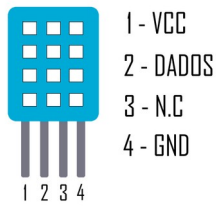
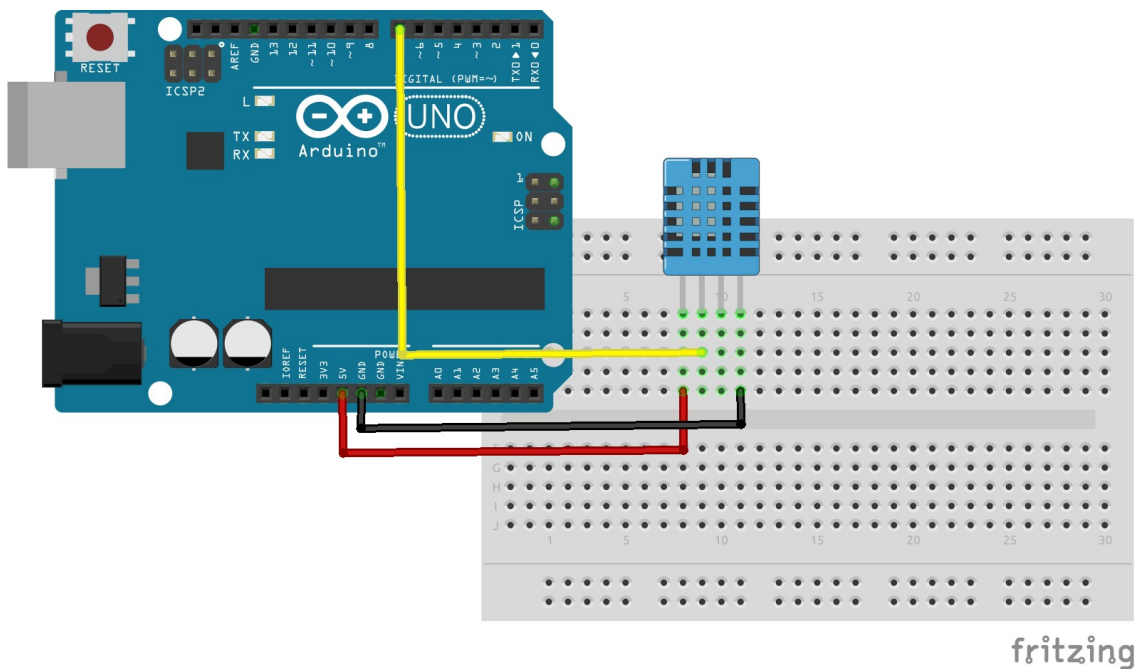


## Sensor de temperatura e umidade



Cuidado: inverter os pinos de fornecimento de energia irá queimar o sensor!

