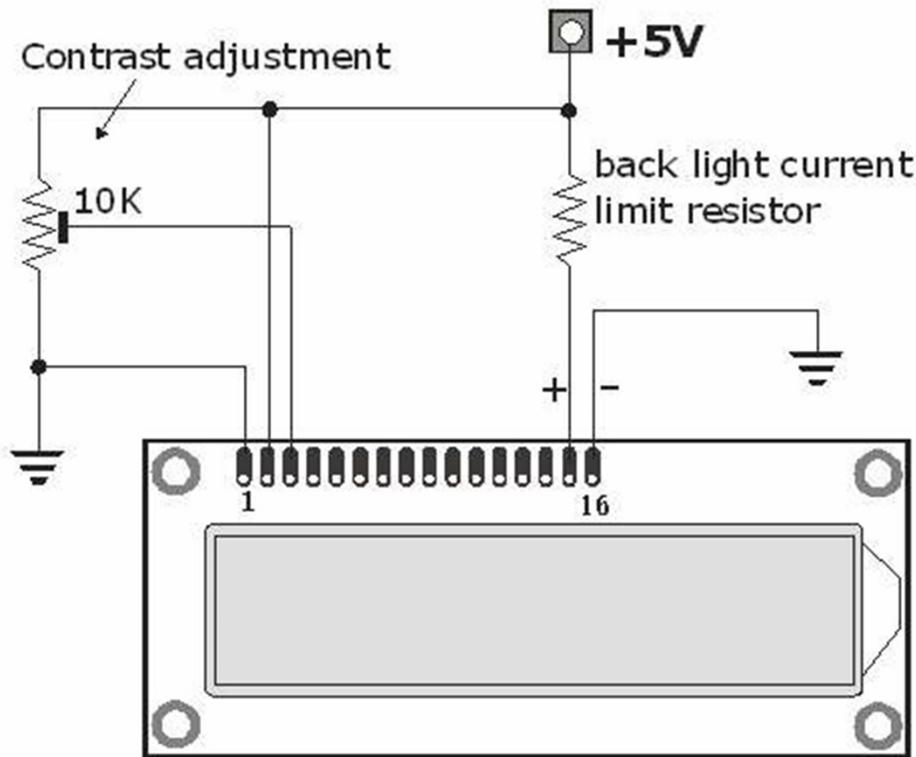
# INTERFACE DE 16 PINOS



Os LCDs textuais usam comunicação do tipo paralela para receber comando do microcontrolador...

vários pinos de interface com sinais simultâneos para controlar a exibição dos caracteres

# INTERFACE DE 16 PINOS

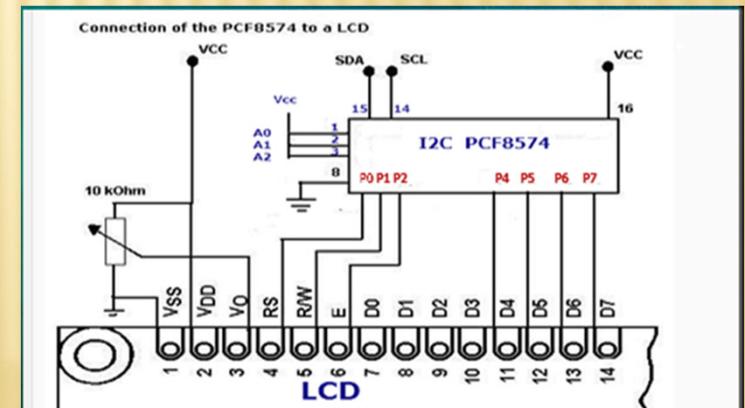


## Pin Assignment

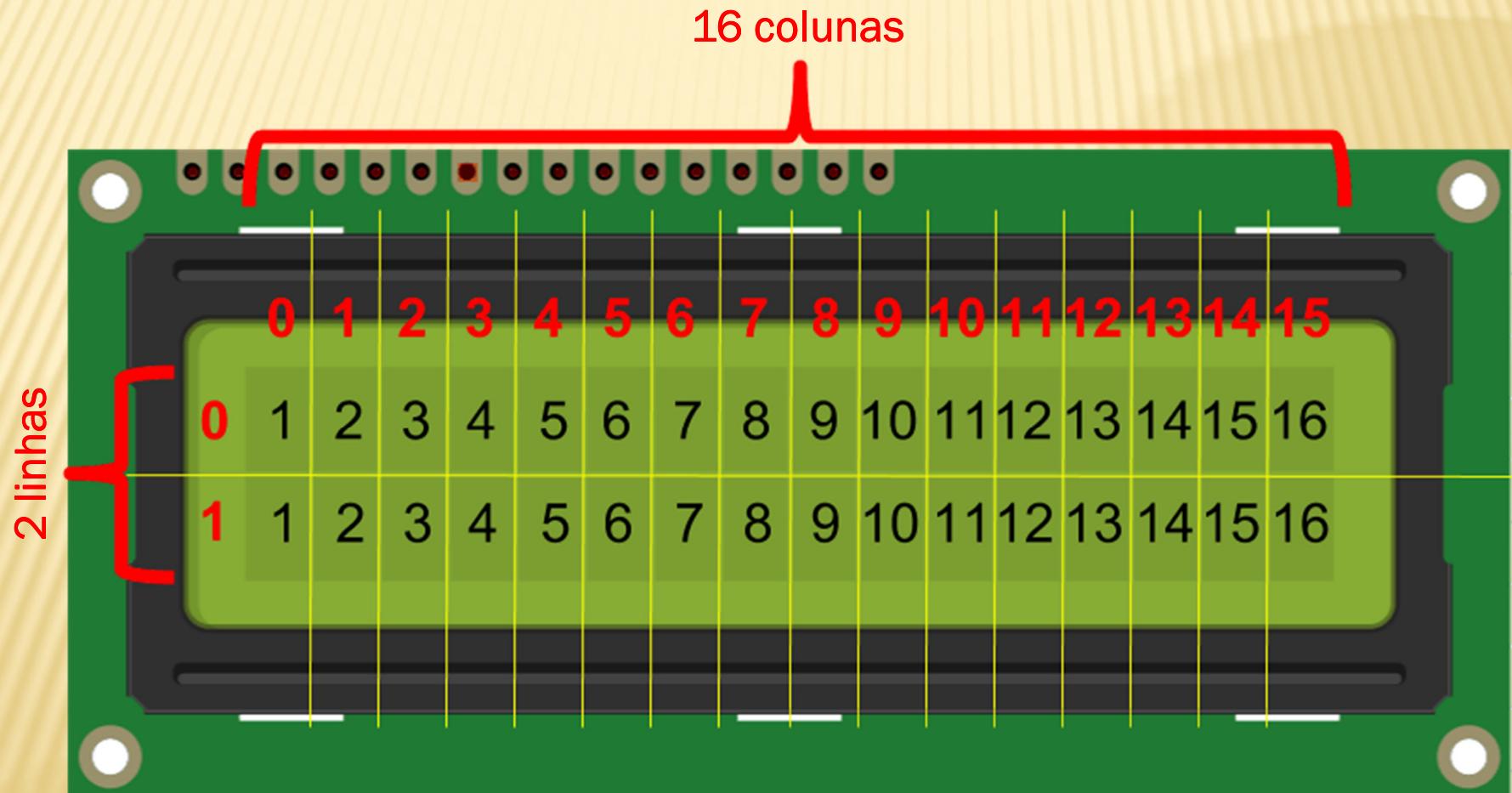
- 1 ----- V<sub>ss</sub> (GND, supply)
- 2 ----- V<sub>cc</sub> (+5V)
- 3 ----- V<sub>ee</sub> (Contrast adj.)
- 4 ----- RS
- 5 ----- R/W (Read/Write)
- 6 ----- E (enable)
- 7~14 -- (Data bit 0~7)
- 15 ----- Back light (+)
- 16 ----- Back light (-)

# INTERFACE I<sub>2</sub>C

- Fácil de usar em projetos com o Arduino
- A versão de LCD com interface I<sub>2</sub>C é mais cara que a versão com interface paralela
- Usa apenas de 4 fios para se conectar ao Arduino
  - V<sub>cc</sub>, Gnd, SCL e DAS
- Módulo LCD com interface I<sub>2</sub>C

