



## Terminal série sur Arduino

Un moniteur ou terminal est un écran/clavier relié à un microcontrôleur pour interagir avec un utilisateur. Les microcontrôleurs ont utilisés depuis leur début la norme série RS232 (en 5 volts) pour communiquer avec un PC (programme Teraterm, Telnet) et maintenant avec des tablettes via BlueTooth. USB a défini des "com ports" qui émulent des prises séries, avec des numéros de ports qui fonctionnent en général.

Arduino utilise un com port pour charger le programmes, et quelques fonctions Arduino permettent d'utiliser cette ressources. Sur Energia et Pinguino, il y a quelques différences partiellement citées en fin de ce document.

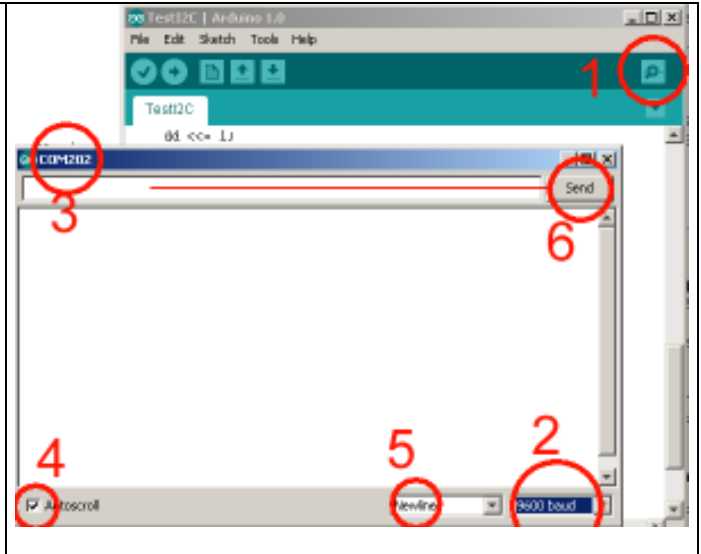
### Appel du terminal Arduino

On appelle le terminal en cliquant sur l'icône (1). Une fenêtre s'ouvre et un reset est envoyé pour redémarrer le programme. Cette fenêtre se ferme si on lance une nouvelle compilation.

Le driver qui permet la communication avec le PC pour le chargement a été initialisé à une vitesse qui est affichée en (2). C'est cette vitesse qu'il faut spécifier pour mettre en route les transferts.

En (3) on a le numéro du com port.

En (4) et (5) des options de mise en page (voir note 1) On reparle plus loin du (6).



### Initialisation et écriture

L'instruction `Serial.begin(9600)` ; définit dans le setup la vitesse des transferts que notre programme va effectuer.

On peut alors écrire des chaînes de caractères (strings) et des nombres d'une façon assez peu commode.

`Serial.print("texte")` // affiche chaine sans saut de ligne

`Serial.print(valeur)` ; // affiche une valeur en décimal

`Serial.print(valeur, DEC)` ; `Serial.print(valeur, HEX)` ; `Serial.print(valeur, BIN)` ;

- attention, les zéros non significatifs sont effacés: (0b00010100,BIN) est affiché 10100 -

`Serial.println()`: l'affichage est suivi d'un saut de ligne

Une variante pour introduire un saut de ligne est d'utiliser le caractère `\n`,

Ceux qui ont appris le C++ sur écran connaissent une plus grande richesse de signes, non supportés par Arduino.

```
Serial.print(27, BIN) ;
```

```
Serial.print("  ") ;
```

```
Serial.print(val, DEC) ;
```

L'affichage d'une variable dépend de son type

```
char x = 'A' Serial.print(x) ; → A Serial.print(val, DEC) ; → 65
```

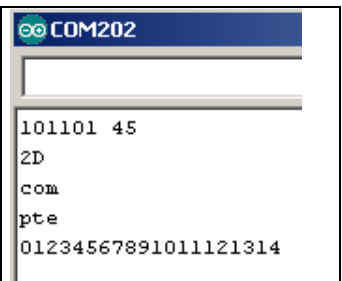
```
byte y = 'A' Serial.print(y) ; → 65
```

```
unsigned char z = 'A' Serial.print(z) ; → 65
```

```
int y
```

Code ASCII (table complète sous [www.didel.com/AsciiCode.gif](http://www.didel.com/AsciiCode.gif))

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	{null}	32	20	040	#32: Space	64	40	100	#64: @	96	60	140	#96: `		
1	1	001	SOH	{start of heading}	33	21	041	#33: !	65	41	101	#65: A	97	61	141	#97: a		
2	2	002	STX	{start of text}	34	22	042	#34: "	66	42	102	#66: B	98	62	142	#98: b		
3	3	003	ETX	{end of text}	35	23	043	#35: #	67	43	103	#67: C	99	63	143	#99: c		
4	4	004	EOF	{end of transmission}	36	24	044	#36: \$	68	44	104	#68: D	100	64	144	#100: d		
5	5	005	ENO	{enquiry}	37	25	045	#37: %	69	45	105	#69: E	101	65	145	#101: e		
...																		
16	10	020	DLE	{data link escape}	48	30	060	#48: 0	80	50	120	#80: P	112	70	160	#112: p		
17	11	021	DC1	{device control 1}	49	31	061	#49: 1	81	51	121	#81: Q	113	71	161	#113: q		
18	12	022	DC2	{device control 2}	50	32	062	#50: 2	82	52	122	#82: R	114	72	162	#114: r		
19	13	023	DC3	{device control 3}	51	33	063	#51: 3	83	53	123	#83: S	115	73	163	#115: s		

<pre>//TestSerie.ino Différents formats void setup() {   Serial.begin(9600); } int val = 45 ; void loop() {   Serial.print(val,BIN);   Serial.print(" ");   Serial.print(val,DEC);   Serial.print('\n');</pre>	<pre>Serial.println(val,HEX); Serial.println("com\npte "); for (byte i=0;i&lt;15;i++) {   Serial.print(i); } while (1) {}</pre> <p>Les zéros non significatifs sont supprimés</p>	
--	---	--

A noter que BIN=2 DEC=10 et HEX=16.