### Primeiro exemplo – temperatura e umidade no monitor serial

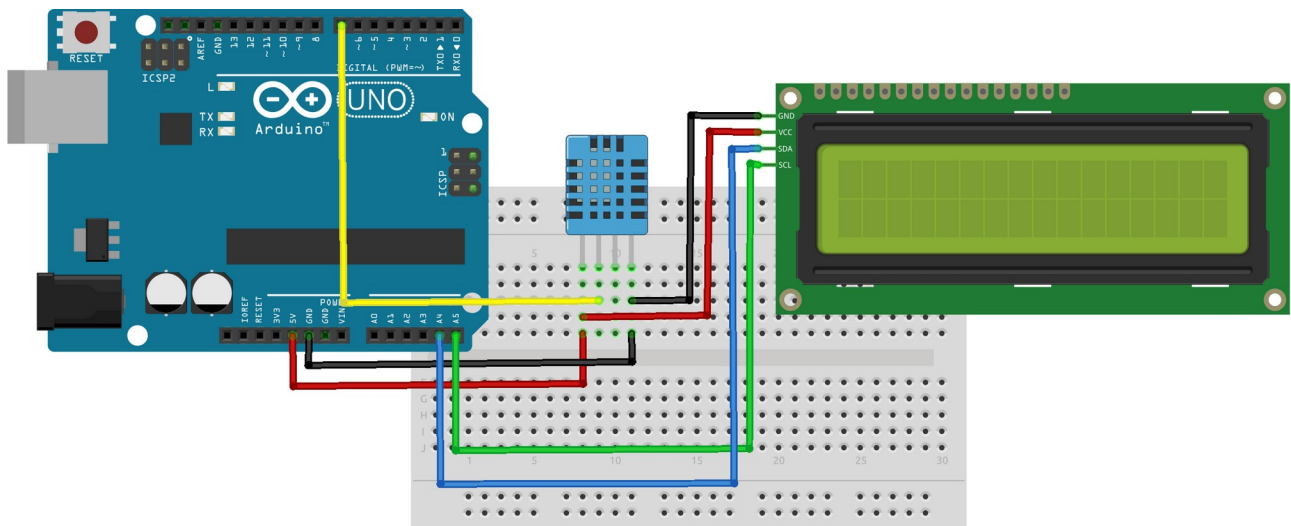


#### temperaturaUmidade

```
1 #include <dht11.h> //biblioteca para controlar o sensor
2 #define pinoSensor 7 //pino em que está ligado o sensor
3
4 dht11 sensor; //declaração de objeto para usar o sensor
5
6 void setup()
7 {
8     Serial.begin(9600); //inicia a comunicação serial com velocidade 9600bps
9     Serial.println("CyberRex - temperatura e umidade");
10    int leitura = sensor.read(pinoSensor); //inicializa o sensor
11 }
12
13 void loop()
14 {
15     Serial.print("Temperatura (C): ");
16     Serial.println((float)sensor.temperature, 2); //exibe a temperatura com 2 casas decimais
17     Serial.print("Umidade (%): ");
18     Serial.println((float)sensor.humidity, 2); //exibe a umidade com 2 casas decimais
19     delay(3000); //aguarda 3 segundos antes de nova leitura
20 }
```

Na linha 1 do código temos a inclusão da biblioteca (Sketch / Incluir Biblioteca / Adicionar biblioteca .ZIP). Na linha 2 a indicação do pino em que foi ligado o sensor (7, neste exemplo). Na linha 4 declaramos o objeto 'sensor', que é do tipo 'dht11' (a biblioteca incluída). Na função `setup()`, inicializamos a comunicação serial (linha 8), exibimos uma mensagem (linha 9) e inicializamos o sensor (linha 10). Na função `loop()` exibimos a mensagem "Temperatura (c)" na linha 15 e na linha 16 o valor (com duas casas decimais). As linhas 17 e 18 fazem o mesmo para a umidade (Umidade (%)) e seu valor, e a linha 19 impõe uma espera de 3 segundos antes de nova leitura.

## Segundo exemplo – temperatura e umidade no *display* lcd



fritzing

```
temperaturaUmidadeLCD $
```

```
1 #include <dht11.h> //biblioteca para controlar o sensor
2 #include <LiquidCrystal_I2C.h> //display LCD com comunicação I2C
3 #define pinoSensor 7 //pino em que está ligado o sensor
4
5 dht11 sensor; //declaração de objeto para usar o sensor
6 LiquidCrystal_I2C lcd(0x27,16,2); // display lcd no endereço 0x27 com 16 colunas 2 linhas
7
8 void setup()
9 {
10  lcd.init(); //inicia o display
11  lcd.clear(); //limpa o display
12  lcd.backlight(); //controla a luz
13  lcd.setCursor(4,0); //posiciona o cursor, coluna 4 da linha 0
14  lcd.print("CyberRex");
15  lcd.setCursor(0,1); //posiciona o cursor, coluna 0 da linha 1
16  lcd.print("Temperat/Umidade");
17  delay(2000); //aguarda 2 segundos exibindo a mensagem da CyberRex
18 }
19
20 void loop()
21 {
22  int leitura = sensor.read(pinoSensor); //inicializa o sensor
23  lcd.clear(); //limpa o display
24  lcd.setCursor(0,0); //posiciona o cursor, coluna 0 da linha 0
25  lcd.print("Temp.  : ");
26  lcd.print((float)sensor.temperature, 2);
27  lcd.setCursor(14,0); //posiciona o cursor, coluna 14 da linha 0
28  lcd.write(B11011111); //teste: lcd.write(223) e lcd.write(0xDF)
29  lcd.print("C");
30  lcd.setCursor(0,1); //posiciona o cursor, coluna 0 da linha 1
31  lcd.print("Umidade: ");
32  lcd.print((float)sensor.humidity, 2);
33  lcd.setCursor(15,1); //posiciona o cursor, coluna 15 da linha 1
34  lcd.print("%");
35  delay(3000); //aguarda 3 segundos antes de nova leitura
36 }
```



Neste segundo exemplo foi incluída uma biblioteca para controle do *display* LCD e o código foi modificado de forma que as mensagens sejam exibidas no *display* e não no monitor serial.

