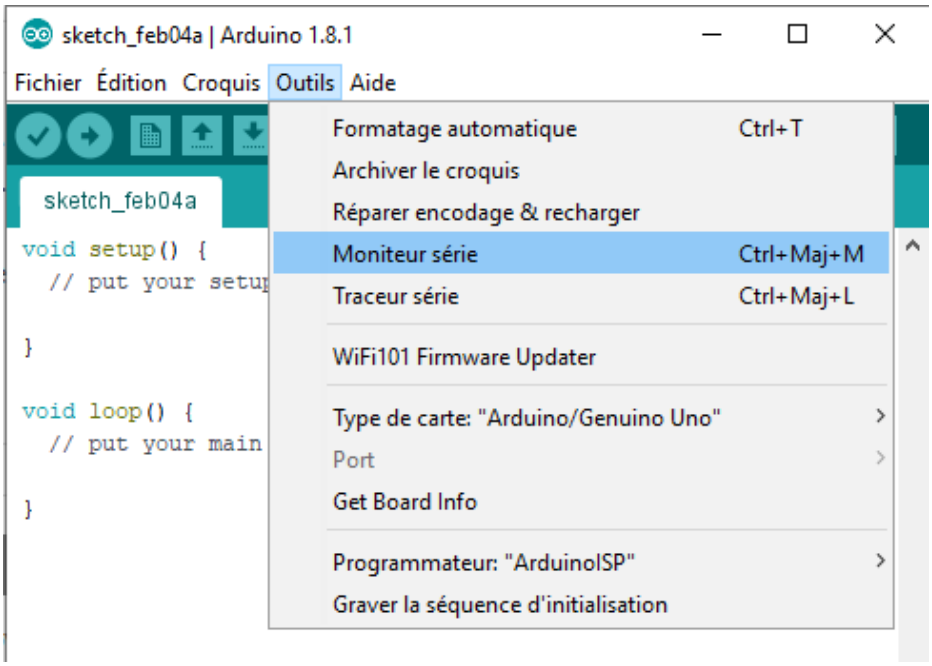
Le moniteur Série

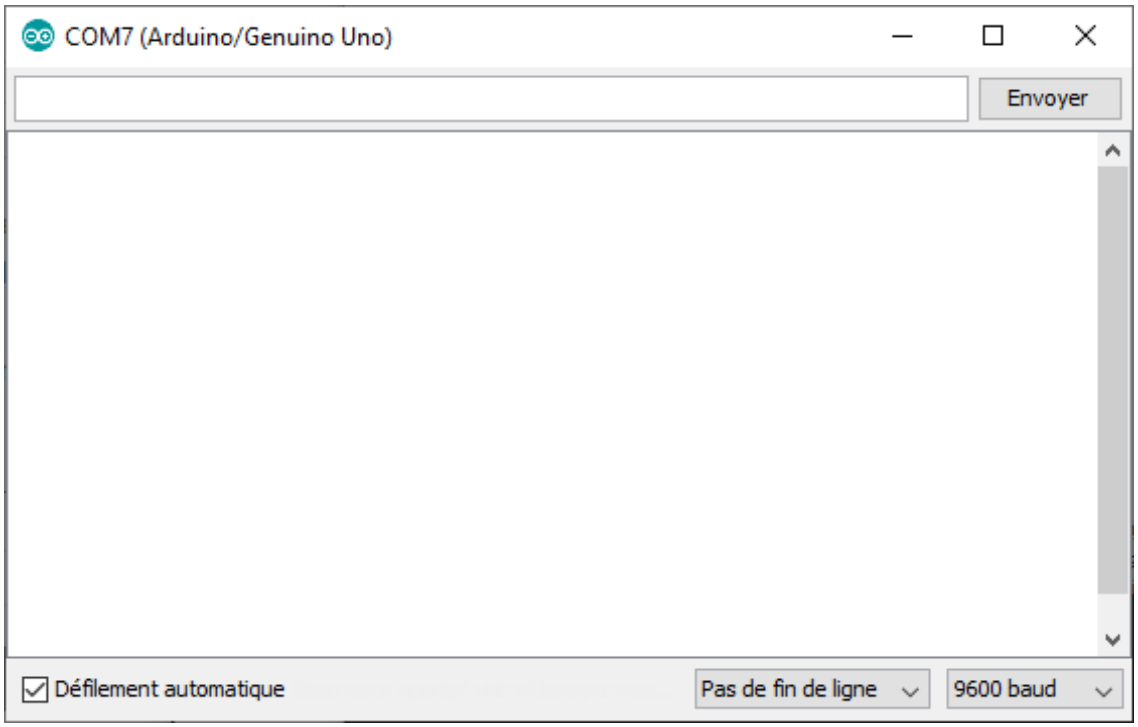
Le logiciel Arduino IDE dispose d'un moniteur série, qui permet de recevoir et d'envoyer des informations via une liaison série. Il est particulièrement intéressant pour les tests des programmes puisqu'il permet d'afficher les valeurs des variables, les états logiques des entrées et des sorties de l'Arduino, etc…

Le moniteur série est accessible depuis le logiciel Arduino en cliquant sur la loupe en haut à droite de la fenêtre du logiciel: 

Ou à partir du menu **Outils/Moniteur série :**



Une nouvelle fenêtre s'ouvre alors :



La fenêtre est composée de deux zones blanches d'exploitation et de plusieurs commandes :

- la grande zone au centre est la zone d'affichage des données reçues par le moniteur et donc envoyées par l'Arduino,

- la petite zone rectangulaire, en haut est une zone de saisie des données à envoyer à l'Arduino à l'aide du bouton "Envoyer" qui se trouve à sa droite. C'est le programme du microcontrôleur qui génère la demande d'envoi de données par l'utilisateur et qui s'occupe de traiter les informations reçues,

- la case cochée "Défilement automatique" permet d'arrêter le défilement des données retournées par l'Arduino si on la décoche,

- une liste déroulante permettant le réglage du mode de défilement des informations (Pas de fin de ligne, Nouvelle ligne, …),

- une liste déroulante permettant de régler le débit de transmission des données par le port série en bauds.

Remarques :

- Le baud est une unité de mesure utilisée dans le domaine des télécommunications en général, et dans le domaine informatique en particulier. Le baud est l'unité de mesure du nombre de symboles transmissibles par seconde.

