TRANSMISSION SERIE A L'AIDE D'ARDUINO

Nous allons nous intéresser à utiliser la transmission série avec Arduino.

Nous allons voir comment envoyer puis recevoir des informations avec l'ordinateur.

Enfin nous ferons quelques exercices

RAPPELS SUR LA LIAISON SERIE RS232

- Transmissions série et parallèle
  - La communication entre 2 systèmes peut se faire de manière :
  - Parallèle
  - Série
  - La communication série est très importante dans le domaine de la télécommunication et plus généralement dans le transfert d'informations

RAPPELS SUR LA LIAISON SERIE RS232

- Transmissions série et parallèle
  - Intérêts d'une liaison série
    - Moins de câblage (- de cuivre donc - cher)
    - Pas de perturbation entre pistes
      - Débits plus élevés
      - Distances de communication plus importantes

RAPPELS SUR LA LIAISON SERIE RS232

Transmissions séries asynchrones

- Les communications asynchrones sont définies par plusieurs paramètres :
  - Les niveaux de tensions
  - La vitesse de transmission (Baud Rate en anglais)
  - Le format des données
  - Le mode de fonctionnement
  - Full-Duplex ou Half-Duplex
- Les supports physiques de communication peuvent être divers :
  - Fils de cuivre,
  - fibre optique,
  - hertzien, ...

RAPPELS SUR LA LIAISON SERIE RS232

Principe de transmission

Dans une communication série RS232, les bits sont envoyés les uns à la suite des autres sur la ligne en commençant par le bit de poids faible. La transmission s'appuie donc sur le principe des registres à décalage. La transmission se fait octet par octet :

- pas d'horloge transmise
- Nécessité de rajouter un bit de "START" ('0' logique) avant l'octet à transmettre, et un bit de "STOP" ('1' logique) après l'octet à transmettre.
- La norme RS232 prévoit également la possibilité de rajouter un autre bit juste avant le bit de STOP :
  - Bit de parité
  - ou un 2ème bit de STOP

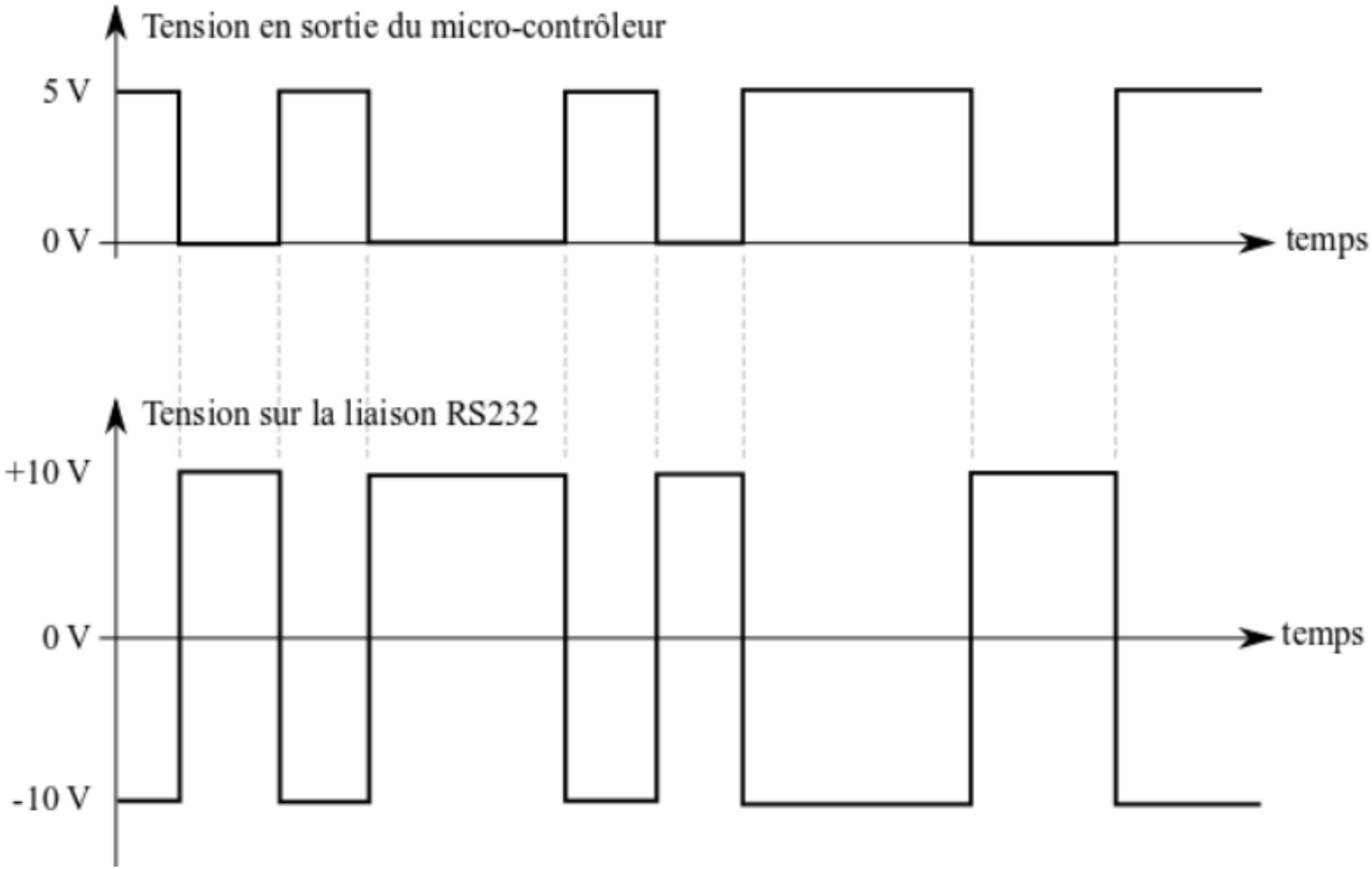
RAPPELS SUR LA LIAISON SERIE RS232

Vitesse de transmission des données

- Les deux équipements doivent être configurés avec la même vitesse (baud rate).
- Elle est exprimée en bauds (ou bits/seconde)
- Ces vitesses sont normalisées :
  - 1200 bauds
  - 2400 bauds
  - 4800 bauds
  - 9600 bauds
  - 19200 bauds
  - 38400 bauds
  - 57600 bauds
  - 115200 bauds