Na linha 1 foi mantida a biblioteca de controle do sensor. Na linha 2 foi incluída a biblioteca de controle do *display* Lcd (que já foi usada em outros experimentos). Na linha 3 ficou declarado o pino no qual ficará ligada a comunicação de dados do sensor (pino 7). Na linha 5 foi declarado o objeto para uso do sensor DHT11, de temperatura e umidade. Na linha 6 foi declarado o objeto para controle do *display*. Sendo uma comunicação serial I2C temos que informar o endereço (neste exemplo 0x27, em hexadecimal); também é informado o aspecto do *display* (neste caso, 16 colunas e 2 linhas).

Na função `setup()` temos:

- a inicialização do *display* (linha 10, função `init()`); a limpeza de seu *cache* de mensagens (linha 11, função `clear()`);
- ligar o LED de iluminação do *display* (linha 12, função `backlight()`);
- posicionar o cursor em uma certa posição (por meio da função `setCursor(4,0)`, a qual posiciona o cursor, ou seja, onde será ‘escrito’, na *coluna*, *linha* informadas – neste caso, *coluna* 4 e *linha* 0);
- na linha 14 solicitamos a exibição de um texto (por meio da função `print()`, para a qual é passado, entre aspas duplas - “”, o texto a ser exibido – neste caso, “CyberRex”);
- na linha 15 posicionamos o cursor na primeira *coluna* (*coluna* 0) da segunda *linha* (*linha* 1) do *display* (por meio da função `setCursor(0,1)`) – observar que todas as informações são indexadas a partir de 0 (zero), de forma que, quando queremos a ‘primeira’ informação (por exemplo a primeira *coluna*), ela será a de ‘número’ 0 (zero), a segunda será a de ‘número - ou índice’ 1 (um), e assim por diante;
- na linha 16 fornecemos mais informações a exibir (por meio do `print()`);
- finalmente, na linha 17 temos uma ‘espera’ de 2 segundos (`delay(2000)`) para que a mensagem inicial possa ser percebida.

Na função `loop()`, temos:

- na linha 22 foi declarada uma variável local do tipo inteira (`int`), chamada de leitura, a qual receberá o resultado da leitura (`read`) do sensor, e, por meio deste comando, irá inicializar o sensor para as leituras subsequentes;
- na linha 23 temos a limpeza do *cache* do *display* (`clear()`);
- na linha 24 posicionamos o cursor (`setCursor()`);
- na linha 25 exibimos a mensagem ‘Temp. :’ (com a função `print()`);
- na linha 26 será exibido o valor da temperatura lido pelo sensor, por meio do comando ‘`lcd.print((float)sensor.temperature, 2);`’: `lcd` é o objeto, `print` o comando (de exibição); o conteúdo que será exibido será a conversão para uma variável do tipo `float` (para números não inteiros), com duas casas decimais (2) do valor de temperatura lido pelo sensor (`sensor.temperature`);
- na linha 27 o curso é posicionado na *coluna* 14 da *linha* 0 (`setCursor(14,0)`);
- na linha 28 temos uma outra forma de exibir caracteres no *display*, o uso da função `write()`: a função `write` aceita caracteres para exibir no *display* nos formatos decimal (basta informar o número, por exemplo 223), hexadecimal (no qual o número é precedido de 0x, por exemplo 0xDF) e binário (no qual o número é precedido de B, por exemplo B11011111), que é o que foi utilizado – os caracteres, e seus códigos, são ‘escolhidos’ a partir do conjunto que o *display* é capaz de exibir, no próximo exemplo vamos explorá-los. O código escolhido vai fazer com que seja exibido no *display* o caractere correspondente ao símbolo de ‘graus’ (°);
- na linha 29 é solicitada a exibição da letra ‘C’ no *display*, complementando a informação de temperatura (portanto, teremos a exibição de °C, sendo o ° proveniente da linha 28 e o C da linha 29);
- entre as linhas 30 e 34 será realizada a mesma estratégia, de exibição de uma mensagem, de um valor, e de uma unidade (%), mas agora para exibição da umidade (`sensor.humidity`);
- na linha 35 fazemos com que ocorra uma espera de 3 segundos (`delay(3000)`) até a próxima leitura, após o que o ciclo recomeça na linha 22.