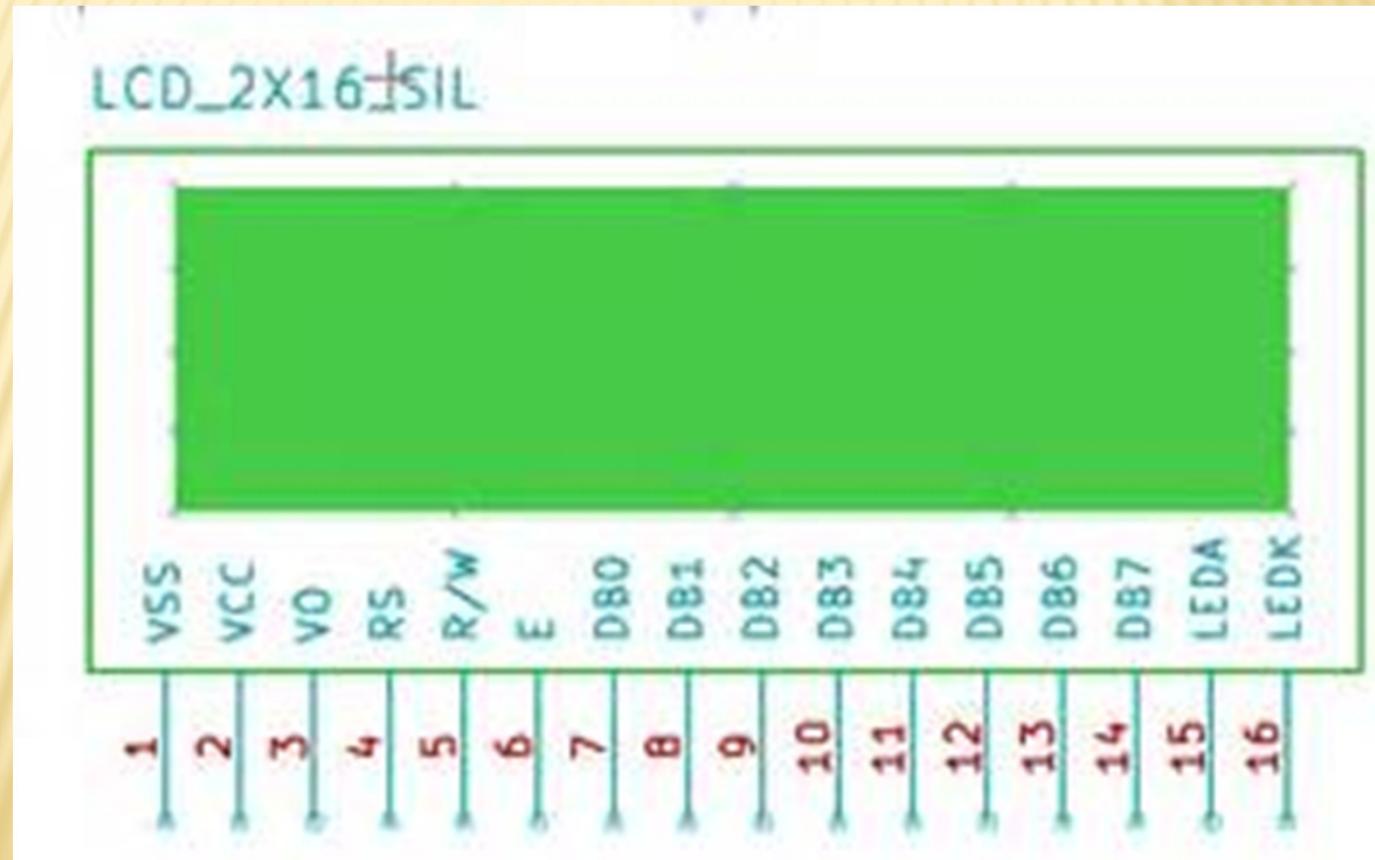


# POSIÇÕES DO CURSOR



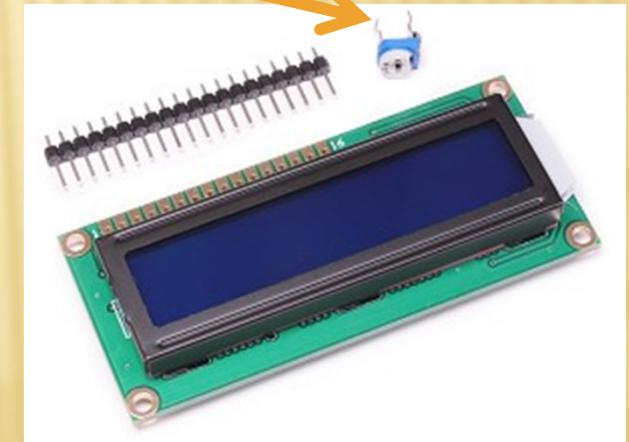
# MÓDULO 1602

- Mod. 1602 (16 colunas e 2 linhas)



# COMPONENTES

- ✖ Arduino
- ✖ Display LCD (*screen*) compatível com o acionador (*driver*) Hitachi HD44780
- ✖ Potenciômetro Linear 10 k $\Omega$  (ajuste de contraste)
- ✖ Resistor 220  $\Omega$
- ✖ Fios para conexões
- ✖ Protoboard