Le terme "baud" provient du patronyme d'Émile Baudot, l'inventeur du code Baudot utilisé en télégraphie.

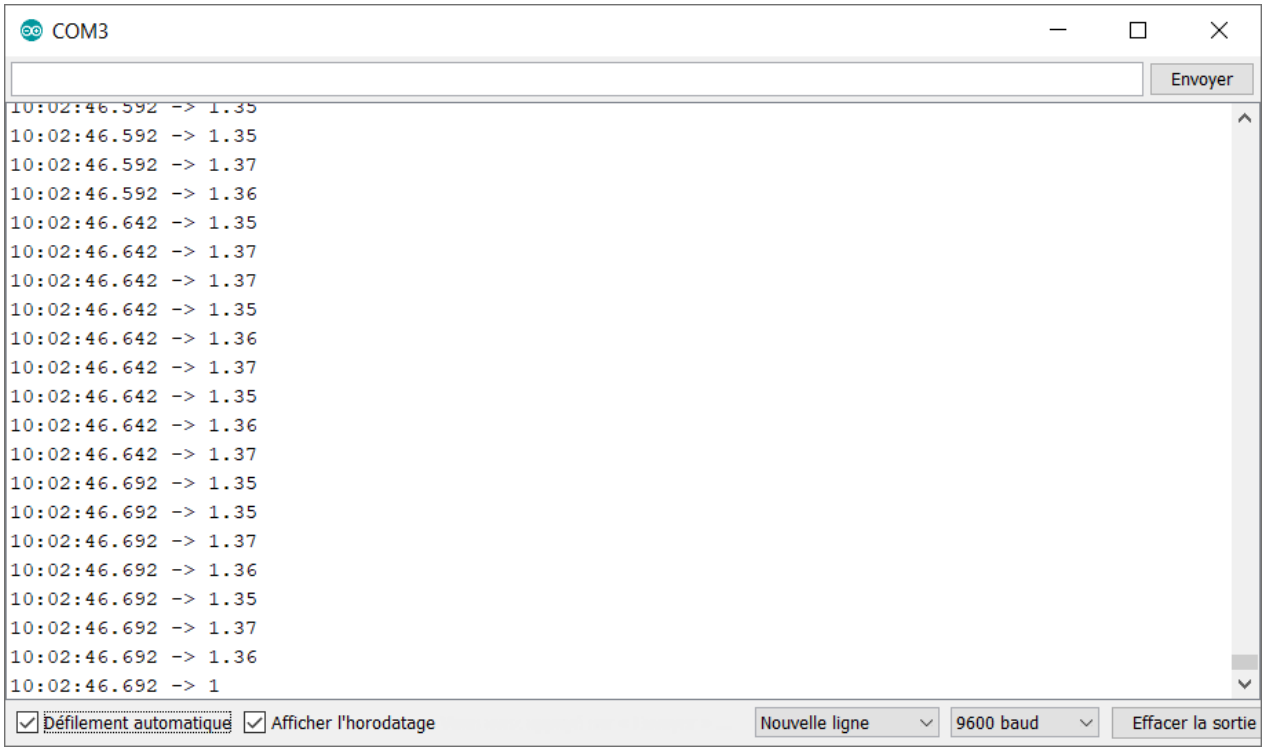
Il ne faut pas confondre le baud avec le bps ou bit par seconde, ce dernier étant l'unité de mesure du nombre d'information effectivement transmise par seconde. Il est en effet souvent possible de transmettre plusieurs bits par symbole. La mesure en bps de la vitesse de transmission est alors supérieure à la mesure en baud.

- La valeur par défaut est de 9600 bauds. La carte Arduino peut monter jusqu'à une cadence de 115200 bauds. Mais un débit de communication de 9600 caractères (chaque caractère étant codé sur 8 bits, soit 1 octet) par seconde (**9600 bauds**) est généralement suffisant.

- Dans les dernières versions du logiciel **Arduino IDE**, deux nouvelles commandes ont été ajoutés au moniteur série :

. "Afficher l'horodatage " (affiche l'heure de la transmission des données)

. "Effacer la sortie " (efface la zone d'affichage des données reçues par le moniteur série)



Pour utiliser le moniteur série, le programme téléversé dans la mémoire de l'Arduino doit établir la liaison série.

Pour cela, dans la fonction Setup() du programme, on utilise la fonction **begin()** de la classe "**Serial**" :

```
void setup() {
  Serial.begin(9600);
}
```

Le port série de la carte **Arduino est alors configuré à 9600 bauds.**

<u>Important</u> :

- Le microcontrôleur et le moniteur série doivent être configurés sur le même débit de transmission.

- Le programme du microcontrôleur est réinitialisé à l'ouverture du moniteur série.

La classe "**Serial**" regroupe toutes les fonctions utiles à la gestion d'un port série. Son travail est de gérer le traitement des données d'une communication série entre le moniteur série et le périphérique Arduino.

Les principales fonctions de la classe Serial :

. **begin();** (Configure la vitesse de transmission du port série)

. **print();** (Envoie une donnée sous forme de chaine de caractères sur le port série)

. **println();** (Envoie une donnée sur le port série et fait un saut à la ligne)

. **write();** (Ecrit des données binaires sur le port série. Ces données sont envoyées comme une série d'octets)

. **read();** (Lit les données contenues dans la mémoire tampon ou "buffer" du port série)

. **flush();** (Vide la mémoire tampon de la liaison série)

. **available();** (Donne le nombre d'octets ou caractères disponibles pour lecture dans la file d'attente ou "buffer" du port série.

1. Envoi de données de l'Arduino vers le moniteur série

1.1 La fonction Serial.print()

Maintenant que la liaison série est établie, il est possible d'envoyer des données depuis la carte Arduino vers le moniteur série avec la fonction "**print()**" de la classe "**Serial**".