Voulez avoir le résultat en base 3? (28,3) affiche 1001 ( les poids en base 3 sont 1 3 9 27 ...)

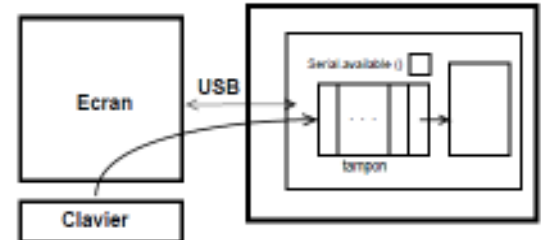
### Envoyer depuis le clavier

Le code Ascii des caractères que l'on tape sur le clavier va dans une mémoire tampon et la fonction `Serial.available()` donne le nombre de bytes en attente dans le tampon.

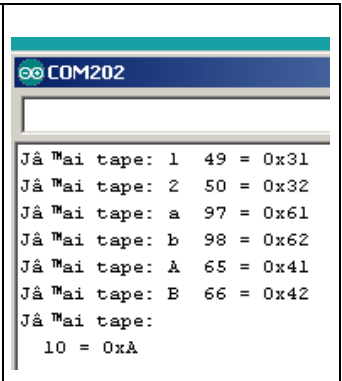
La ligne supérieure du terminal (6) reçoit les caractères à transmettre, mais il faut

- cliquer le début de la ligne pour avoir le curseur
- taper le ou les caractères à transmettre
- cliquer sur Send ou touche Return

Lire le caractère avec `Serial.read()` ;



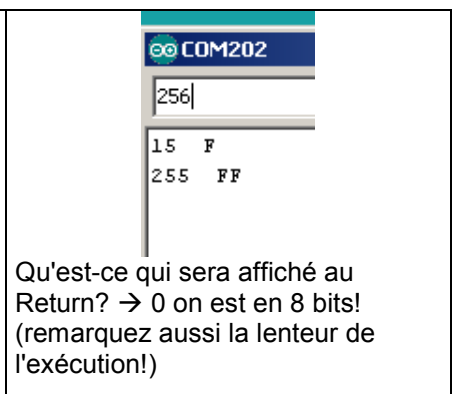
Exemple:

<pre>//TestSerialread.ino void setup() {   Serial.begin(9600); } char codeRecu ; void loop () {   if (Serial.available() &gt; 0) {     codeRecu = Serial.read();     Serial.print ("J'ai tape: "); Serial.print (codeRecu);     Serial.print (" "); Serial.print (codeRecu,DEC);     Serial.print (" = 0x"); Serial.print (codeRecu,HEX);     Serial.println ();   } }</pre>	
--	--

La **cadace** en **bas d'écran** (CR et son code) disparaît si vous mettez le mode "no line ending" en (5)

### Lire un paramètre

Ce qui est essentiel, pour le déverminage en pour plusieurs applications, c'est de pouvoir taper un nombre et modifier une variable. C'est documenté dans un recoin bien caché d'Arduino, que l'on trouve en tapant "Arduino parse" ou "parseint"

<pre>// TestParse.ino Affiche le nombre tapé // essayez 15, -1, 256, ..., changez de type void setup() {   Serial.begin(9600); } byte valeurIn; void loop() {   if (Serial.available() &gt; 0) {     valeurIn = Serial.parseInt();     Serial.print(valeurIn); Serial.print(" ");     Serial.println(valeurIn,HEX);   } }</pre>	 <p>Qu'est-ce qui sera affiché au Return? → 0 on est en 8 bits! (remarquez aussi la lenteur de l'exécution!)</p>
---	---

A noter que l'on peut aussi utiliser un while, les caractères sont envoyés tous en même temps.

### Lire plusieurs paramètres

Les nombres que l'on entre peuvent être séparés par un espace ou une virgule, et terminés par le Return ou Send (mais s'il y a un espace avant le return, c'est comme si on a un nombre de plus).

Dans le programme, on lit les paramètres en suivant:

```
if (Serial.available() > 0) {
  red = Serial.parseInt();
  green = Serial.parseInt();
  blue = Serial.parseInt();
}
```

**Serial.write ()**

Cette fonction semble identique à Serial.print, mais elle rend la longueur de la chaîne de caractère transmise.

**Gestion du tampon**

Des fonctions de gestion du tampon sont utiles pour des applications dans lesquelles il faut récupérer des comportements inattendus. Chercher serial.Flush

**Terminal sur Pinguino**

Ce document ancien correspond au Pinguino9-05: [www.didel.com/diduino/SerialMonitor.pdf](http://www.didel.com/diduino/SerialMonitor.pdf)

**Terminal sur Energia**

La compatibilité avec Arduino semble bonne.

**Linux**

Les transferts avec Energia sous Linux posent parfois problème.

jdn 131113/16/20140518