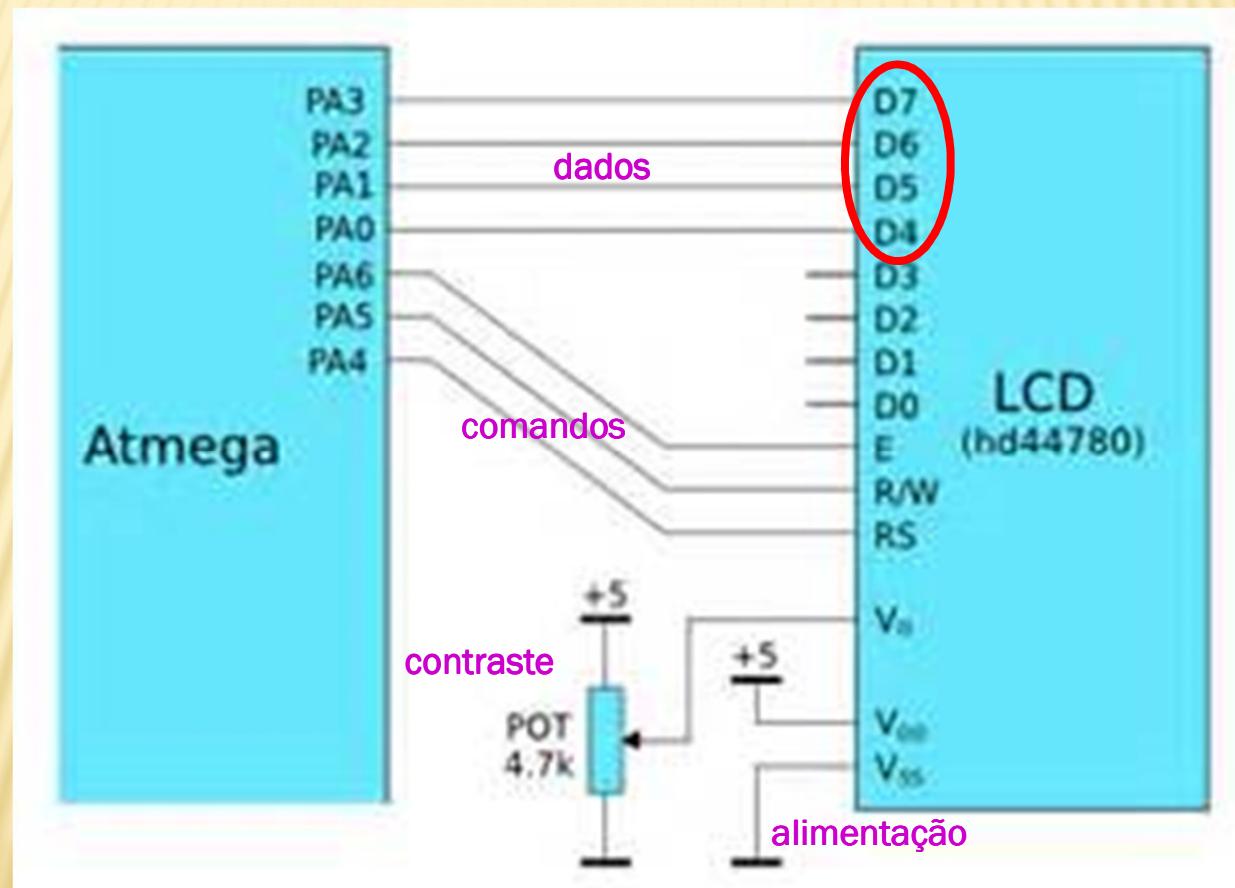
# SINAIS DO MÓDULO 1602

## ✖ Sinais

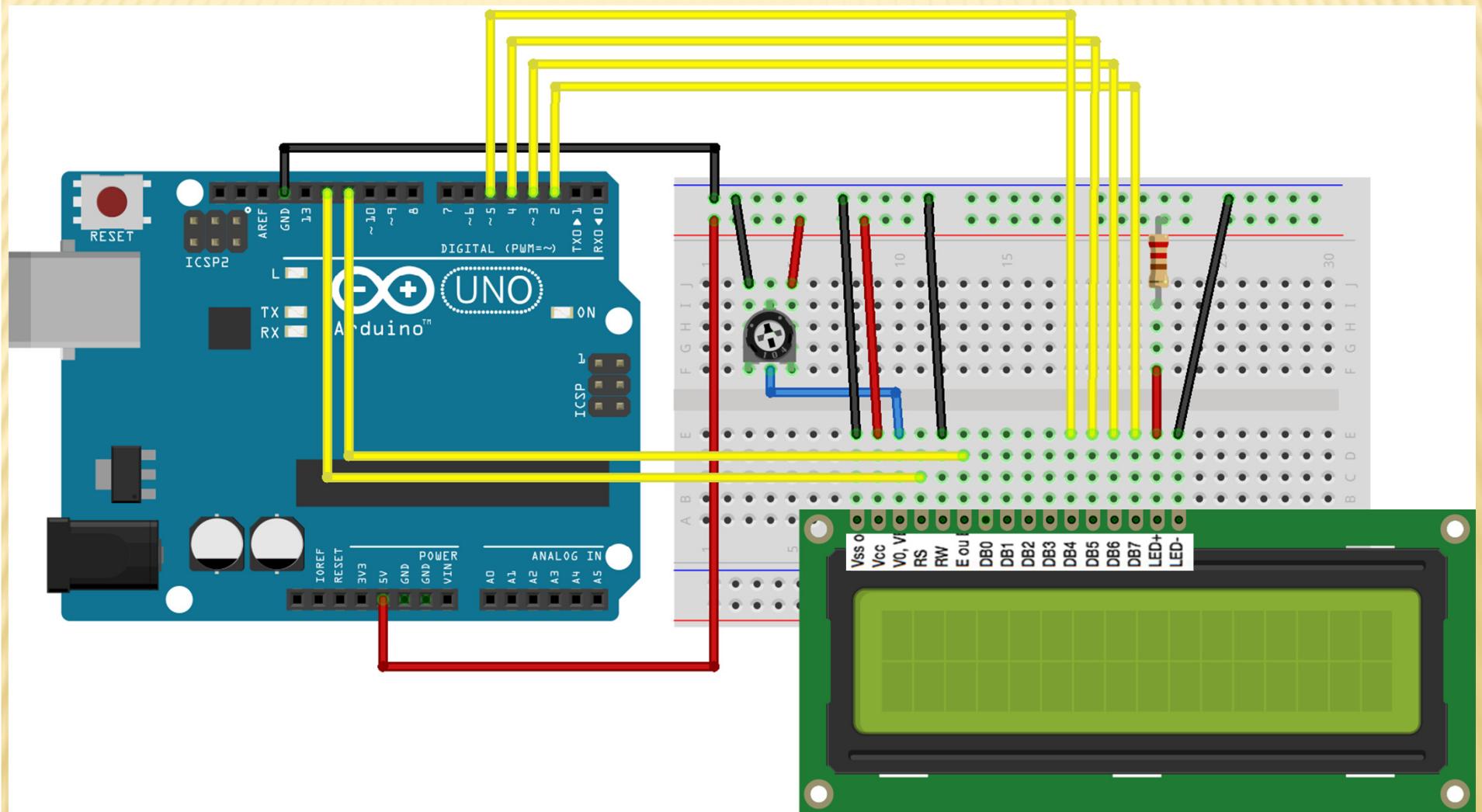
Pin no.	Symbol	Function
1	GND	Power supply ground
2	VCC	+5V supply
3	VEE	Contrast adjustment voltage
4	RS	Register select (H: data, L: instruction)
5	R/ $\overline{W}$	Read/Write data (H: LCD $\rightarrow$ $\mu$ C, L: $\mu$ C $\rightarrow$ LCD)
6	E	Enable pulse
7	D0	Data bit 0
8	D1	Data bit 1
9	D2	Data bit 2
10	D3	Data bit 3
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7
15	A	Anode of backlight LED
16	K	Cathode of backlight LED

# LIGAÇÕES COM DADOS MULTIPLEXADOS

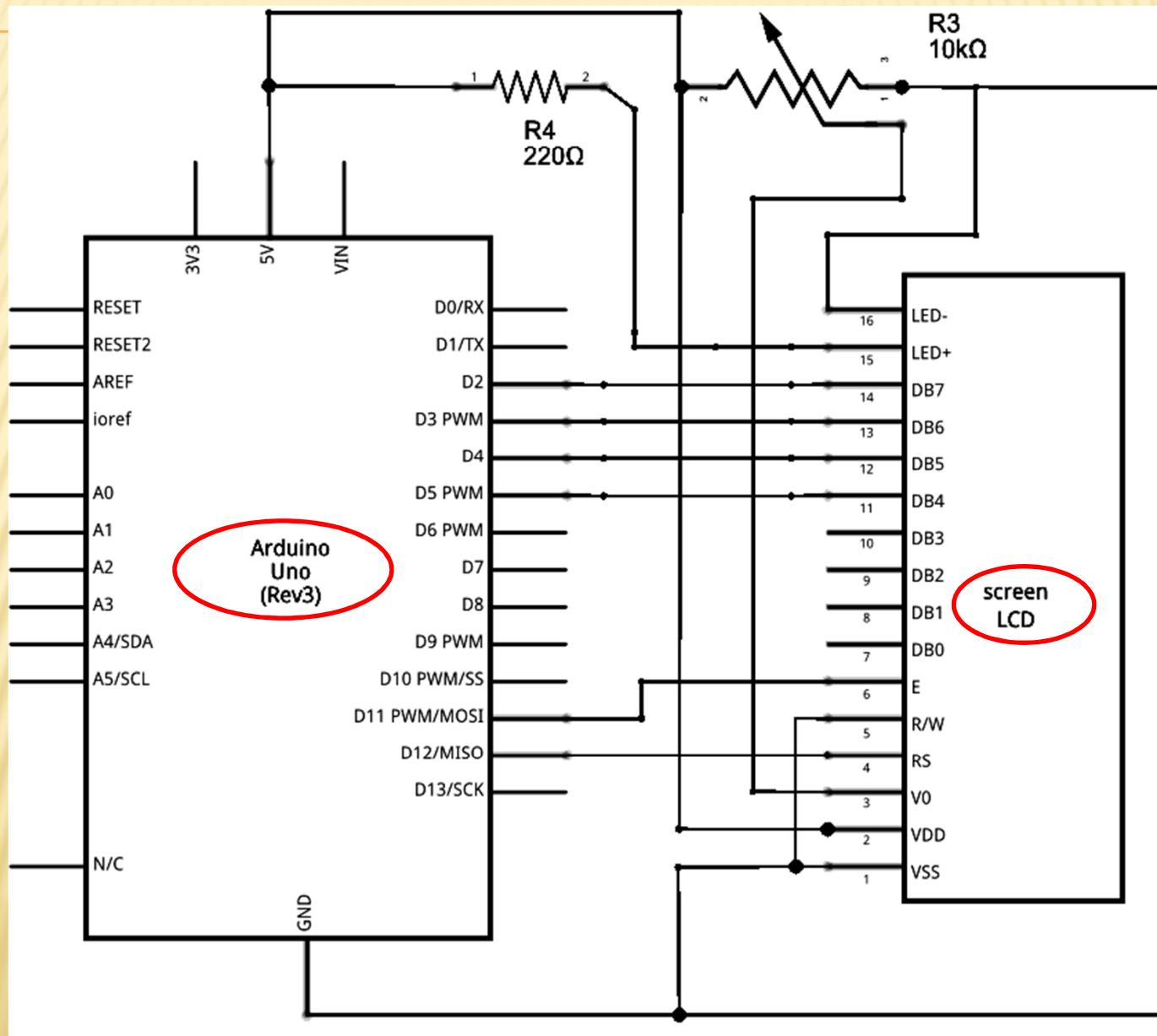
Usa nibble superior: D4-D7



# LIGAÇÕES COM DADOS MUXS



# ESQUEMÁTICO



# Datasheet

Instruction Set for the HD44780 LCD Module

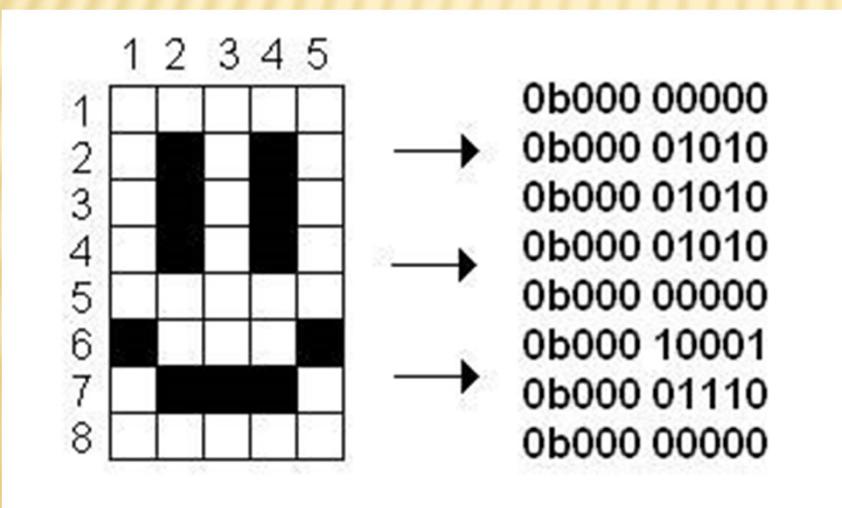
Comandos	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	COMMAND CODE	COMMAND CODE	E-CYCLE $f_{osc}=250\text{KHz}$
SCREEN CLEAR	0	0	0	0	0	0	0	0	0	1		Screen Clear, Set AC to 0 Cursor Reposition	1.64ms
CURSOR RETURN	0	0	0	0	0	0	0	0	1	*		DDRAM AD=0, Return, Content Changeless	1.64ms
INPUT SET	0	0	0	0	0	0	0	1	I/D	S		Set moving direction of cursor, Appoint if move	40us
DISPLAY SWITCH	0	0	0	0	0	0	1	D	C	B		Set display on/off,cursor on/off, blink on/off	40us
SHIFT	0	0	0	0	0	1	S/C	R/L	*	*		Remove cursor and whole display, DDRAM changeless	40us
FUNCTION SET	0	0	0	0	1	DL	N	F	*	*		Set DL,display line,font	40us
CGRAM AD SET	0	0	0	1							ACG	Set CGRAM AD, send receive data	40us
DDRAM AD SET	0	0	1								ADD	Set DDRAM AD, send receive data	40us
BUSY/AD READ CT	0	1	BF								AC	Executing internal function, reading AD of CT	40us
CGRAM/ DDRAM DATA WRITE	1	0									DATA WRITE	Write data from CGRAM or DDRAM	40us
CGRAM/ DDRAM DATA READ	1	1									DATA READ	Read data from CGRAM or DDRAM	40us
											I/D=1: Increment Mode; I/D=0: Decrement Mode S=1: Shift S/C=1: Display Shift; S/C=0: Cursor Shift R/L=1: Right Shift; R/L=0: Left Shift DL=1: 8D DL=0: 4D N=1: 2R N=0: 1R F=1: 5x10 Style; F=0: 5x7 Style BF=1: Execute Internal Function; BF=0: Command Received	DDRAM: Display data RAM CGRAM: Character Generator RAM ACG: CGRAM AD ADD: DDRAM AD & Cursor AD AC: Address counter for DDRAM & CGRAM	E-cycle changing with main frequency. Example: If fcp or $f_{osc}=270\text{KHz}$ 40us x 250/270 =37us