On peut envoyer différents types d'informations :

. Envoie d'une chaîne de caractères, sans saut de ligne à la fin : **Serial.print("Test");**

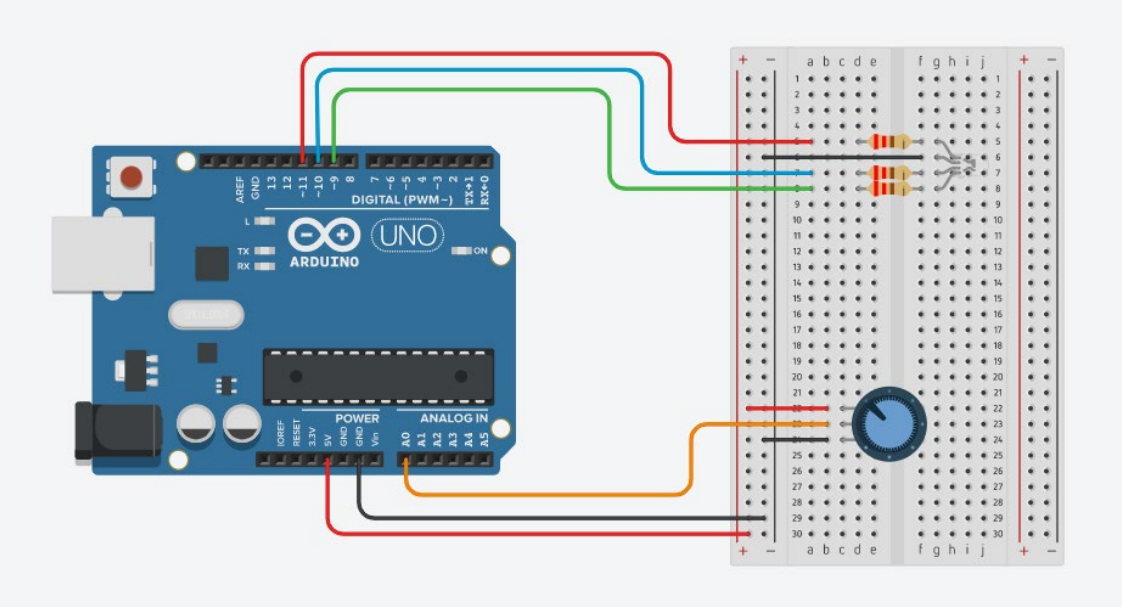
. Envoie d'une chaîne de caractères, avec saut de ligne à la fin : **Serial.println("Test");**

. Envoie de la valeur d'une variable, sans saut de ligne à la fin : **int a = 3;**
Serial.print(a);

Exemple d'application d'envoi de données de l'Arduino vers le moniteur série :

Voici un circuit qui servira de support à tous les exemples d'application de ce document.

Il composé d'une DEL RVB dont les DELs rouge, verte et bleue sont respectivement connectés aux sorties, 11, 9 et 10 de l'Arduino et d'un potentiomètre, dont le "point milieu" est relié à l'entrée analogique A0 du microcontrôleur :

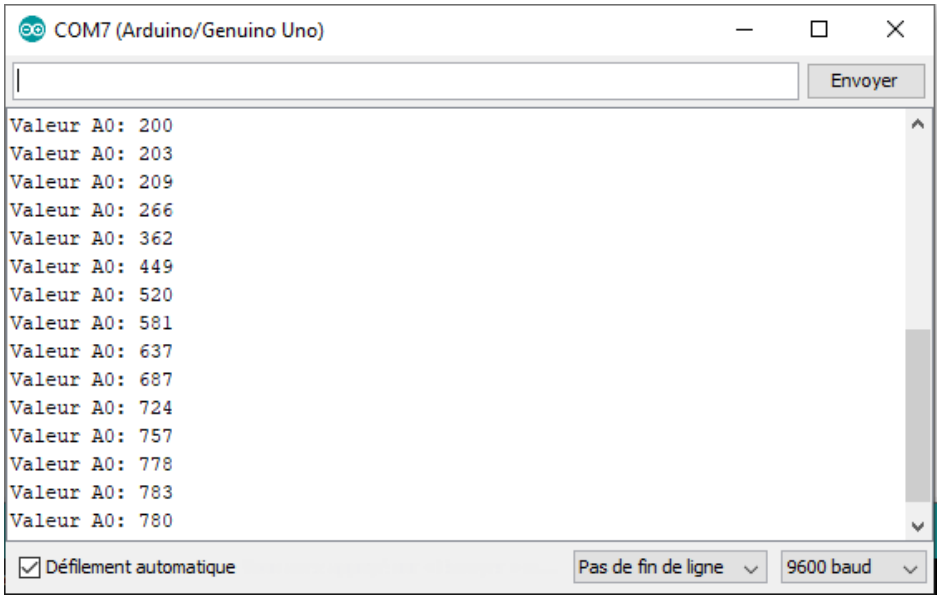


Le programme (nommé "AnalogRead.ino" dans le dossier "Codes/INO") qui sera téléversé dans la mémoire de l'Arduino affiche la valeur de l'entrée analogique **A0** dans le moniteur série (valeur entre 0 et 1023 en fonction de la position du "point milieu" du potentiomètre). Les données ne sont envoyées que si la valeur lue est différente de la précédente (écart de plus de 2 unités entre les valeurs lues) :

AnalogRead

```
const int PinPOT = A0;
int ValPot = 0;
int OldValPot =0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  ValPot = analogRead(PinPOT);
  if (abs(ValPot - OldValPot)>2) {
    Serial.print("Valeur A0: ");
    Serial.println(ValPot);
    OldValPot = ValPot;
    delay(100); }
}
```

Et le résultat dans le moniteur série :



1.2 La fonction Serial.write()

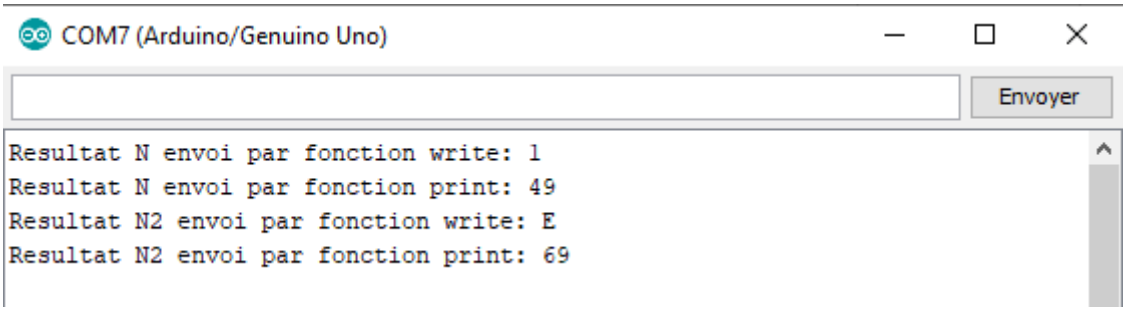
Il est également possible d'utiliser la fonction **"Serial.write()"**, pour envoyer des données sur la liaison série.