RAPPELS SUR LA LIAISON SERIE RS232

Norme RS232

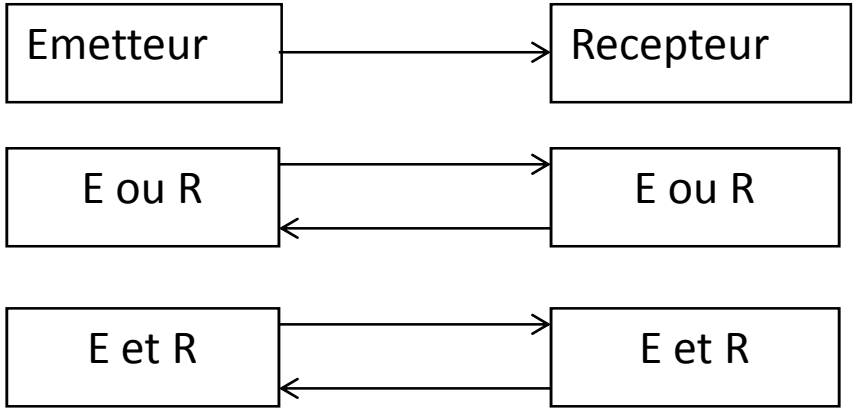


RAPPELS SUR LA LIAISON SERIE RS232

Mode de transmission

La transmission des données peut se faire de manière :

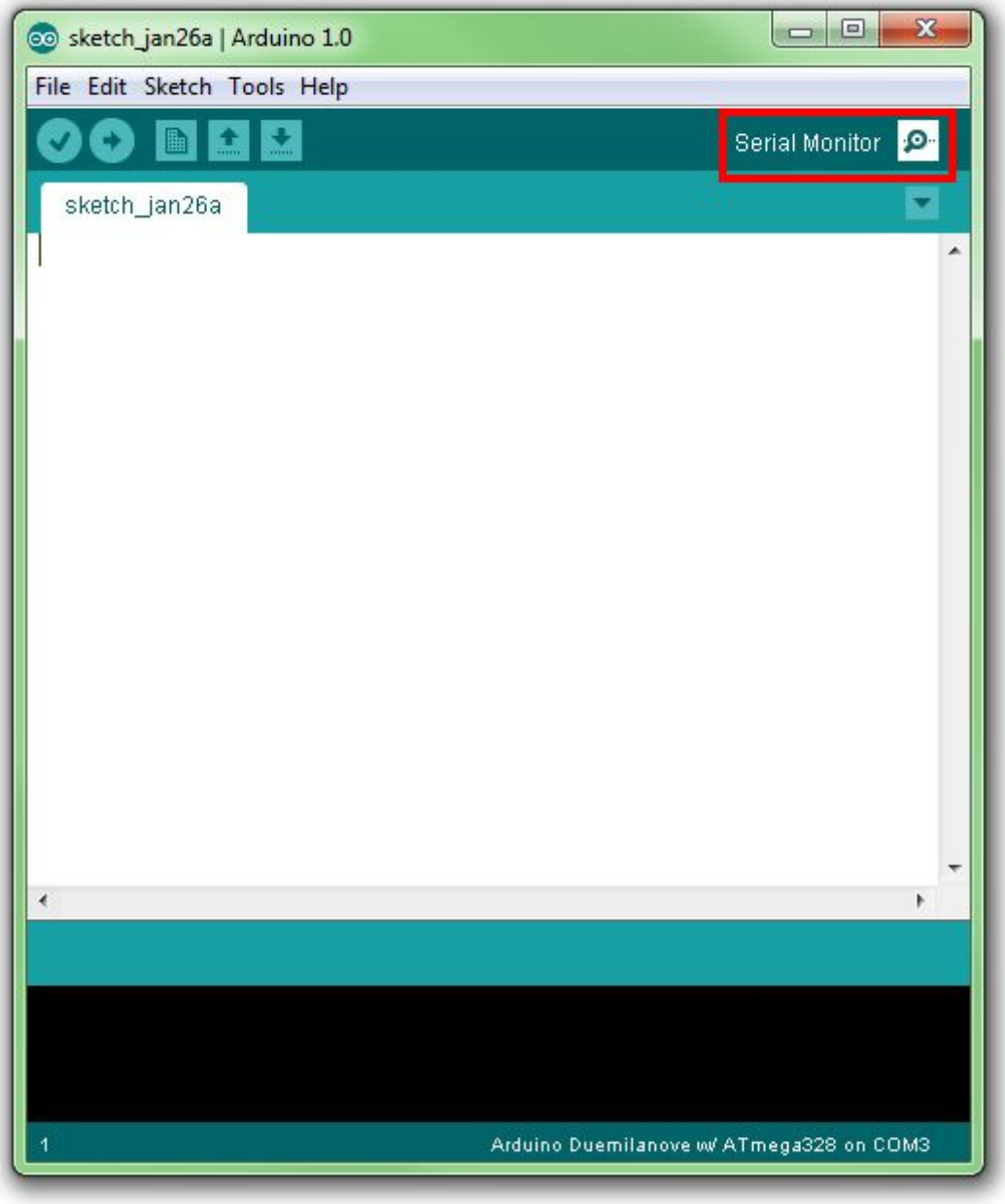
- unidirectionnelle (**`simplex`**)
- alternée (**`half-duplex`**)
- simultanée (**`full-duplex`**)



TRANSMISSION SERIE A L'AIDE D'ARDUINO

Du côté de l'ordinateur

Pour pouvoir utiliser la communication de l'ordinateur, il suffit de cliquer sur le bouton dans la barre de menu pour démarrer l'outil.



TRANSMISSION SERIE A L'AIDE D'ARDUINO

Une nouvelle fenêtre s'ouvre : c'est le **terminal série** :

Dans cette fenêtre, vous allez pouvoir envoyer des messages sur la voie série de votre ordinateur (qui est émulée par l'Arduino) ; recevoir les messages que votre Arduino vous envoie ; et régler deux trois paramètres tels que la vitesse de communication avec l'Arduino et l'autoscroll qui fait défiler le texte automatiquement.