# Tabela de Caracteres do HD44780

b7- b3- b4 -b0	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111			
CG RAM (1)	Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ Ⓕ Ⓖ Ⓗ Ⓘ Ⓙ Ⓕ Ⓗ Ⓙ Ⓗ	Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ Ⓕ Ⓖ Ⓗ Ⓘ Ⓙ Ⓕ Ⓗ Ⓙ Ⓗ	Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ Ⓕ Ⓖ Ⓗ Ⓘ Ⓙ Ⓕ Ⓗ Ⓙ Ⓗ	---	---	---	---	---	---	---	---	---	---			
(2)	! 1 A Q a q , A T 4 S q	! 1 A Q a q , A T 4 S q	! 1 A Q a q , A T 4 S q	!	1	A	Q	a	q	,	A	T	4	S	q	
(3)	" 2 B R b r " I U X P E	" 2 B R b r " I U X P E	" 2 B R b r " I U X P E	"	2	B	R	b	r	"	I	U	X	P	E	
(4)	# 3 C S c s , U T E S ~	# 3 C S c s , U T E S ~	# 3 C S c s , U T E S ~	#	3	C	S	c	s	,	U	T	E	S	~	
(5)	\$ 4 D T d t , I T T P Q	\$ 4 D T d t , I T T P Q	\$ 4 D T d t , I T T P Q	\$	4	D	T	d	t	,	I	T	T	P	Q	
(6)	% 5 E U e u , O A U C Q	% 5 E U e u , O A U C Q	% 5 E U e u , O A U C Q	%	5	E	U	e	u	,	O	A	U	C	Q	
(7)	& 6 F U f v V K N E P S	& 6 F U f v V K N E P S	& 6 F U f v V K N E P S	&	6	F	U	f	v	V	K	N	E	P	S	
CG RAM (8)	* 7 G W g w , V A L Q P M	* 7 G W g w , V A L Q P M	* 7 G W g w , V A L Q P M	*	7	G	W	g	w	,	V	A	L	Q	M	
(1)	( 8 H X h x , V A N R K )	( 8 H X h x , V A N R K )	( 8 H X h x , V A N R K )	(	8	H	X	h	x	,	V	A	N	R	)	
(2)	) 9 I Y i y , T J N Y	) 9 I Y i y , T J N Y	) 9 I Y i y , T J N Y	)	9	I	Y	i	y	,	T	J	N	Y		
(3)	* : J Z j z , C A L J Z	* : J Z j z , C A L J Z	* : J Z j z , C A L J Z	*	:	J	Z	j	z	,	C	A	L	J	Z	
(4)	+ ; K [ k , { A V H P }	+ ; K [ k , { A V H P }	+ ; K [ k , { A V H P }	+	;	K	[	k	,	{	A	V	H	P	}	
(5)	, < L ¥ 1   Y A S F W F M	, < L ¥ 1   Y A S F W F M	, < L ¥ 1   Y A S F W F M	,	<	L	¥	1		Y	A	S	F	W	F	M
(6)	- = M ] m } Y Z H N T +	- = M ] m } Y Z H N T +	- = M ] m } Y Z H N T +	-	=	M	]	m	}	Y	Z	H	N	T	+	
(7)	. > N ^ n → a t k ^	. > N ^ n → a t k ^	. > N ^ n → a t k ^	.	>	N	^	n	→	a	t	k	^			
CG RAM (8)	/ ? O _ o ← u V F B G	/ ? O _ o ← u V F B G	/ ? O _ o ← u V F B G	/	?	O	_	o	←	u	V	F	B	G		

# Caractere Personalizado

- ✖ São permitidos até 8 caracteres personalizados, numerados de 0 a 7
- ✖ A aparência de cada caractere personalizado é especificada por uma matriz de 8 bytes, um para cada linha
- ✖ Os 5 bits menos significativos de cada byte determinam os pixels na linha. Para exibir um caractere personalizado na tela, `write()` seu número



```

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

byte smiley[8] = { B00000,
                   B01010,
                   B01010,
                   B01010,
                   B00000,
                   B10001,
                   B01110,
                   B00000      };

void setup() {
  lcd.createChar(0,smiley);
  lcd.begin(16,2);
  lcd.write(byte(0));
}

void loop() { }
    
```

# BIBLIOTECA - LIQUIDCRYSTAL

- ✖ Projetada para o Arduino controlar LCD's compatíveis com o chipset Hitachi HD44780 que é encontrado na maioria dos displays textuais de LCD
- ✖ Comunicação na forma paralela, com 4 ou 8 bits
- ✖ Quantidade de pinos de E/S
  - + 8 sinais de dados + 3 sinais de controle<sup>1</sup> = 11 ou 10 pinos
  - + 4 sinais de dados + 3 sinais de controle = 7 ou 6 pinos

<sup>1</sup> RS, ENABLE e opcionalmente R/W

# BIBLIOTECA - LIQUIDCRYSTAL

## ✖ Método Construtor de Objetos:

### + LiquidCrystal(pinos)

#### ✖ Sintaxe

- \* `LiquidCrystal(rs, enable, d4, d5, d6, d7)`
- \* `LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)`
- \* `LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)`
- \* `LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)`

#### ✖ Parâmetros

- \* `rs`: número do pino Arduino conectado ao pino RS do LCD
- \* `rw`: número do pino Arduino está conectado ao pino RW do LCD (opcional)
- \* `enable`: número do pino Arduino conectado ao pino de habilitação do LCD
- \* `d0, d1, d2, d3, d4, d5, d6, d7`: números dos pinos Arduino conectados aos pinos de dados correspondentes do LCD.
- \* `d0, d1, d2 e d3` são opcionais; Se omitidos, o LCD será controlado apenas por quatro linhas de dados (`d4, d5, d6, d7`).

# BIBLIOTECA - LIQUIDCRYSTAL

- ✖ Método Construtor de Objetos:

- + LiquidCrystal(pinos)

- ✖ Exemplo

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,10,5,4,3,2); // cria objeto lcd

void setup() {
    lcd.begin(16,2); // configura objeto lcd
    lcd.print("Ola Mundo!");
}

void loop() {
```

# BIBLIOTECA - LIQUIDCRYSTAL

- ✖ Outras funções (métodos) da Classe LiquidCrystal:
  - ✖ begin(), clear(), home(), setCursor(col, row),
  - ✖ write(dados), print(dados[,base<sup>\*</sup>]),
  - ✖ cursor(), noCursor(),
  - ✖ blink(), noBlink(), display(), noDisplay(),
  - ✖ scrollDisplayLeft(), scrollDisplayRight(),
  - ✖ autoscroll(), noAutoscroll(), leftToRight(), rightToLeft()
  - ✖ createChar()

\* BIN, OCT, DEC, HEX

# Exemplo: "Hello World!"

## ✖ Código

```
/* Uso da biblioteca LiquidCrystal - Hello World
```

Demonstra o uso de um display LCD 16x2. A biblioteca *LiquidCrystal* trabalha com todos os displays LCD compatíveis com o acionador (driver) Hitachi HD44780. Este sketch mostra (prints) "Hello World!" no LCD e mostra o tempo...

O circuito:

- \* pino LCD RS no pino digital 12
- \* pino LCD Enable no pino digital 11
- \* pino LCD D4 no pino digital 5
- \* pino LCD D5 no pino digital 4
- \* pino LCD D6 no pino digital 3
- \* pino LCD D7 no pino digital 2
- \* pino LCD R/W à terra (ground)
- \* pino LCD VSS à terra (ground)
- \* pino LCD VCC ao +5 V
- \* potenciômetro de 10k :
  - \* as pontas em +5 V e terra (ground)
  - \* o centro no pino VO do LCD VO (pino 3)

Biblioteca originalmente adicionada em 18/04/2008 por David A. Mellis e modificada em 5 Jul 2009 por Limor Fried (<http://www.ladyada.net>) exemplo acrescentado em 09/07/2009 por Tom Igoe e modificado em 22/11/2010 por Tom Igoe

Este cód. exemplo é de domínio público: [www.arduino.cc/en/Tutorial/LiquidCrystal](http://www.arduino.cc/en/Tutorial/LiquidCrystal) 23

# Exemplo: "Hello World!"

## ✗ Código