Si dans le cas de l'envoi de chaines de caractères, il n'y a pas de différence avec la fonction **"Serial.print()"**, l'envoi de nombres décimaux ou hexadécimaux ne donnent pas le même résultat suivant la fonction utilisée, comme le montre le code suivant (nommé "Write.ino" dans le dossier "Codes/INO") :

Write

```
int N=49;
int N2=0x45;

void setup() {
  Serial.begin(9600);
  Serial.write("Resultat N envoi par fonction write: "); Serial.write(N); Serial.println("");
  Serial.print("Resultat N envoi par fonction print: "); Serial.println(N);
  Serial.write("Resultat N2 envoi par fonction write: "); Serial.write(N2); Serial.println("");
  Serial.print("Resultat N2 envoi par fonction print: "); Serial.println(N2);
}

void loop() {

}
```

Résultats dans le moniteur série :



La fonction **"Serial.print()"** affiche bien les valeurs des variables N et N2 (0x45 en hexadécimal est égal à 69 en décimal) car ces variables sont envoyées comme des chaines de caractères, chaque caractère étant codé selon le code ASCII.

La fonction **"Serial.write()"**, elle, écrit des données binaires sur le port série. Ces données sont envoyées comme une série d'octets.

Si les données à transmettre sont des chaines de caractères, les données binaires transmises sont les codes ASCII des caractères. Mais dans le cas de nombre, les données envoyées sont le résultat de la conversion en binaire de ces nombres.

Ainsi la variable N est transmise sous la forme : **00110001** (49 en décimal) qui est le code ASCII du chiffre **"1"** (1 est donc affiché), et la variable N2 sous la forme : **01000101** (69 en décimal ou 0x45 en hexadécimal), soit le code ASCII de la lettre **"E"** (E est donc affiché).

2. Envoi de données du moniteur série vers l'Arduino

L'Arduino peut également recevoir des données depuis le moniteur série.

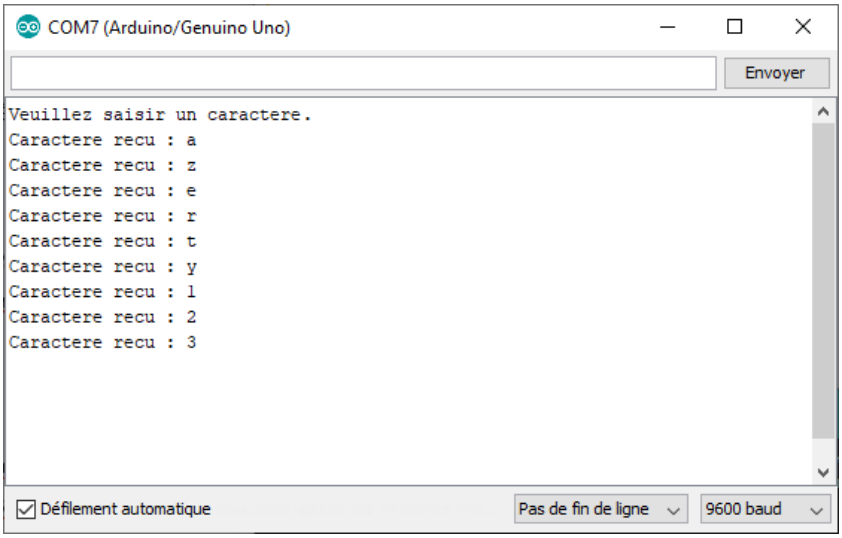
2.1 Programme pour la réception d'un caractère

Caractere

```
void setup() {
  Serial.begin(9600);
  Serial.println("Veuillez saisir un caractere.");
}

void loop() {
  if (Serial.available() > 0) {
   char c = Serial.read();
   Serial.print("Caractere recu : ");
   Serial.println(c);
}
}
```

Dans ce programme (nommé "Caractere.ino" dans le dossier "Codes/INO"), on demande à l'utilisateur de saisir un caractère dans le moniteur série. Une fois, la saisie effectuée, le nombre d'octets (caractères) disponible pour lecture dans la file d'attente du port série n'est plus nul (**Serial.available() > 0**), le caractère envoyé est alors lu à l'aide de la fonction **"read()"** et stocké dans la variable **"c"** définie comme étant un caractère.

La variable **"c"** est alors affichée dans le moniteur série :



<u>Remarques</u> :

- L'instruction "**char c**" déclare une variable, "**c**" d'un octet de mémoire (8 bits) qui contient une valeur correspondant à un caractère. Les caractères uniques sont écrits entre des guillemets simples, comme ceci : **'A'**

- Les caractères sont stockés de la même façon que les nombres : à chaque caractère correspond une valeur numérique comprise entre 0 et 127 (code ASCII). Ceci signifie également qu'il est possible de faire des opérations sur les caractères, dans lesquelles la valeur ASCII du caractère est utilisée (par exemple 'A'+1 a la valeur 66, car la valeur ASCII de la lettre capitale A est 65).

- Si la donnée envoyée contient plus d'un caractère (par exemple, le nombre "123"), la donnée est considérée comme une suite de 3 caractères qui seront affichés à la suite tant que la fonction "**Serial.available()**" retournera une valeur non nulle.

- Dans le code ASCII, chaque caractère est codé sous 7 bits (valeur numérique comprise entre 0 et 127 en décimal) et les accents ne sont pas pris en charge. C'est pourquoi, les messages affichés dans le moniteur série, par l'instruction "**Serial.print()**", sont envoyés sans les accents ("caractere" sans le "è"), sinon voici ce qui est affiché :

- Le code ASCII (American Standard Code for Information Interchange) est la façon standard de coder un texte numériquement :

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		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Le code ASCII ne contient pas de caractères accentués parce qu'il a été mis au point pour la langue anglaise. Pour coder ce type de caractère, il faut recourir à un autre code. Le code ASCII a donc été étendu à 8 bits (un octet) pour pouvoir coder plus de caractères (on parle d'ailleurs de code ASCII étendu...).