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Du côté du programme

L'objet *Serial*

Pour utiliser la voie série et communiquer avec notre ordinateur (par exemple), nous allons utiliser un *objet* (une sorte de variable mais plus évoluée) qui est intégré nativement dans l'ensemble Arduino : l'objet **Serial**.

Cet objet rassemble des informations (vitesse, bits de données, etc.) et des fonctions (envoi, lecture de réception,…) sur ce qu'est une voie série pour Arduino.

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Le setup

Pour commencer, nous allons donc initialiser l'objet Serial. Ce code sera à copier à chaque fois que vous allez créer un programme qui utilise la voie série. Le logiciel Arduino à prévu, dans sa *bibliothèque Serial*, tout un tas de fonctions qui vont nous êtres très utiles, voir même indispensables afin de bien utiliser la voie série. Ces fonctions, sont par exemple:

Functions

if (Serial)	available()	availableForWrite()	begin()	end()	find()
findUntil()	flush()	parseFloat()	parseInt()	peek()	print()
println()	read()	readBytes()	readBytesUntil()		readString()
readStringUntil()		setTimeout()	write()		serialEvent()

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Dans le but de créer une communication entre votre ordinateur et votre carte Arduino, il faut déclarer cette nouvelle communication et définir la vitesse à laquelle ces deux dispositifs vont communiquer. Et oui, si la vitesse est différente, l'Arduino ne comprendra pas ce que veut lui transmettre l'ordinateur et vice versa ! Ce réglage va donc se faire dans la fonction setup, en utilisant la fonction begin() de l'objet Serial.

Lors d'une communication informatique, une vitesse s'exprime en bits par seconde ou **bauds**. Ainsi, pour une vitesse de 9600 bauds on enverra jusqu'à 9600 '0' ou '1' en une seule seconde. Les vitesses les plus courantes sont 9600, 19200 et 115200 bits par seconde.

```
void setup()
{
   //on démarre la liaison en la réglant à une vitesse de 9600 bits par seconde.
   Serial.begin(9600);
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Envoyer des données

Généralement, des informations provenant de capteurs connectés à la carte ou de son programme (par exemple la valeur d'une variable) seront transmises via la connexion série vers le PC.

La carte Arduino traite les informations provenant de ces capteurs, s'il faut elle adapte ces informations, puis elle les transmet.

println()
La fonction que l'on va utiliser pour débuter, s'agit de println(). Deux fonctions sont quasiment identiques, mais à quoi servent-elles ?

print() : cette fonction permet d'envoyer des données sur la voie série. On peut par exemple envoyer un caractère, une chaine de caractère ou d'autres données.

println() : c'est la même fonction que la précédente, elle permet simplement un retour à la ligne à la fin du message envoyé.

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Pour utiliser ces fonctions, rien de plus simple :

Serial.print("Salut les étudiants licence S6 Télécom ‘’);

Bien sûr, au préalable, vous devrez avoir « déclaré/créé » votre objet Serial et définis une valeur de vitesse de communication :

```
void setup()
{
   //création de l'objet Serial (=établissement d'une nouvelle communication série)
   Serial.begin(9600);
   //envoie de la chaine " Salut les étudiants licence S6 Télécom " sur la voie série
   Serial.print(" Salut les étudiants licence S6 Télécom");
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

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

   //l'objet exécute une première fonction
   Serial.print(" Salut les étudiants licence S6 Télécom ");
   //puis une deuxième fonction, différente cette fois-ci
   Serial.println("Vive le Groupe A!");
   //et exécute à nouveau la même
   Serial.println("Non Vive Le Groupe B");
}
```

Sur le terminal série, on verra ceci :

```
Salut les étudiants licence S6 Télécom Vive le Groupe A !
Non Vive Le Groupe B
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Avec la fonction print(), il est aussi possible d'envoyer des chiffres ou des nombres car ce sont des caractères :

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

   Serial.println(9);          //chiffre
   Serial.println(42);          //nombre
   Serial.println(32768);       //nombre
   Serial.print(3.1415926535);   //nombre à virgule
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

La fonction print() affiche par défaut seulement deux chiffres après la virgule. Il suffit de rajouter le nombre de décimales que l'on veut afficher :

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

   Serial.println(3.1415926535, 0);
   Serial.println(3.1415926535, 2); //valeur par défaut
   Serial.println(3.1415926535, 4);
   Serial.println(3.1415926535, 10);
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Envoyer la valeur d'une variable

Au lieu de mettre un caractère ou un nombre, il suffit de passer la variable en paramètre pour qu'elle soit ensuite affichée à l'écran :

```
int variable = 512;
char lettre = 'a';

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

   Serial.println(variable);
   Serial.print(lettre);
}
```

Résultat

```
512
a
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Envoyer d'autres données

Prenons l'exemple d'un nombre choisi judicieusement : 65.
Pourquoi ce nombre en particulier ? Et pourquoi pas 12 ou 900 ?

La table ASCII, de l'américain « **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange », soit en bon français : « Code américain normalisé pour l'échange d'information » est, selon Wikipédia : « la norme de codage de caractères en informatique la plus connue, la plus ancienne et la plus largement compatible »

maVariable = 'A'; //l'ordinateur stocke la valeur 65 dans sa mémoire (cf. table ASCII)

maVariable = maVariable + 1; //la valeur stockée passe à 66 (= 65 + 1)

//à l'écran, on verra s'afficher la lettre "B"

TRANSMISSION SERIE A L'AIDE D'ARDUINO

TABLE ASCII

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

TRANSMISSION SERIE A L'AIDE D'ARDUINO

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

   Serial.println(65, BIN); //envoie la valeur 1000001
   Serial.println(65, DEC); //envoie la valeur 65
   Serial.println(65, OCT); //envoie la valeur 101 (ce n'est pas du binaire !)
   Serial.println(65, HEX); //envoie la valeur 41
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

```
void loop()
{
   char i = 0;
   char lettre = 'a'; // ou 'A' pour envoyer en majuscule

   Serial.println("------  L'alphabet des héros  ------"); //petit message d'accueil

   //on commence les envois
   for(i=0; i<26; i++)
   {
      Serial.print(lettre); //on envoie la lettre
      lettre = lettre + 1; //on passe à la lettre suivante
      delay(250); //on attend 250ms avant de réenvoyer
   }
   Serial.println(""); //on fait un retour à la ligne

   delay(5000); //on attend 5 secondes avant de renvoyer l'alphabet
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Recevoir des données
Cette fois, il s'agit de l'Arduino qui reçoit les données que nous, utilisateur, allons transmettre à travers le terminal série.
Pareillement à cette conversion, l'objet Serial dispose d'une fonction pour « écouter » la voie série afin de savoir si oui ou non il y a une communication de données.