```
// inclui o código da biblioteca:  
#include <LiquidCrystal.h>  
  
// inicia a biblioteca com os números dos pinos da interface  
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  
  
void setup() {  
    lcd.begin(16, 2);          // configura a qtde de colunas e linhas do LCD:  
    lcd.print("Hello World!"); // mostra a mensagem no LCD  
}  
  
void loop() {  
    // posiciona o cursor na coluna 0 e na linha 1  
    // obs: linha 1 é a segunda linha (inferior), linha 0 é a primeira (superior):  
    lcd.setCursor(0, 1);  
    // mostra a qtde de segundos desde o último reset do Arduino:  
    lcd.print(millis()/1000);  
}
```

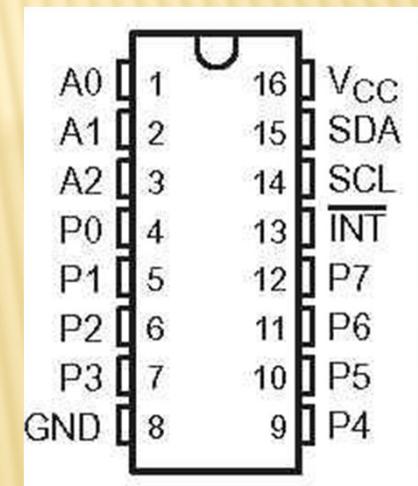
Fonte: [24](https://www.arduino.cc/en/Tutorial>HelloWorld</a></p></div><div data-bbox=)

# Expansor de E/S - PCF8574

Via comunicação I2C

## Características

- + Possui uma porta de E/S quase-bidirecional de 8 bits (P0–P7)
- + Possui saídas com travas (*latches*) de alta capacidade de corrente para o acionamento direto de LEDs
- + Cada E/S quase bidirecional pode ser usada como uma entrada ou saída sem o uso de um sinal de controle de direção de dados
- + Faixa de endereços **I2C**: 0x20 a 0x27 (enfileiramento de até 8 unidades/barramento I2C)
- + Bit menos significativo de endereço é usado para seleção de operação: *Write* (0) ou *Read* (1)
- + Oito PCF8574's podem ser conectados ao barramento **I2C**, com seleção de endereço via hardware em cada dispositivo



# Expansor de E/S - PCF8574

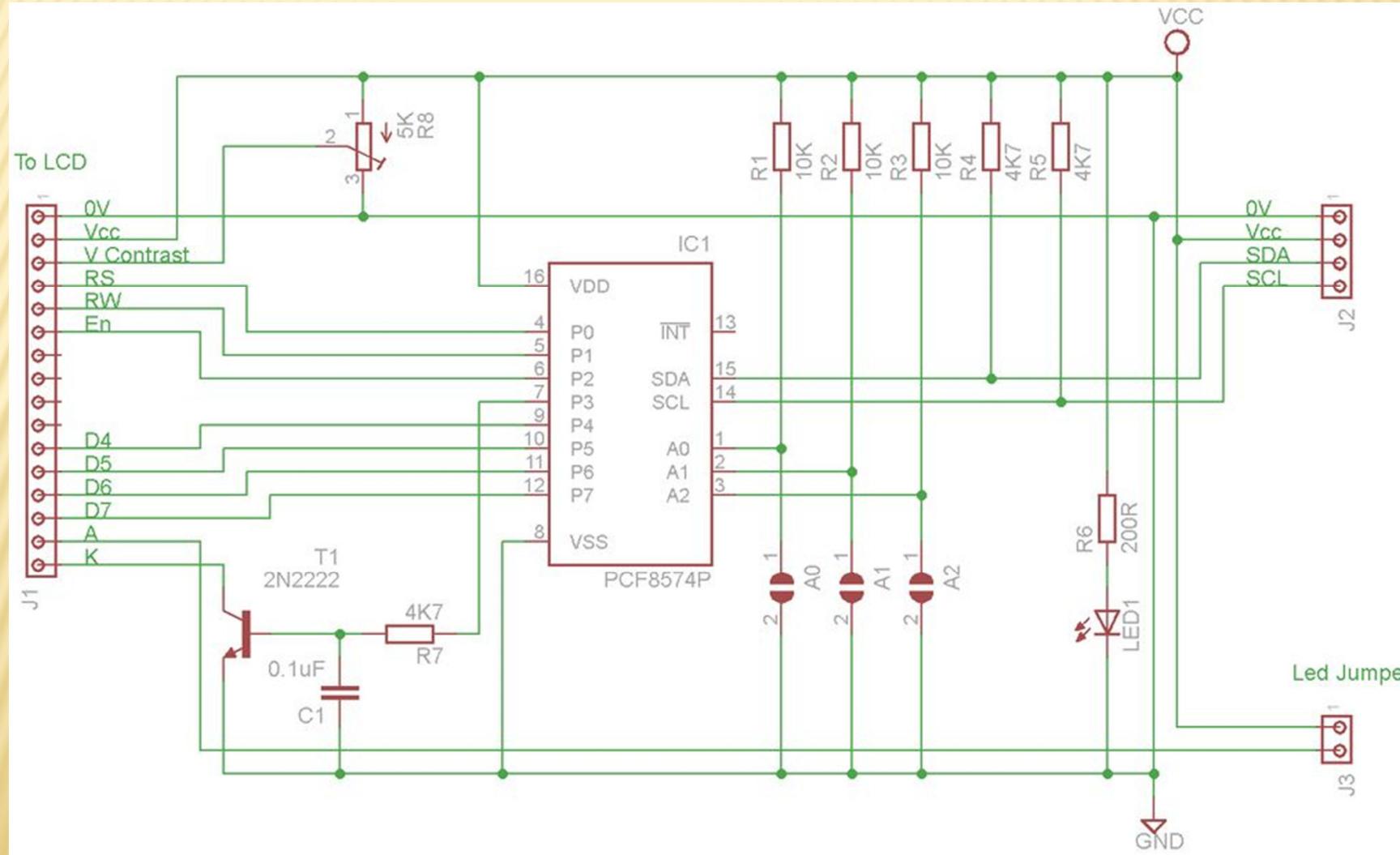
Via comunicação I2C

## ✗ Endereçamento

// PCF8574P - Endereços de 32 a 39	32-> A0=0 A1=0 A2=0
// ou	33-> A0=1 A1=0 A2=0
// PCF8574T	34-> A0=0 A1=1 A2=0
// ou	35-> A0=1 A1=1 A2=0
// PCF8574TS	36-> A0=0 A1=0 A2=1
//	37-> A0=1 A1=0 A2=1
//	38-> A0=0 A1=1 A2=1
//	39-> A0=1 A1=1 A2=1
//	
// PCF8574AP - Endereços de 56 a 63	56-> A0=0 A1=0 A2=0
// ou	57-> A0=1 A1=0 A2=0
// PCF8574AT	58-> A0=0 A1=1 A2=0
// ou	59-> A0=1 A1=1 A2=0
// PCF8574ATS	60-> A0=0 A1=0 A2=1
//	61-> A0=1 A1=0 A2=1
//	62-> A0=0 A1=1 A2=1
//	63-> A0=1 A1=1 A2=1

# Expansor de E/S - PCF8574

- ✖ Módulo I2C - LCD HD44780

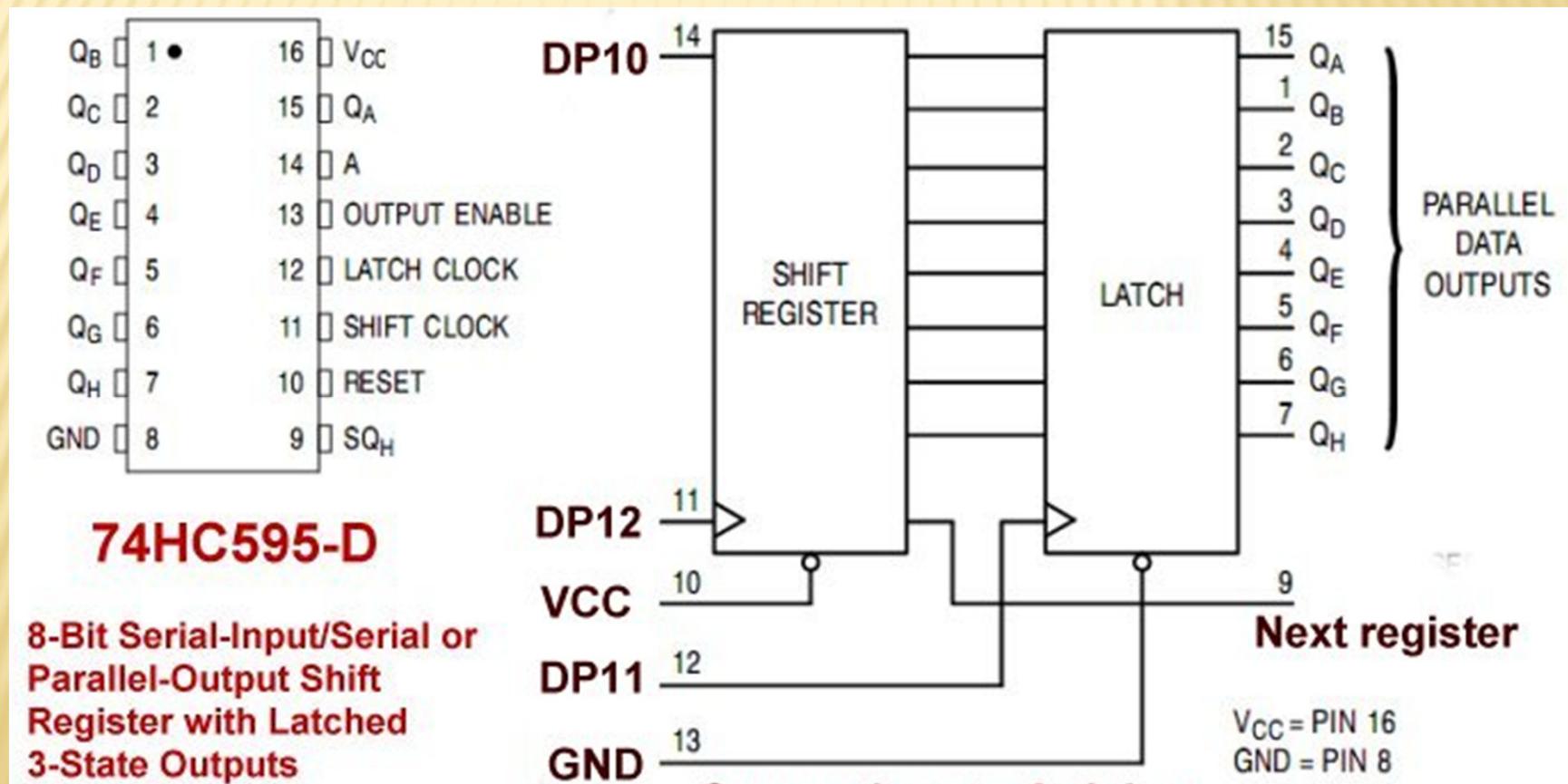