Ce code attribue les valeurs 0 à 255 aux lettres majuscules et minuscules, aux chiffres, aux marques de ponctuation et aux autres symboles :

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	|	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	⊛	249	·
137	ë	153	Ö	170	¬	186	╢	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	▉	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Source: www.LookupTables.com

- il est possible d'afficher des caractères avec accent dans le moniteur série en ajoutant une fonction aux programmes Arduino :

```
void PrintAccents(String TEXTE, boolean CR) {
 int CAR;
 for (byte I=0; I < TEXTE.length(); I++) {
    if (byte (TEXTE[I]) == 195)
      {CAR = int(TEXTE[++I]);
       switch (CAR) {
         case -68 : {Serial.print(char (252)); break;} // ü
         case -69 : {Serial.print(char (251)); break;} // û
         case -71 : {Serial.print(char (249)); break;} // ù
         case -74 : {Serial.print(char (246)); break;} // ö
         case -76 : {Serial.print(char (244)); break;} // ô
         case -81 : {Serial.print(char (239)); break;} // ï
         case -82 : {Serial.print(char (238)); break;} // î
         case -85 : {Serial.print(char (235)); break;} // ë
         case -86 : {Serial.print(char (234)); break;} // ê
         case -87 : {Serial.print(char (233)); break;} // é
         case -88 : {Serial.print(char (232)); break;} // è
         case -89 : {Serial.print(char (231)); break;} // ç
         case -92 : {Serial.print(char (228)); break;} // ä
         case -94 : {Serial.print(char (226)); break;} // â
         case -96 : {Serial.print(char (224)); break;} // à
         case -126 : {Serial.print(char (194)); break;} // Â
         case -128 : {Serial.print(char (192)); break;} // À
         case -108 : {Serial.print(char (212)); break;} // Ô
         case -114 : {Serial.print(char (206)); break;} // Î
         case -118 : {Serial.print(char (202)); break;} // Ê
         case -119 : {Serial.print(char (201)); break;} // É
         case -120 : {Serial.print(char (200)); break;} // È
         case -121 : {Serial.print(char (199)); break;}}} // Ç
    else Serial.print(TEXTE[I]);}
 if (CR) Serial.println(); }
```

Et d'utiliser la fonction "**PrintAccents()**" à la place de "**Serial.Print()**" pour afficher un message comportant des accents dans le moniteur série.

Exemple (programme "Accents.ino" dans le dossier "Codes\INO") :

Accents

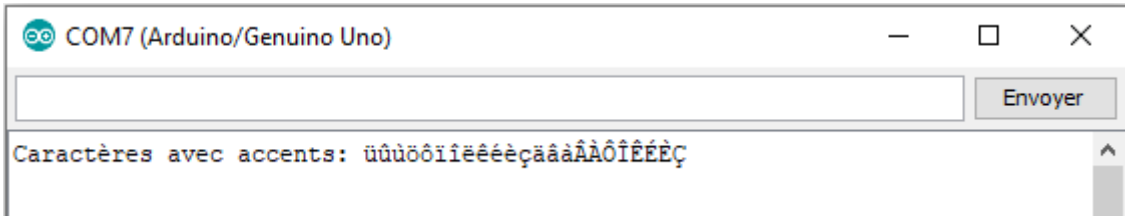
```
void PrintAccents(String TEXTE,boolean CR) {
  int CAR;
  for (byte I=0; I < TEXTE.length(); I++) {
       if (byte (TEXTE[I]) == 195)
          {CAR = int(TEXTE[++I]);
           switch (CAR) {
             case -68 : {Serial.print(char (252)); break;} // ü
             case -69 : {Serial.print(char (251)); break;} // û
             case -71 : {Serial.print(char (249)); break;} // ù
             case -74 : {Serial.print(char (246)); break;} // ö
             case -76 : {Serial.print(char (244)); break;} // ô
             case -81 : {Serial.print(char (239)); break;} // ï
             case -82 : {Serial.print(char (238)); break;} // î
             case -85 : {Serial.print(char (235)); break;} // ë
             case -86 : {Serial.print(char (234)); break;} // ê
             case -87 : {Serial.print(char (233)); break;} // é
             case -88 : {Serial.print(char (232)); break;} // è
             case -89 : {Serial.print(char (231)); break;} // ç
             case -92 : {Serial.print(char (228)); break;} // ä
             case -94 : {Serial.print(char (226)); break;} // â
             case -96 : {Serial.print(char (224)); break;} // à
             case -126 : {Serial.print(char (194)); break;} // Â
             case -128 : {Serial.print(char (192)); break;} // À
             case -108 : {Serial.print(char (212)); break;} // Ô
             case -114 : {Serial.print(char (206)); break;} // Î
             case -118 : {Serial.print(char (202)); break;} // Ê
             case -119 : {Serial.print(char (201)); break;} // É
             case -120 : {Serial.print(char (200)); break;} // È
             case -121 : {Serial.print(char (199)); break;}}} // Ç
       else Serial.print(TEXTE[I]);}
  if (CR) Serial.println(); }

void setup() {
  Serial.begin(9600);
  PrintAccents("Caractères avec accents: ",0);
  PrintAccents("üûùöôïîëêéèçäâàÂÀÔÎÊÉÈÇ",1);
  }

void loop() {
  }
```

En fait pour l'Arduino, les caractères, qu'ils soient accentués ou pas, sont codés sur 8 bits de manière signée, c'est-à-dire de -127 à 127 en décimale, alors que dans le code ASCII étendu, Les caractères accentués sont codés de 128 à 255.

La fonction "**PrintAccents()**" va analyser chaque caractère de la chaine qui doit être envoyée dans le moniteur série, à l'aide d'une boucle "**for**". Si le caractère n'est pas accentué, il est affiché tel quel, sinon (" **if (byte (TEXTE[I]) == 195)** ") en fonction de la valeur de sa conversion en nombre décimal ("**CAR = int(TEXTE[++I])** "), on affiche le caractère correspondant en utilisant son code en ASCII étendu (de 128 à 255).