Réception de données
On m'a parlé ?
Pour vérifier si on a reçu des données, on va régulièrement interroger la carte pour lui demander si des données sont disponibles dans son **buffer de réception**.

Un buffer est une zone mémoire permettant de stocker des données sur un cours instant. Dans notre situation, cette mémoire est dédiée à la réception sur la voie série.

Il en existe un aussi pour l'envoi de donnée, qui met à la queue leu leu les données à envoyer et les envoie dès que possible.
Pour cela, on utilise la fonction available() (de l'anglais « disponible ») de l'objet Serial.

Cette fonction renvoie le nombre de caractères dans le buffer de réception de la voie série. Voici un exemple de traitement :

TRANSMISSION SERIE A L'AIDE D'ARDUINO

```
void loop()
{
   int donneesALire = Serial.available(); //lecture du nombre de caractères disponibles dans le
buffer
   if(donneesALire > 0) //si le buffer n'est pas vide
   {
      //Il y a des données, on les lit et on fait du traitement
   }
   //on a fini de traiter la réception ou il n'y a rien à lire
}
```

Cette fonction de l'objet Serial, available(), renvoie la valeur -1 quand il n'y a rien à lire sur le buffer de réception.

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Lire les données reçues
Une fois que l'on sait qu'il y a des données, il faut aller les lire pour éventuellement en faire quelque chose.

La lecture se fera tout simplement avec la fonction… read() !

Cette fonction renverra le premier caractère arrivé non traité .

On accède donc caractère par caractère aux données reçues.

Ce type de fonctionnement est appelé FIFO (First In First Out, premier arrivé, premier traité).

Si jamais rien n'est à lire, la fonction renverra -1 pour le signaler

TRANSMISSION SERIE A L'AIDE D'ARDUINO

```
void loop()
{
   //on lit le premier caractère non traité du buffer
   char choseLue = Serial.read();

   if(choseLue == -1) //si le buffer est vide
   {
      //Rien à lire, rien lu
   }
   else //le buffer n'est pas vide
   {
      //On a lu un caractère
   }
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Le serialEvent

Si vous voulez éviter de mettre le test de présence de données sur la voie série dans votre code, Arduino a rajouter une fonction qui s'exécute de manière régulière.

Cette dernière se lance régulièrement avant chaque redémarrage de la loop.

Ainsi, si vous n'avez pas besoin de traiter les données de la voie série à un moment précis, il vous suffit de rajouter cette fonction.

Pour l'implémenter c'est très simple, il suffit de mettre du code dans une fonction nommé « serialEvent() » (attention à la casse) qui sera a rajouté en dehors du setup et du loop.

Le reste du traitement de texte se fait normalement, avec Serial.read par exemple. Voici un exemple de squelette possible :

TRANSMISSION SERIE A L'AIDE D'ARDUINO

```
const int maLed = 11; //on met une LED sur la broche 11

void setup()
{
   pinMode(maLed, OUTPUT); //la LED est une sortie
   digitalWrite(maLed, HIGH); //on éteint la LED
   Serial.begin(9600); //on démarre la voie série
}
void loop()
{
   delay(500); //fait une petite pause
   //on ne fait rien dans la loop
   digitalWrite(maLed, HIGH); //on éteint la LED

}
void serialEvent() //déclaration de la fonction d'interruption sur la voie série
{
   //lit toutes les données (vide le buffer de réception)
   while(Serial.read() != -1);

   //puis on allume la LED
   digitalWrite(maLed, LOW);
}
```

TRANSMISSION SERIE A L'AIDE D'ARDUINO

Exemple de code complet

Voici maintenant un exemple de code complet qui va aller lire les caractères présents dans le buffer de réception s'il y en a et les renvoyer tels quels à l'expéditeur (mécanisme d'écho).

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

void loop()
{
   char carlu = 0; //variable contenant le caractère à lire
   int cardispo = 0; //variable contenant le nombre de caractère disponibles dans le buffer

   cardispo = Serial.available();

   while(cardispo > 0) //tant qu'il y a des caractères à lire
   {
      carlu = Serial.read(); //on lit le caractère
      Serial.print(carlu); //puis on le renvoi à l'expéditeur tel quel
      cardispo = Serial.available(); //on relit le nombre de caractères dispo
   }
   //fin du programme
}
```