# Expansor de E/S - 74HC595

Registrador de Deslocamento

## Características

- + Registrador de deslocamento (*shift register*) de 8 bits, Entrada Serial – Saída Paralela
- + Tensão de funcionamento e consumo: 2 V a 6 V / 80 uA
- + Corrente de saída (máx): 35 mA

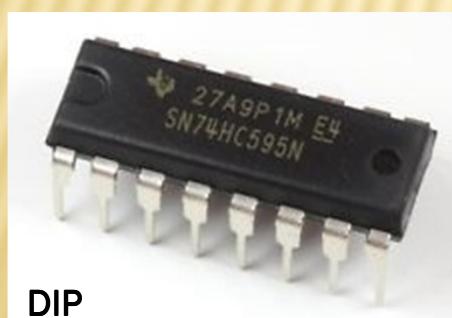
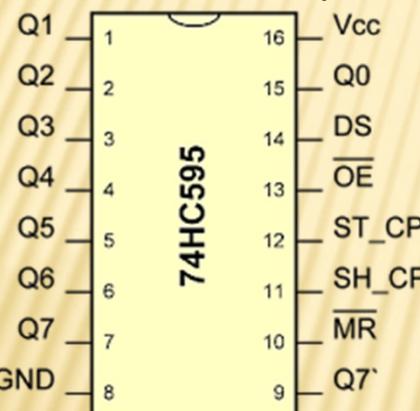


# Expansor de E/S - 74HC595

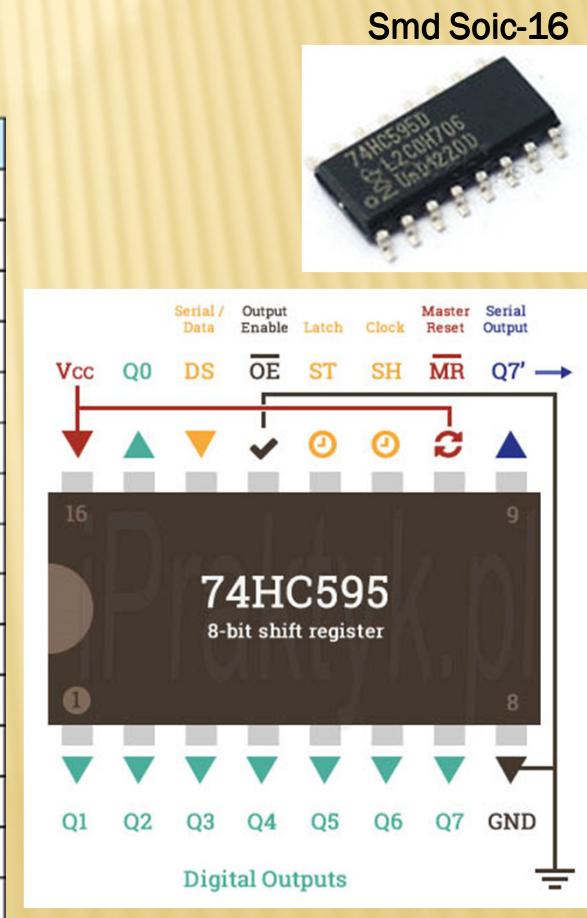
Registrador de Deslocamento

## Características

- + Tensão de entrada mínima de alto nível: 3,15 V @ (Vcc = 4,5 V)
- + Tensão de entrada máxima de baixo nível: 1,35 V @ (Vcc = 4,5 V)
- + Pode ser facilmente cascateverado com mais CI's para obter mais saídas
- + Frequência Máxima do Relógio: 25 MHz @ 4,5 V
- + Encapsulamentos: PDIP, GDIP, PDSO de 16 pinos



Pin	Symbol	Description
1	Q1	Parallel data output (bit-1)
2	Q2	Parallel data output (bit-2)
3	Q3	Parallel data output (bit-3)
4	Q4	Parallel data output (bit-4)
5	Q5	Parallel data output (bit-5)
6	Q6	Parallel data output (bit-6)
7	Q7	Parallel data output (bit-7)
8	GND	Ground (0 V)
9	Q7'	Serial Data Output
10	$\overline{MR}$	Master Reset (Active Low)
11	SH_CP	Shift Register Clock Input
12	ST_CP	Storage Register Clock Input
13	$\overline{OE}$	Output Enable (Active Low)
14	DS	Serial Data Input
15	Q0	Parallel data output (bit-8)
16	Vcc	Positive Supply Voltage

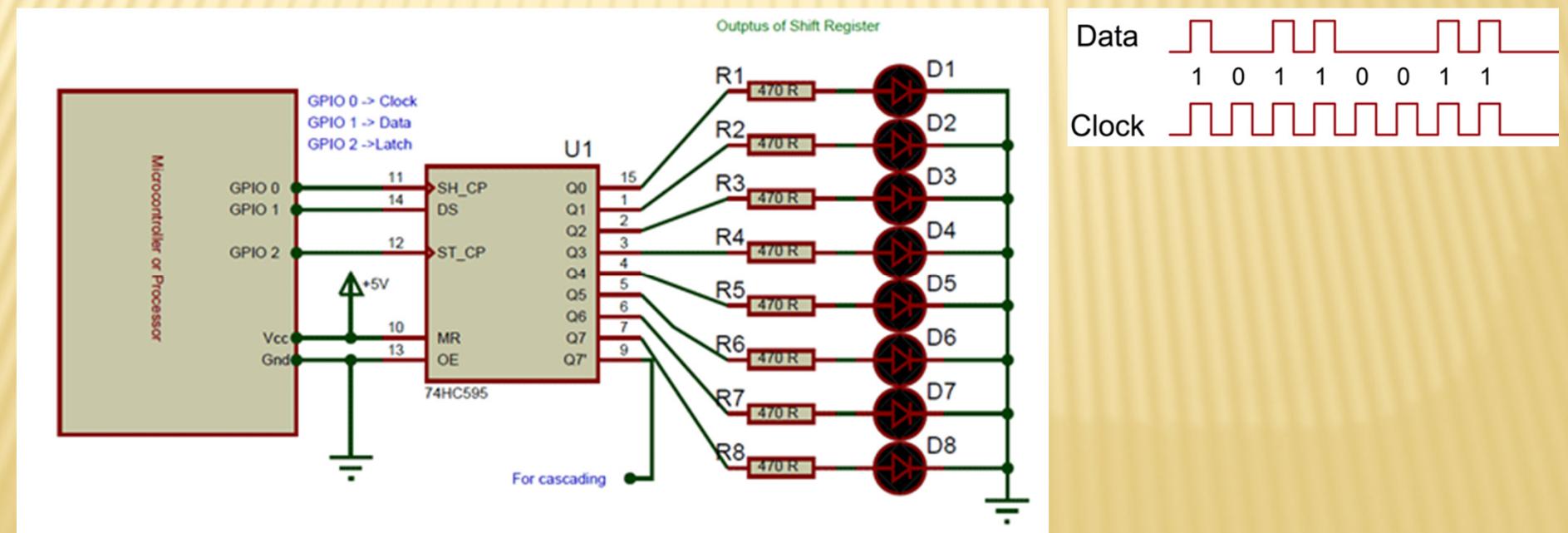


# Expansor de E/S - 74HC595

Registrador de Deslocamento

## Exemplo: acionamento de 8 LEDs

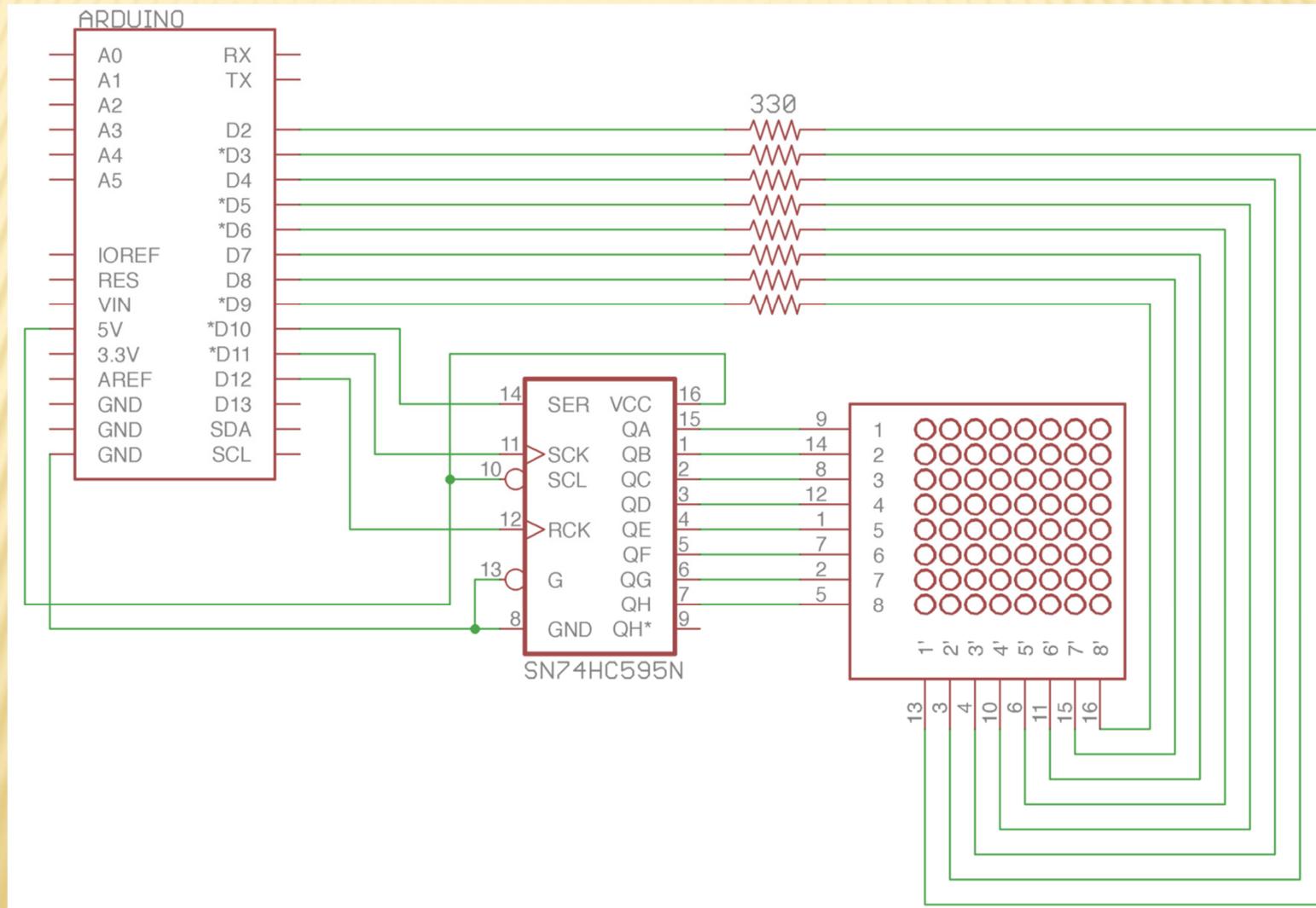
- + Os pinos 11, 14 e 12 são conectados aos pinos GPIO do microcontrolador e são usados para o controle da operação
- + No pino 11 temos o sinal de *clock* que sincroniza a operação de deslocamento dos dados.
- + No pino 14 temos os bits que serão deslocados pelo registrador.
- + No Pino 12 temos o sinal de travamento (*latch*) que atualiza os dados recebidos nos pinos de saída quando vai para HIGH.



# Expansor de E/S - 74HC595

Registrador de Deslocamento

- Exemplo: acionamento de matriz de LEDs



# Outros Expansores de E/S

Registrador de Deslocamento

<b>Item</b>	<b>Cl</b>	<b>Descrição</b>
1	4035	4-Bit Parallel in Parallel out Shift Register