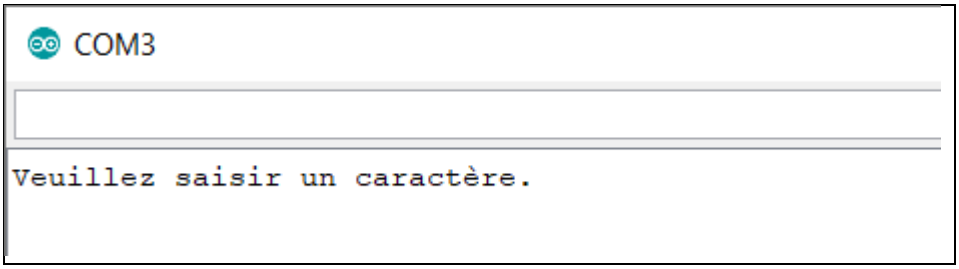
Un retour à la ligne est effectué si la valeur de CR est égale à 1.

Résultat dans le moniteur série :



Edit:

Avec les nouvelles versions du logiciel Arduino IDE (à partir de 1.8.10), il est maintenant possible d'afficher des messages avec des accents dans le moniteur série :



Et la fonction "**PrintAccents()**" ne peut être utilisée qu'avec les versions précédentes du logiciel Arduino IDE.

Exemple d'application du programme de réception d'un caractère :

Avec notre circuit, support des exemples d'application, le programme (nommé "LedRVB.ino" dans le dossier "Codes/INO") qui sera téléversé dans la mémoire de l'Arduino demande à l'utilisateur de saisir un caractère :

. '**R**', '**V**' ou '**B**' pour allumer respectivement la DEL rouge, verte ou bleue

. '**0**' pour éteindre les DELs

Le caractère envoyé est lu et l'action correspondante est effectuée. Le programme ne fait rien si le caractère lu n'est pas un des caractères attendus.

LedRVB

```
const int PinLEDR = 11;
const int PinLEDB = 10;
const int PinLEDV = 9;
const int PinPOT = A0;

const char LedR = 'R';
const char LedV = 'V';
const char LedB = 'B';
const char Off = '0';
char c=' ';

void setup() {
  Serial.begin(9600);
  pinMode (PinLEDR, OUTPUT);
  pinMode (PinLEDB, OUTPUT);
  pinMode (PinLEDV, OUTPUT);
  Serial.println ("Pour allumer la DEL rouge, envoyer: R");
  Serial.println ("Pour allumer la DEL verte, envoyer: V");
  Serial.println ("Pour allumer la DEL bleue, envoyer: B");
  Serial.println ("Pour eteindre les DELs, envoyer: 0");
}

void loop() {
  if (Serial.available() > 0) {
  c = Serial.read();
  }

  if (c == LedR) {
    Serial.println (c);
    digitalWrite(PinLEDR,1);
    digitalWrite(PinLEDV,0);
    digitalWrite(PinLEDB,0);
    }
  if (c == LedV) {
    Serial.println (c);
    digitalWrite(PinLEDR,0);
    digitalWrite(PinLEDV,1);
    digitalWrite(PinLEDB,0);
    }
```

```
  if (c == LedB) {
    Serial.println (c);
    digitalWrite(PinLEDR,0);
    digitalWrite(PinLEDV,0);
    digitalWrite(PinLEDB,1);
  }
  if (c == Off) {
    Serial.println (c);
    digitalWrite(PinLEDR,0);
    digitalWrite(PinLEDV,0);
    digitalWrite(PinLEDB,0);
  }
  Serial.flush();
  c=' ';
}
```

2.2 Programme pour la réception d'une chaine de caractères

Les chaines de caractères sont représentées sous la forme de tableau de variables de type "**char**" se terminant par un caractère nul (code ASCII 0, à distinguer du '0' qui a le code ASCII 48) et permettant ainsi à la fonction "**Serial.print()**" de savoir où se termine la chaîne de caractères

Exemples :

```
char Str1[15];                    // Déclare un tableau de 15 caractères sans l'initialiser
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};   //Déclare un tableau de caractères (avec un caractère supplémentaire non utilisé) et le compilateur ajoutera le caractère nul requis
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};   //Idem, mais en déclarant explicitement le caractère nul
char Str4[ ] = "arduino";         // Déclare un tableau de caractères Initialisé avec une chaîne de caractère constante entre guillemets doubles : le compilateur dimensionnera le tableau pour stocker la chaîne de caractère constante et le caractère nul terminal
char Str5[8] = "arduino";         //Déclare un tableau de caractères Initialisé avec une taille explicite et une chaîne de caractère constante (avec un caractère supplémentaire non utilisé) et le compilateur ajoutera le caractère nul requis
```

La réception d'une chaine de caractères envoyée depuis le moniteur série, peut être considéré comme un envoi successif de caractères uniques que l'on peut stocker dans un tableau de caractères afin de reconstituer la chaine.

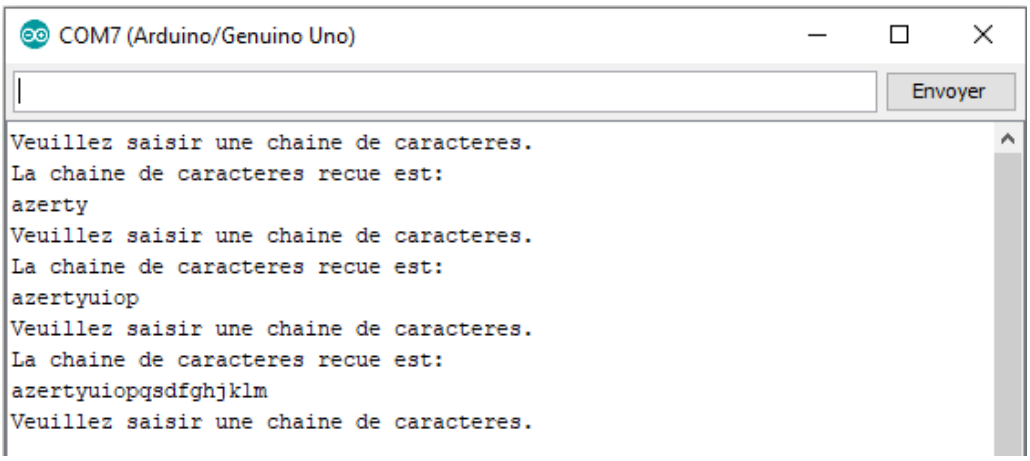
Le programme (nommé "Chaines.ino" dans le dossier "Codes/INO") ci-dessous effectue cette reconstitution :