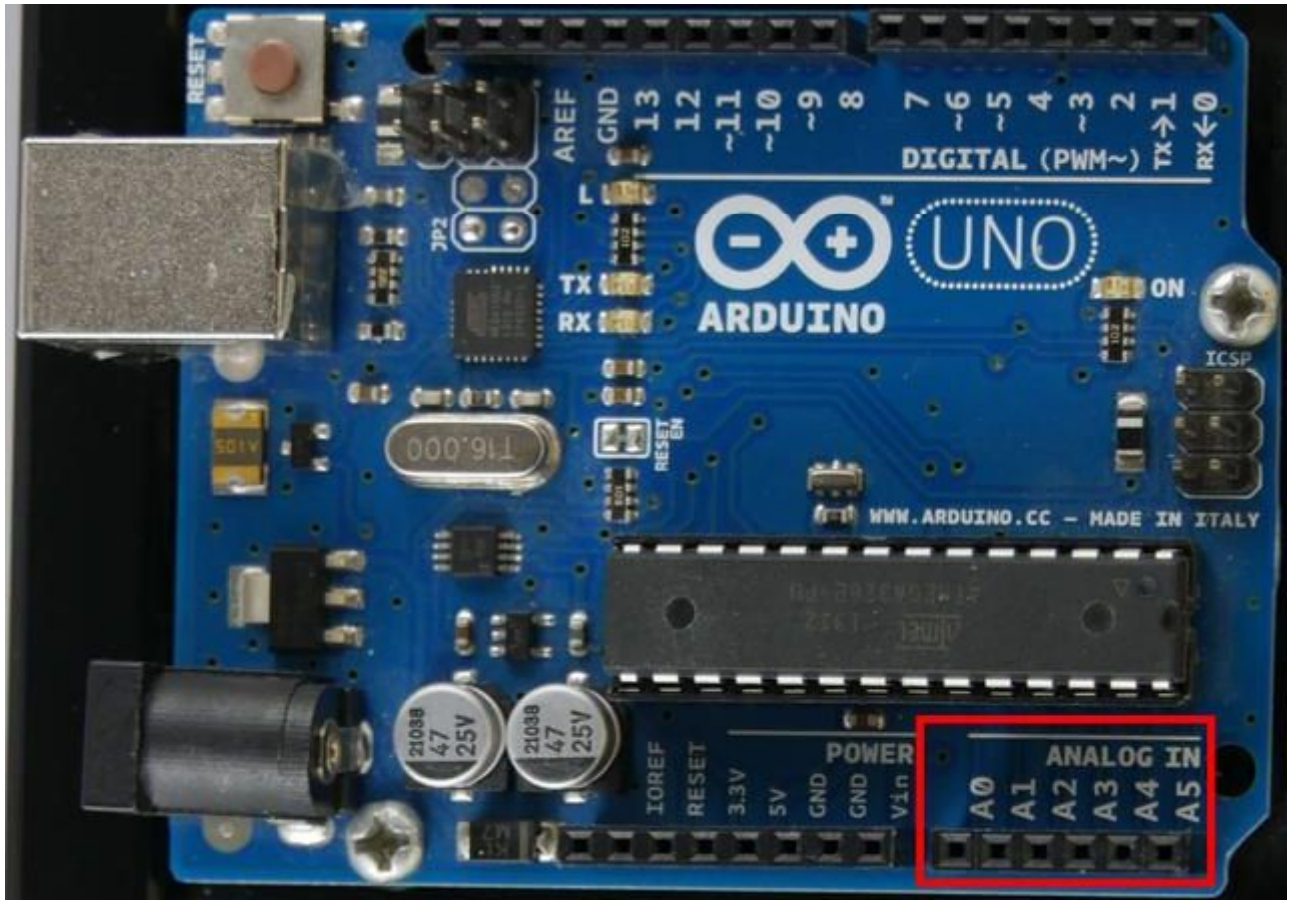
TRANSMISSION SERIE A L'AIDE D'ARDUINO

Un autre exemple à comprendre

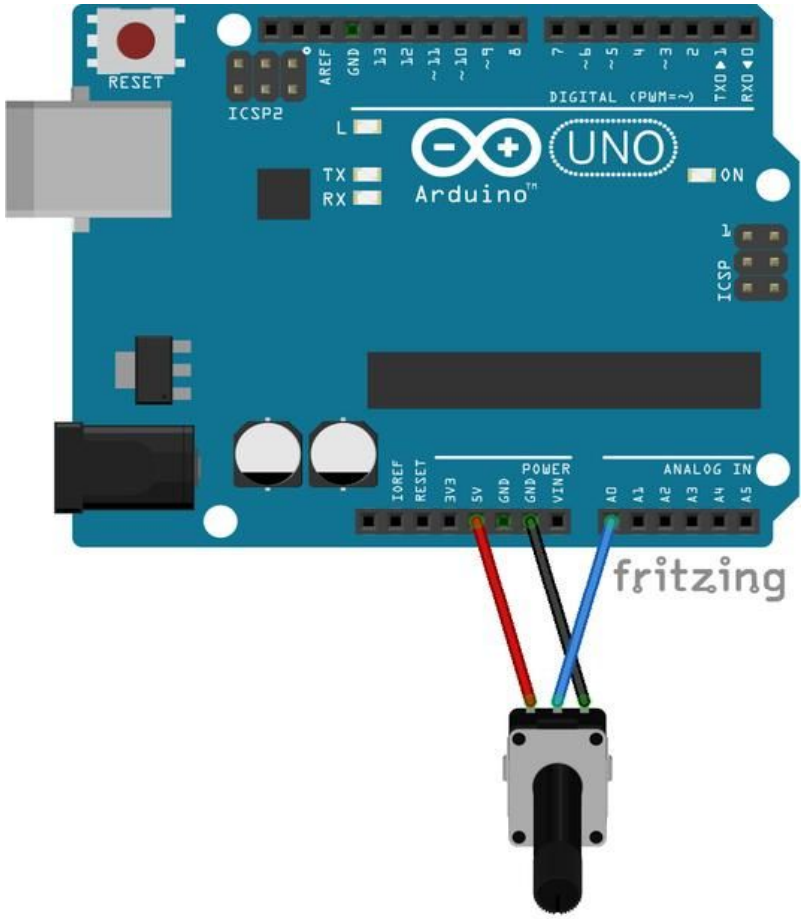
```
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop()
{
  if (Serial.available() > 0) {
    char c = Serial.read();
    if (c == '1') {
      digitalWrite(13, HIGH);
    } else {
      digitalWrite(13, LOW);
    }
  }
  int val = analogRead(A0);
  Serial.println(val);
  delay(100);
}
```