2	74LS379	Quad Parallel Shift Register
3	4014	4 Bit static shift register
4	74LS166	8 Bit Shift Register
5	74LS323	8 Bit Shift/Storage Register
6	74LS164	S/P Shift Register
7	4015	Dual 4 Bit Static Register
8	74LS299	8 Bit Shift/Storage Register

---

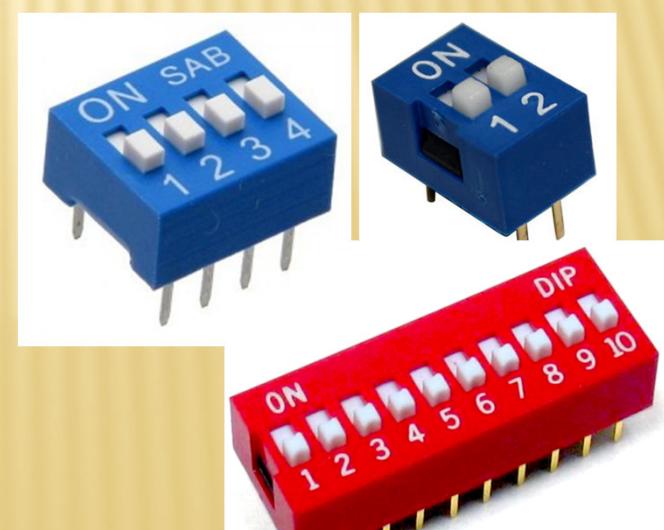
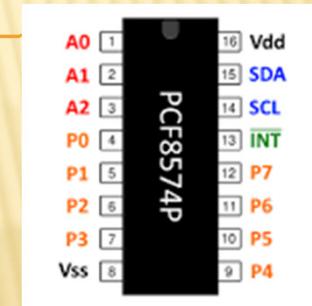
# **PROJETO 1:**

# **LEDS & DIP SWITCHES**

# 12 LEDs

- Os 8 LEDs estão conectados aos pinos E/S do primeiro dispositivo PCF8574 através de resistores limitadores de corrente de  $330\ \Omega$
- Outros 4 LEDs são conectados ao *nibble* superior dos pinos de E/S do segundo PCF8574 através de resistores limitadores de corrente de  $330\ \Omega$
- Uma chave DIP de 4 vias é conectada ao *nibble* inferior dos pinos de E/S do segundo dispositivo PCF8574 através de resistores de *pull-up* de  $10\ k\Omega$

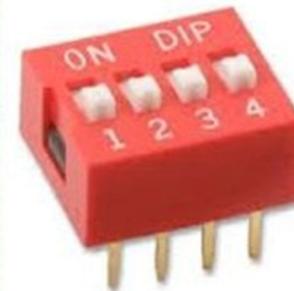
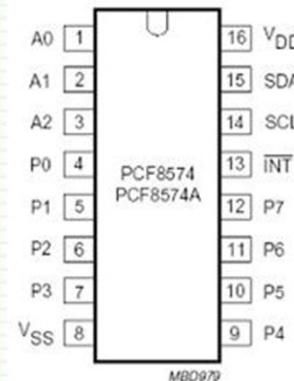
Interruptores DIP (*DIP Switches*): são interruptores elétricos manuais dispostos em encapsulamento DIP (*Dual InLine Package*). Podem ser soldados diretamente ao PCB. Comumente são usados para selecionar os modos de operação de um dispositivo eletrônico para situações específicas



# 12 LEDs

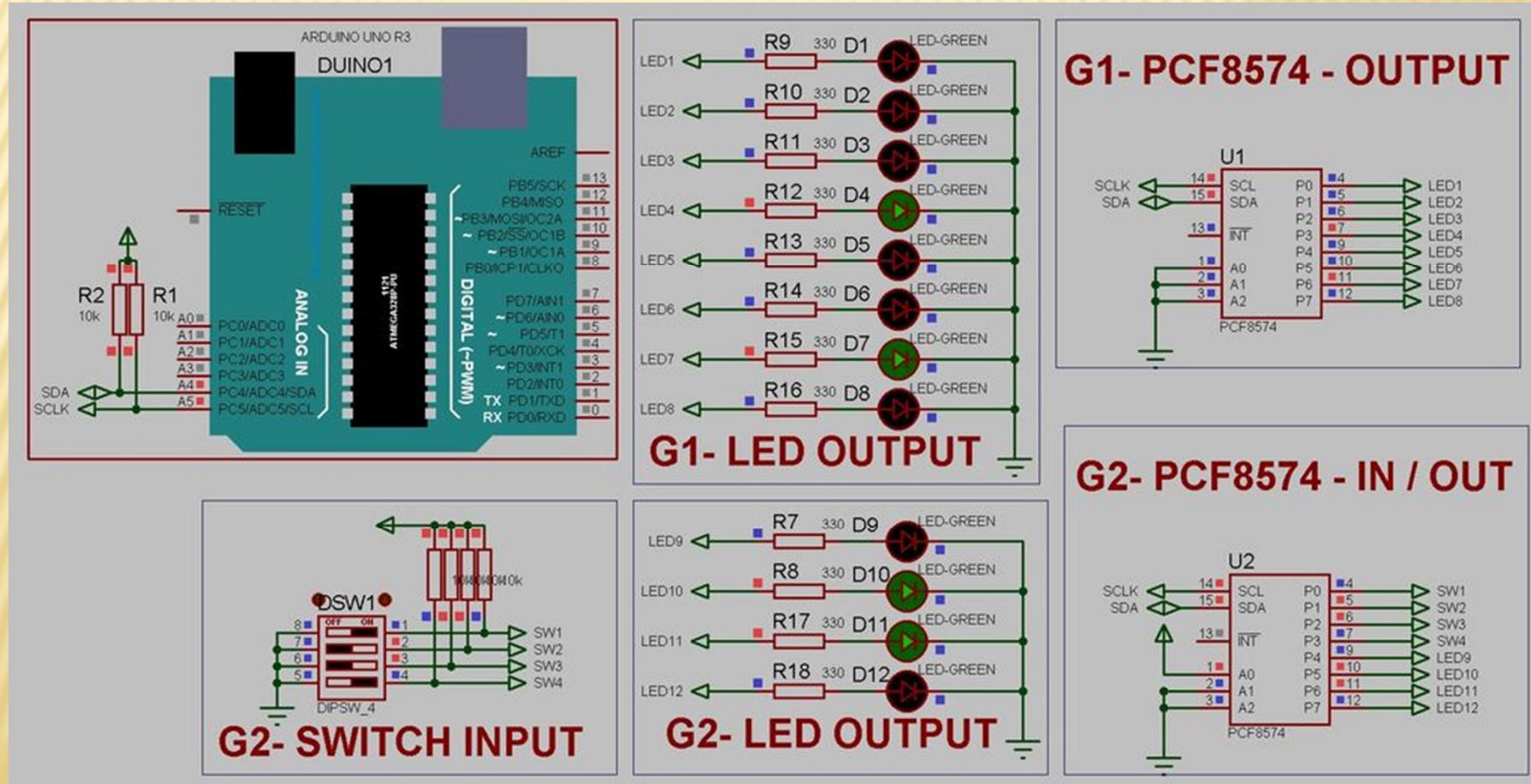
No projeto de sistemas incorporados, sempre que a falta de pinos digitais de E/S para conectar os sensores e atuadores, o **expansor de E/S** é usado para fornecer mais E/S digitais ao sistema...

Componentes:



# 12 LEDs

Interface de comunicação I2C: o pino de dados I2C (SDA) é conectado ao pino A4 e o pino do clock (SCL) é conectado ao pino A5...



# 12 LEDS

- Software

- A biblioteca Wire permite comunicação com o barramento I2C/TWI do Arduino
- Um contador binário é incrementado a cada 300 mS e o seu conteúdo é escrito no primeiro expansor de E/S, onde os LEDs estão conectados e configurados como pinos de saída digital
- O *nibble* inferior do segundo expansor de E/S é lido e seu valor é exibido no *nibble* superior do mesmo expansor de E/S
- O *nibble* inferior é configurado como ENTRADA e conectado ao DIP-SWITCH de 4 vias. O *nibble* superior é configurado como saída e está conectado a 4 LEDs

# 12 LEDS