Chaines

```
void setup() {
Serial.begin(9600);
}

void loop() {
  int Val=0;
  char tampon[20]="";
  Serial.println("Veuillez saisir une chaine de caracteres.");
  while(Val==0)
    {
    delay(200);
    Val=Serial.available();
    }
  for (int i=0; i < Val; i++)
    {
    tampon[i]=Serial.read();
    delay(15);
    }
  Serial.println("La chaine de caracteres recue est:");
  Serial.println(tampon);
  Serial.flush();
}
```

Dans ce programme, il est demandé de saisir une chaine de caractères. Tant que le nombre de caractères disponibles sur le port série est nul (**Val==0**), la mémoire tampon du port série est vérifiée (**Val=Serial.available()**), toutes les 200 ms, jusqu'à ce que tous les caractères de la chaine soient comptabilisés. Ceux-ci sont alors lu un par un à travers une boucle "**for**" (**tampon[i]=Serial.read()**) et sont stockés dans un tableau de 20 caractères au maximum appelé "**tampon**" qui est ensuite affiché dans le moniteur série. Puis la mémoire du port série est vidée et il est de nouveau demander de saisir une chaine de caractères.

Voici le résultat dans le moniteur série :



Une autre méthode pour recevoir une chaine de caractères depuis le moniteur série est d'utiliser la fonction "**readString()**" qui lit les caractères contenus dans la mémoire tampon du port série et retourne une chaine de caractère. Au-delà, d'un certain temps (par défaut, 1 s), la fonction se termine. Le temps d'attente pour recevoir les données peut être réglé avec la fonction "**setTimeout()**" en ms.

Par exemple, dans ce nouveau programme (nommé "ReadString.ino" dans le dossier "Codes/INO"), toujours avec le même circuit, on va maintenant demander à l'utilisateur d'envoyer un message. Si le message est "**ON**", la DEL rouge s'allume, et si c'est "**OFF** ", la DEL rouge s'éteint.

ReadString

```
const int PinLEDR = 11;
String msg="";

void setup() {
  Serial.begin(9600);
  pinMode (PinLEDR, OUTPUT);
  Serial.println ("Pour allumer la DEL rouge, envoyer: ON");
  Serial.println ("Pour eteindre la DEL rouge, envoyer: OFF");
  Serial.setTimeout(100);
}

void loop() {
  if (Serial.available() > 0) {
    msg = Serial.readString();
    }
  if (msg == "ON") {
    Serial.println (msg);
    digitalWrite(PinLEDR,1);
    }
  if (msg == "OFF") {
    Serial.println (msg);
    digitalWrite(PinLEDR,0);
    }
  Serial.flush();
  msg="";
}
```

<u>Remarques</u> :

- Contrairement aux caractères uniques (variable de type "**char**") qui sont écrit entre des guillemets simples (**'A'**), les chaines de caractères sont écrites entre des doubles guillemets (**"ABC"**).

- L'instruction " **String msg="";** " déclare une variable "**chaine de caractères**" initialement vide.

- les objets **String** sont des instances de la classe **String()**, qui est intégrée au logiciel Arduino depuis la version 0019 et qui permet d'utiliser et de manipuler des chaînes de caractères.

Tous les exemples suivants sont des déclarations valides pour une nouvelle instance String :

```
String stringOne = "Hello String";                          // en utilisant une chaîne de caractères
String stringOne =  String('a');                  // conversion d'un caractère simple en objet String String
stringTwo =  String("This is a string");       // conversion d'une chaîne de caractère en objet String String
stringOne =  String(stringTwo + " with more");          // concaténation d'un objet String et d'une chaîne
String stringOne =  String(13);                          // conversion d'un nombre en base 10 par défaut
String stringOne =  String(analogRead(0), DEC);                 // conversion d'une valeur int en base 10
String stringOne =  String(45, HEX);                  // conversion de la valeur 45 en base hexadecimale
String stringOne =  String(255, BIN);                       // conversion de la valeur 255 en base binaire
String stringOne =  String(millis(), DEC);                     // conversion d'une valeur long en base 10
String monString=char(65);                  // ajoute le caractère 'A' au String à partir du code ASCII 65
```

2.3 Programme pour la conversion d'une chaine de caractères en nombre

Dans les programmes pour Arduino, Il est souvent nécessaire de convertir des chaines de caractères en une variable représentant un nombre entier ou à virgule pour effectuer des calculs.

2.3.1 La fonction "atoi()"

Il existe, pour cela, trois fonctions "**atoi()**", "**atol()**" et "**atof()**", permettant respectivement de transformer une chaine de caractères en nombre entier court, en nombre entier long et en nombre à virgule.

Attention cependant, car par défaut ces fonctions retournent une valeur nulle en cas d'erreur (comme par exemple, si on essaye de convertir des lettres en nombre) et donc se pose le problème de savoir si la valeur retournée est bien un zéro saisi ou un zéro d'erreur…

En application, le programme (nommé "Atoi.ino" dans le dossier "Codes/INO") suivant, qui utilise la fonction "**atoi()**" permet de régler la luminosité de la DEL rouge du circuit précédant,

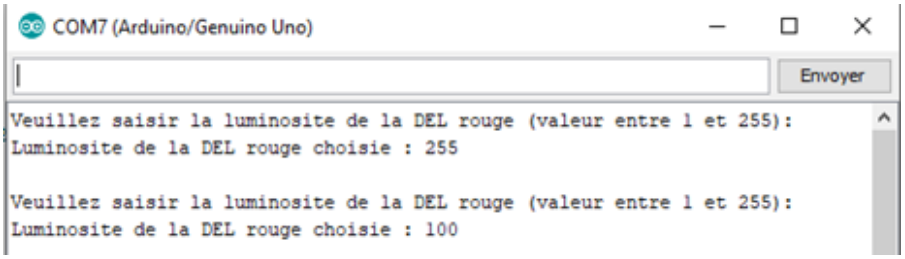
en retour d'une chaine de caractères envoyé depuis le moniteur série dont la conversion en nombre entier doit être compris entre 1 et 255.