## Como ver todos os caracteres que o *display* consegue exibir?

Par isso, podemos fazer um *loop* que mostre os valores decimais entre 0 e 255 (pois são 256 combinações possíveis). Podemos aproveitar o *hardware* já montado e realizar o código que segue:

- **observação:** *nem todos os códigos geram caracteres, e, dependendo do chip que controla o display pode haver variações; veja a tabela na página seguinte para referência -*

```
todosOsCaracteresLCDserial
1 #include <LiquidCrystal_I2C.h> //display LCD com comunicação I2C
2
3 LiquidCrystal_I2C lcd(0x27,16,2); // display lcd no endereço 0x27 com 16 colunas 2 linhas
4
5 void setup()
6 {
7   lcd.init(); //inicia o display
8   lcd.clear(); //limpa o display
9   lcd.backlight(); //controla a luz
10 }
11
12 void loop()
13 {
14   for (int valor = 0; valor <256; valor++){
15     lcd.clear(); //limpa o display
16     lcd.print("Valor    = ");
17     lcd.print(valor);
18     lcd.setCursor(0,1);
19     lcd.print("Caractere = ");
20     lcd.write(valor); //exibe o valor atual
21     delay(1500);
22   }
23 }
```

A linha 1 inclui a biblioteca, a 3 declara o objeto *display* (no endereço 0x27, com 16 colunas e 2 linhas), a linha 7 inicializa o *display*, na linha 8 o *cache* é limpo e na linha 9 seu LED interno é acionado.

O laço de exibição dos caracteres ocorre entre as linhas 14 e 21:

- na linha 14 é declarado o laço (*for*), no qual uma variável inteira (*valor*) assumirá valores entre 0 e 255 (< 256, menor do que 256), variando de 1 em 1 (*valor++*);
- na linha 15 o *display* será limpo (*clear()*);
- na linha 16 será exibida a mensagem “Valor =” (note que, como não foi usado o *setCursor*, e como *display* acabou de ser ‘limpo’, a primeira posição de escrita é 0,0 – coluna 0 da linha 0, ou, primeira coluna da primeira linha);
- na linha 17 será exibido o conteúdo da variável ‘*valor*’ (como foi solicitado o uso da função *print()*, será exibido o conteúdo da variável – para exibir o caractere correspondente temos que utilizar a função *write()*, o que será realizado na linha 19);
- na linha 18 informamos que a próxima mensagem será exibida na primeira coluna (que é a 0) da segunda linha (que é a 1, pois começa a numeração em 0), o que é realizado por meio da função *setCursor(0,1)*;
- na linha 19 exibimos a mensagem ‘Caractere = ’;
- na linha 20 exibimos o caractere correspondente à variável ‘*valor*’. Note que, agora, vamos exibir o caractere e não o número que o representa;
- na linha 21 aguardamos um segundo e meio (*delay(1500)*) antes de prosseguir no laço de forma a podermos visualizar os caracteres no *display*.

Conjunto de caracteres para o *chip* controlador HD44780:

Higher 4bit Lower 4bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
xxxx0000		0	1	A	P	`	F		-	9	3	0	p
xxxx0001		!	1	A	Q	a	9		7	4	3	q	
xxxx0010		"	2	B	R	b	r		ı	ı	x	e	e
xxxx0011		#	3	C	S	c	s		ı	ı	e	e	o
xxxx0100		\$	4	D	T	d	t		ı	ı	k	ı	o
xxxx0101		%	5	E	U	e	u		•	ı	ı	ı	ı
xxxx0110		&	6	F	V	f	v		ı	ı	ı	ı	ı
xxxx0111		'	7	G	W	g	w		ı	ı	ı	ı	ı
xxxx1000		(	8	H	X	h	x		ı	ı	ı	ı	ı
xxxx1001		)	9	I	Y	i	y		ı	ı	ı	ı	ı
xxxx1010		*	:	J	Z	j	z		ı	ı	ı	ı	ı
xxxx1011		+	:	K	L	k	l		ı	ı	ı	ı	ı
xxxx1100		,	<	L	*	ı	ı		ı	ı	ı	ı	ı
xxxx1101		=	=	M	I	m	i		ı	ı	ı	ı	ı
xxxx1110		.	>	N	^	n	÷		ı	ı	ı	ı	ı
xxxx1111		/	?	O	_	o	+		ı	ı	ı	ı	ı

**Exemplo:** a) localizamos o caractere desejado (por exemplo o símbolo °); b) localizamos na coluna em que está o caractere desejado o Higher 4 bit (os quatro bits de mais alta ordem), neste caso 1011; c) localizamos na linha em que está o caractere desejado o Lower 4 bit (os bits de mais baixa ordem), neste caso 1111; d) montamos o código colocando primeiro os 4 bits de mais alta ordem (1011) e depois os de baixa ordem (1111) juntos: 10111111 (que é o decimal 223 e o hexadecimal 0xDF)<sup>1</sup>.

1 A calculadora do windows faz conversões de base; você também pode ver online em: <https://clevert.com.br/t-pt-br/base-convert>