```
// File: Expansao_E_S.ino          Por: Prof. Kaw
// Data: 04.09.2018                Fonte: M.Pugazhendi (muthuswamy.pugazhendi@gmail.com)
#include <Wire.h>
#define PCF8574_1 B0100000
#define PCF8574_2 B0100001

void setup() {
  Wire.begin();
  EExpansorGrav(PCF8574_2, 0x0F);
}

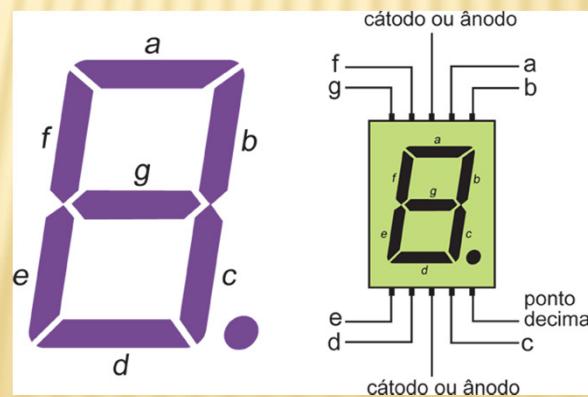
void loop() {
  byte i, j, k;
  for(i = 0; i < 255; i++) {
    EExpansorGrav(PCF8574_1,i);    delay(50);
    j = EExpansorLeit(PCF8574_2);  delay(50);
    k = (j << 4) | 0x0F;
    EExpansorGrav(PCF8574_2,k);    delay(200);
  }
}

void EExpansorGrav(byte endereco, byte dado) {          // escreve um byte no expansor de E/S
  Wire.beginTransmission(endereco);
  Wire.write(dado);
  Wire.endTransmission();
}

byte EExpansorLeit(int endereco) {                      // Lê um byte do expansor de E/S
  byte dado = 0;
  Wire.requestFrom(endereco,1);
  if(Wire.available())
    dado = Wire.read();
  return dado;
}
```

# PROJETO 2:

# DISPLAY DE 7 SEGMENTOS



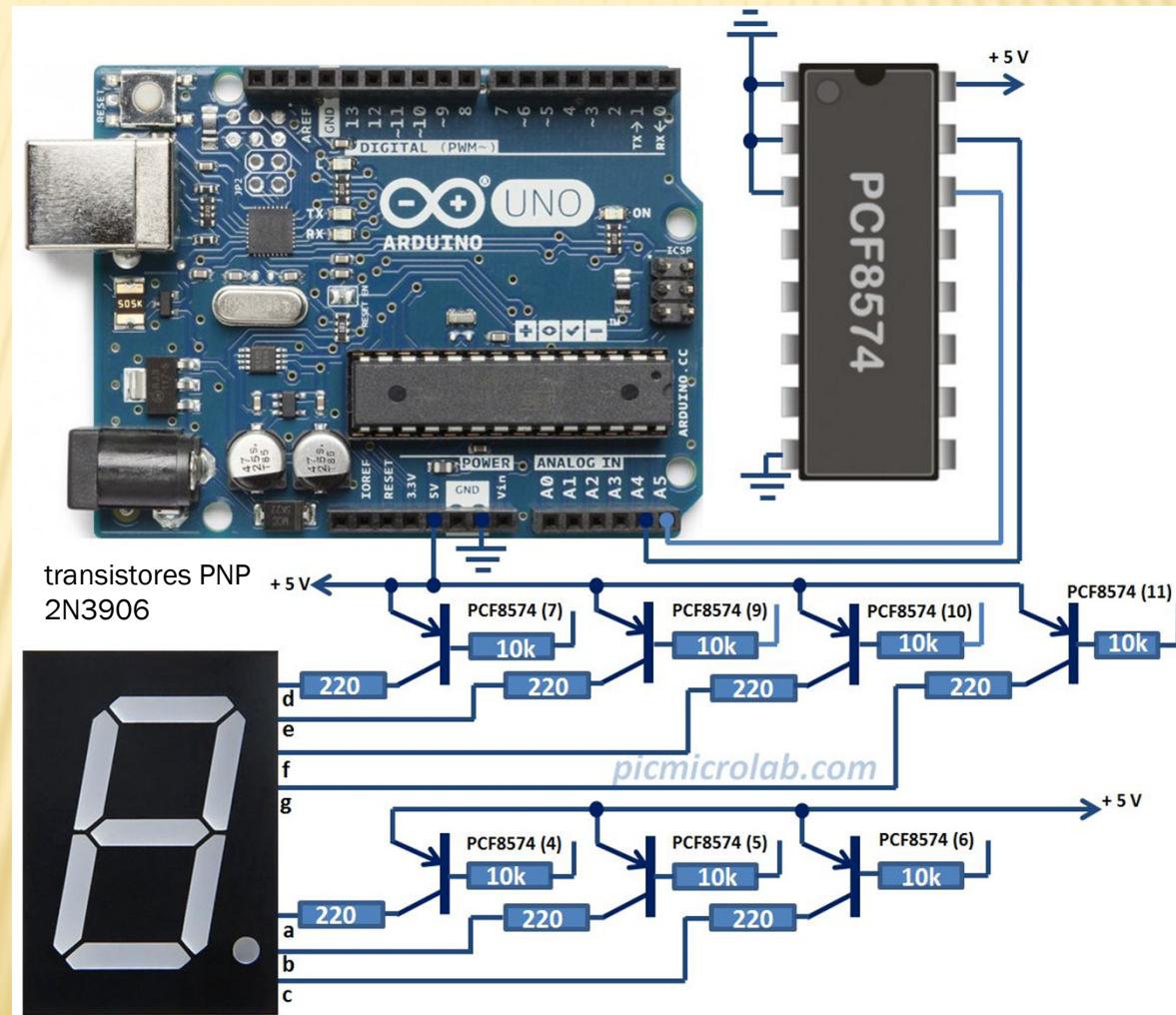
# DISPLAY DE 7 SEGMENTOS

Com dois pinos de E/S: interface de comunicação I2C:

Biblioteca padrão Wire  
foi usada para  
a rotina I2C...

O endereço do dispositivo foi definido como 000, conectando os pinos A0, A1 e A2 ao terra

Este projeto básico pode ser facilmente expandido para até oito displays de 7 segmentos, adicionando mais unidades PCF8574 (um endereço diferente para cada chip expansor de porta E/S).



# DISPLAY DE 7 SEGMENTOS

```
#include <Wire.h>

byte digsDec[] = { 0x40, 0xf9, 0x24, 0x30, 0x19, 0x12, 0x02, 0xf8, 0x00, 0x10 }, dig = 9;

void setup() {
    Wire.begin();                                     // agrega-se ao barramento I2C (ender. opc. p/ Mestre)
}

void inc(byte x) {
    if (++x > 9)
        x = 0;
}

void loop() {
    Wire.beginTransmission(0x20);                     // transmite ao PCF8574 (faz uso do barramento I2C)
    Wire.write(digsDec[inc(dig)]);                   // envia o código do dígito 1 ao display de 7 segmentos
    Wire.endTransmission();                          // para de transmitir (libera o barramento I2C)
    delay(1000);
}
```

# PROJETO 3: TACÔMETRO