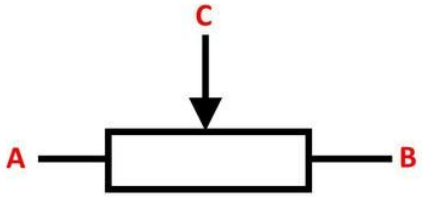
CONVERSION ANALOGIQUE NUMERIQUE AVEC ARDUINO

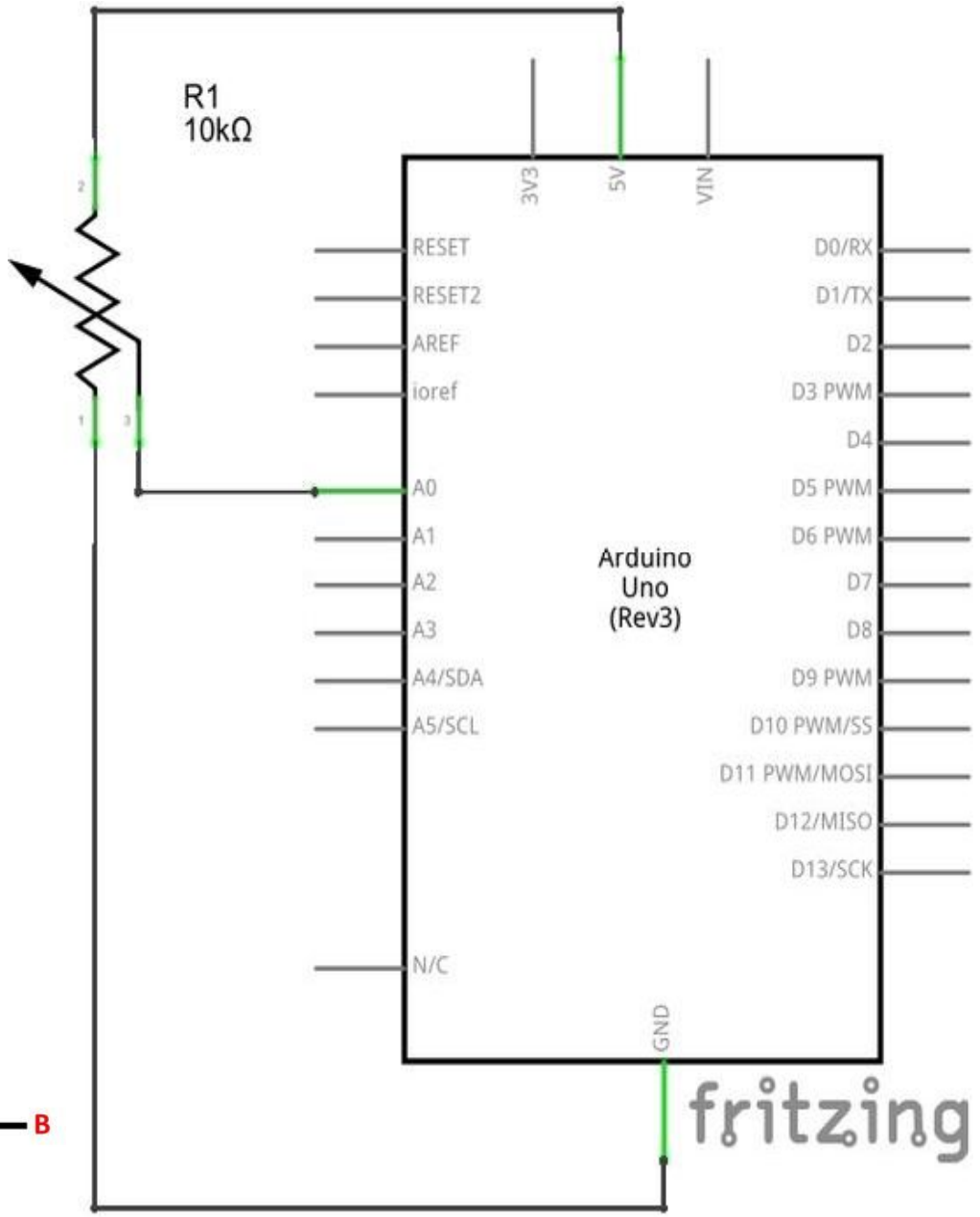
Dans le cas d'une carte Arduino UNO, il y a 6 entrées analogiques, pouvant mesurer des tensions comprises entre 0 et 5 volts, avec une précision de 10 bits (soit 1024 points).
Si on fait rapidement le calcul, 1024 points sur une plage de 5 volts donne une précision absolue de 0,0048828125 volt, soit environ 4,9mV.