Atoi

```
const int PinLEDR = 11;
int Luminosite = 0;
int Luminosite2 = 0;
int N=0;

void setup() {
  Serial.begin(9600);
  pinMode (PinLEDR, OUTPUT);
  digitalWrite(PinLEDR,0);
}

void loop() {
  while(Luminosite<1 || Luminosite>255)
  {
  int Val=0;
  char tampon[10]="";
  Serial.println("Veuillez saisir la luminosite de la DEL rouge (valeur entre 1 et 255):");
  while(!Val)
    {
    delay(200);
    Val=Serial.available();
    }
  for (int i=0; i < Val; i++)
    {
    tampon[i]=Serial.read();
    delay(15);
    }
  Luminosite = atoi(tampon);
  }
  Serial.print("Luminosite de la DEL rouge choisie : "); Serial.println(Luminosite);
  Serial.println("");
  analogWrite(PinLEDR,Luminosite);
  Luminosite=0;
}
```

On retrouve dans ce programme, la partie de code permettant de recevoir une chaine de caractères depuis le port série. La chaine est convertie en nombre entier avec la fonction "**atoi()**". Si le résultat n'est pas compris entre 1 et 255 ("**while(Luminosite<1 || Luminosite>255)** ") , l'utilisateur doit saisir de nouveau, une valeur.

Voici le résultat dans le moniteur série :



2.3.2 La fonction "sscanf()"

Une autre méthode est d'utiliser la fonction "**sscanf()**" qui permet de couper et de convertir une chaine de caractères en une série de variables allant du caractère unique (type "**char**") au nombres décimaux (type "**int**", "**float**").

Exemple de conversion une chaine de caractères en variables (programme "sscanf.ino" dans le dossier "Codes/INO") :

sscanf

```
int jour;
int an;
char str[20] = "08 Fevrier 2020";
char mois[10]="";


void setup() {
 Serial.begin(9600);
 sscanf(str, "%d %s %d", &jour, mois, &an );
 Serial.println(jour);
 Serial.println(mois);
 Serial.println(an);
}

void loop() {
}
```

Cet exemple permet d'extraire le jour, le mois et l'année d'une chaine de caractère représentant une date :

- Le premier argument de la fonction est **"str"**. Il représente la chaine de caractère à convertir,
- l'argument **"%d"** indique à la fonction qu'il faut extraire un nombre décimal,
- l'argument **"%s"** pour extraire une chaine de caractères (la fin de la chaine est repérée par un espace, un saut de ligne),

- les autres arguments sont les variables dans lesquelles il faut stocker les valeurs converties (les variables commençant par un **"&"** sont destinées à recevoir un nombre).

La fonction **"sscanf()"** retourne le nombre de variables qu'elle a pu convertir, si cette valeur est différente du nombre d'arguments de conversion c'est qu'il y a eu une erreur.
Ainsi, dans notre programme de réglage de la luminosité de la DEL rouge, on peut remplacer la ligne :

```
Luminosite = atoi(tampon);
```

Par :

```
int N=0;
char c;
N = sscanf(tampon, "%d %c", &Luminosite, c);
if (N!=1){
    Luminosite=0; }
```

Avec ce code (programme "sscanf2.ino" dans le dossier "Codes/INO") :

- l'argument **"%c**" indique à la fonction qu'il faut extraire un caractère,

- si la chaine de caractères **"tampon"** représente uniquement un nombre entier décimal, celle-ci est convertie et le nombre est stocké dans la variable **"**Luminosité**"** (dans ce cas, N qui représente le nombre de variables extraites par la fonction est égal à 1)**,**

- si la chaine de caractères ne contient pas de nombre entier décimal, mais uniquement des lettres, la variable **"Luminosité"** reste égal à 0 (même si dans ce cas aussi N=1 car la fonction a extrait le premier caractère dans la variable **"c"**) et la condition pour sortir de la boucle n'est pas remplie,

- et enfin si la chaine de caractère est constituée d'un nombre et de caractères, on a alors **N = 2**, la variable **"Luminosité"** est réinitialisée à 0 pour forcer l'utilisateur à saisir seulement un nombre décimal entier.

2.3.3 La fonction "Serial.parseInt()"

Dans les méthodes précédentes, les chaines de caractères envoyés depuis le moniteur série sont dans un premier temps reçue telles quelles par l'Arduino puis sont converties en nombre si celles-ci le permettent.

Il est également possible de convertir directement les chaines de caractères en entier long avant la réception par l'Arduino avec la fonction **"parseInt()"** de la classe **"Serial"**

Cette fonction retourne le premier entier long du tampon de la liaison série. Les caractères lettres ou le signe **"-"** sont ignorés. Au-delà, d'un certain temps (par défaut, 1 s), la fonction se termine et retourne **"0"**. Le temps d'attente pour recevoir les données peut être réglé avec la fonction "**setTimeout()**" en ms.

Le code de notre programme de réglage de la luminosité de la DEL rouge devient alors (programme "Parseint.ino" dans le dossier "Codes/INO"):

Parseint

```
const int PinLEDR = 11;
int Luminosite = 0;

void setup() {
  Serial.begin(9600);
  pinMode (PinLEDR, OUTPUT);
  digitalWrite(PinLEDR,0);
  Serial.println("Veuillez saisir la luminosite de la DEL rouge (valeur entre 1 et 255):");
  Serial.println("");
  Serial.setTimeout(100);
}

void loop() {
  while(Luminosite<1 || Luminosite>255)
  {
    Luminosite=Serial.parseInt();
  }
  Serial.print("Luminosite de la DEL rouge choisie : "); Serial.println(Luminosite);
  Serial.println("");
  analogWrite(PinLEDR,Luminosite);
  Luminosite=0;
}
```

Avec cette méthode, une chaine de caractères envoyés depuis le moniteur série contenant un nombre et du texte est considérée comme valide. Les caractères représentant du texte sont juste ignorés.

Si la chaine ne contient que du texte, la fonction **"Serial.parseInt()"** retourne **"0" (Luminosite = 0)** et la condition pour sortir de la boucle n'est pas remplie.