CONVERSION ANALOGIQUE NUMERIQUE AVEC ARDUINO





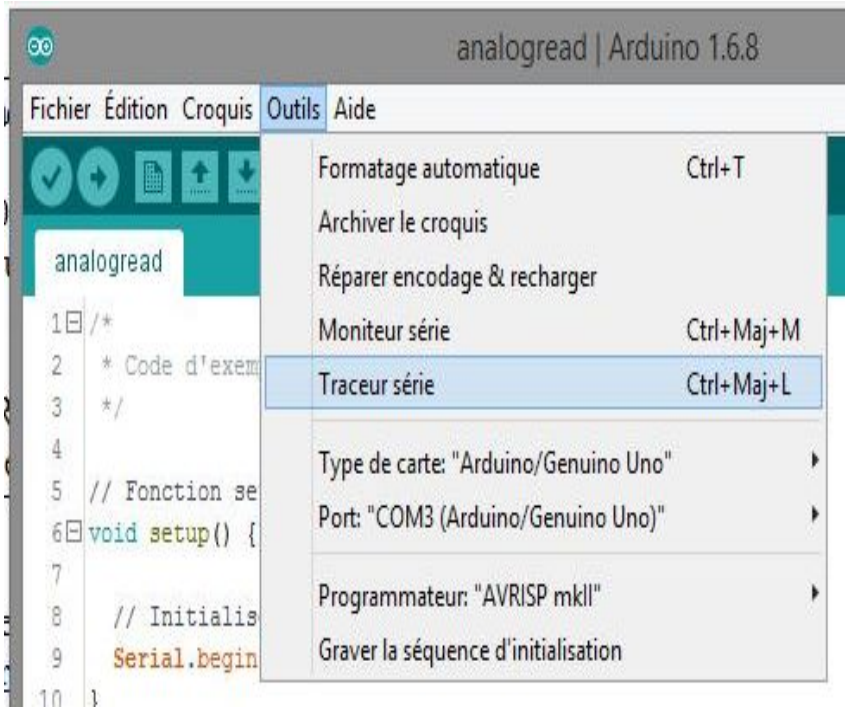


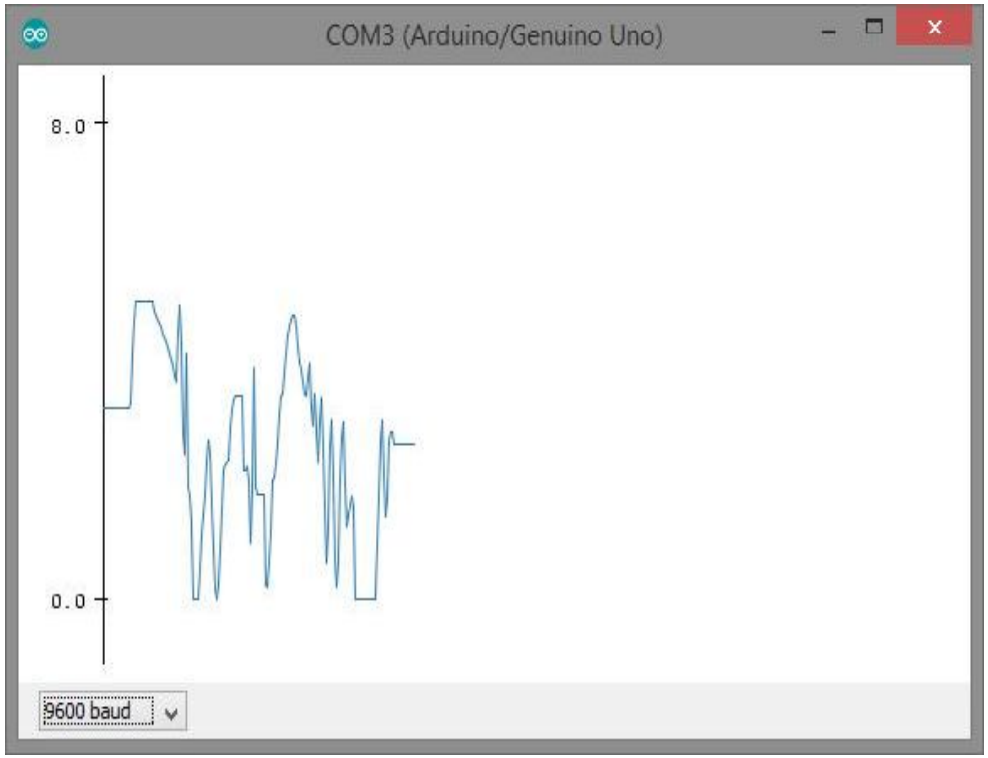
CONVERSION ANALOGIQUE NUMERIQUE AVEC ARDUINO

Faire une courbe en temps réel avec l'environnement Arduino

Si votre code envoi régulièrement sur le port série un nombre (entier ou à virgule) sur une ligne (avec Serial.println()), vous pouvez demander au logiciel Arduino de vous tracer un graphique en temps réel de ces valeurs.

```
void setup() {
Serial.begin(9600);
}
void loop() {
int valeur = analogRead(A0);
float tension = valeur * (5.0 / 1023.0);
Serial.println(tension);
delay(250);
}
```





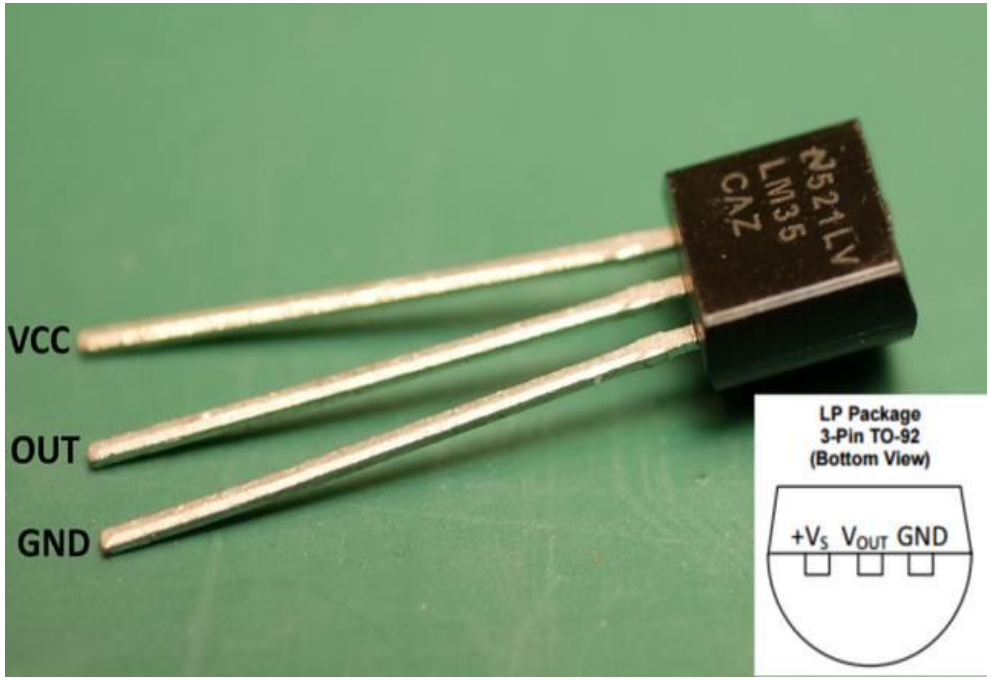
CONVERSION ANALOGIQUE NUMERIQUE AVEC ARDUINO

Mesurer une température avec un capteur LM35 et une carte Arduino

Qu'est-ce qu'un capteur LM35 ?

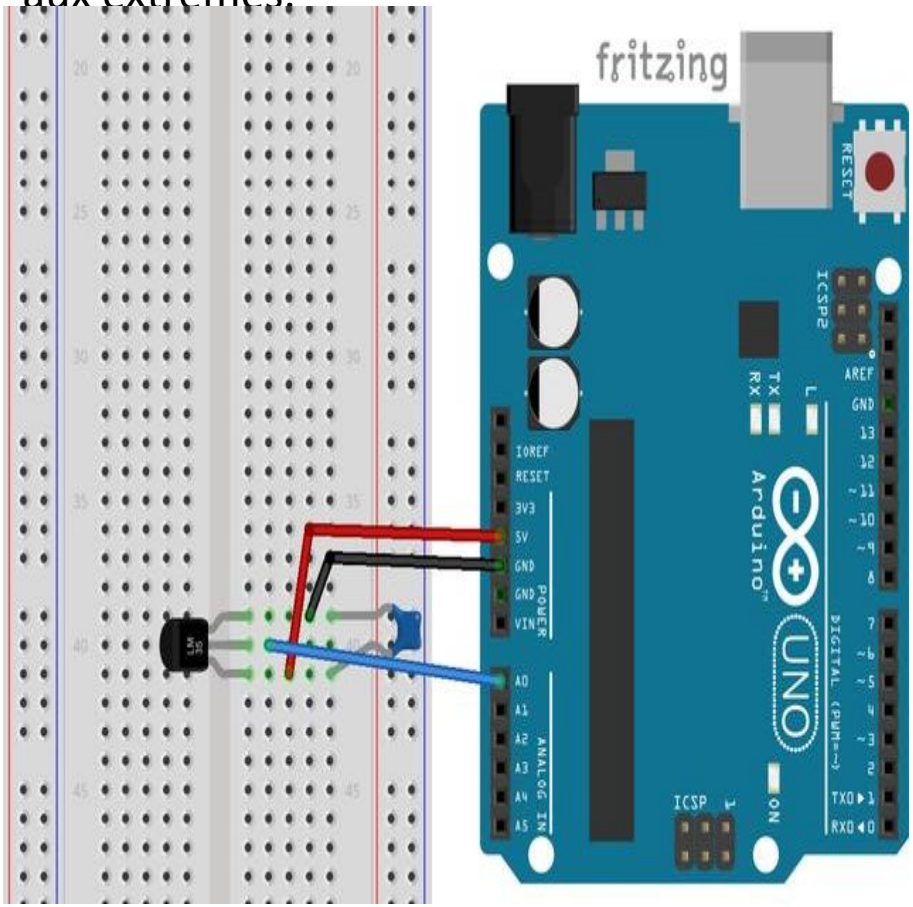
Le capteur de température LM35 est un capteur analogique de température fabriqué par Texas Instruments. Il est extrêmement populaire en électronique, car précis, peu couteux, très simple d'utilisation et d'une fiabilité à toute épreuve.
Le capteur de température LM35 est capable de mesurer des températures allant de -55°C à +150°C dans sa version la plus précise et avec le montage adéquat, de quoi mesurer n'importe quelle température.

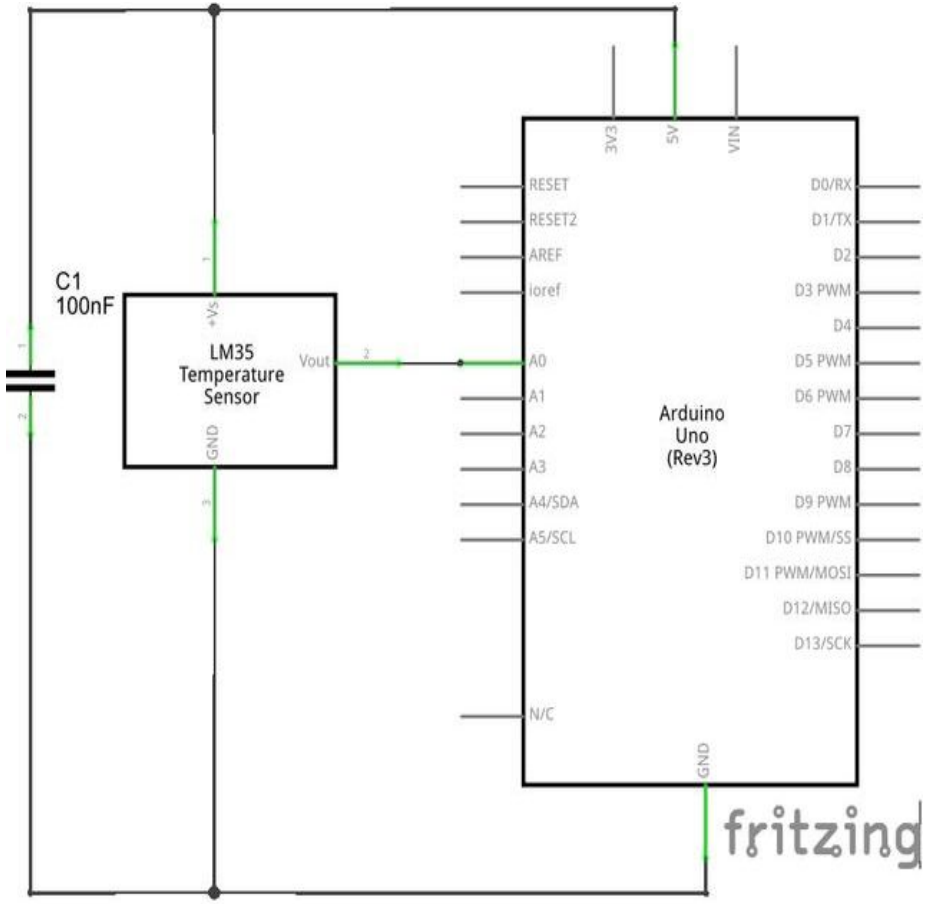
N.B. Les versions grand publiques sont capables de mesurer des températures comprises entre -40°C et +110°C.
La sortie analogique du capteur est proportionnelle à la température. Il suffit de mesurer la tension en sortie du capteur pour en déduire la température. Chaque degré Celsius correspond à une tension de +10mV.



CONVERSION ANALOGIQUE NUMERIQUE AVEC ARDUINO

il existe trois versions :
Le LM35DZ, capable de mesurer des températures de 0 à 100°C avec une précision de 1.5°C aux extrêmes.
Le LM35CZ, capable de mesurer des températures de -40 à 110°C avec une précision de 1.5°C aux extrêmes.
Le LM35CAZ, capable de mesurer des températures de -40 à 110°C avec une précision de 1°C aux extrêmes.





CONVERSION ANALOGIQUE NUMERIQUE AVEC ARDUINO

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

void loop() {
int valeur_brute = analogRead(A0);
float temperature_celcius = valeur_brute * (5.0 / 1023.0 * 100.0);
Serial.println(temperature_celcius);
delay(250);
}
```