

CM DIE

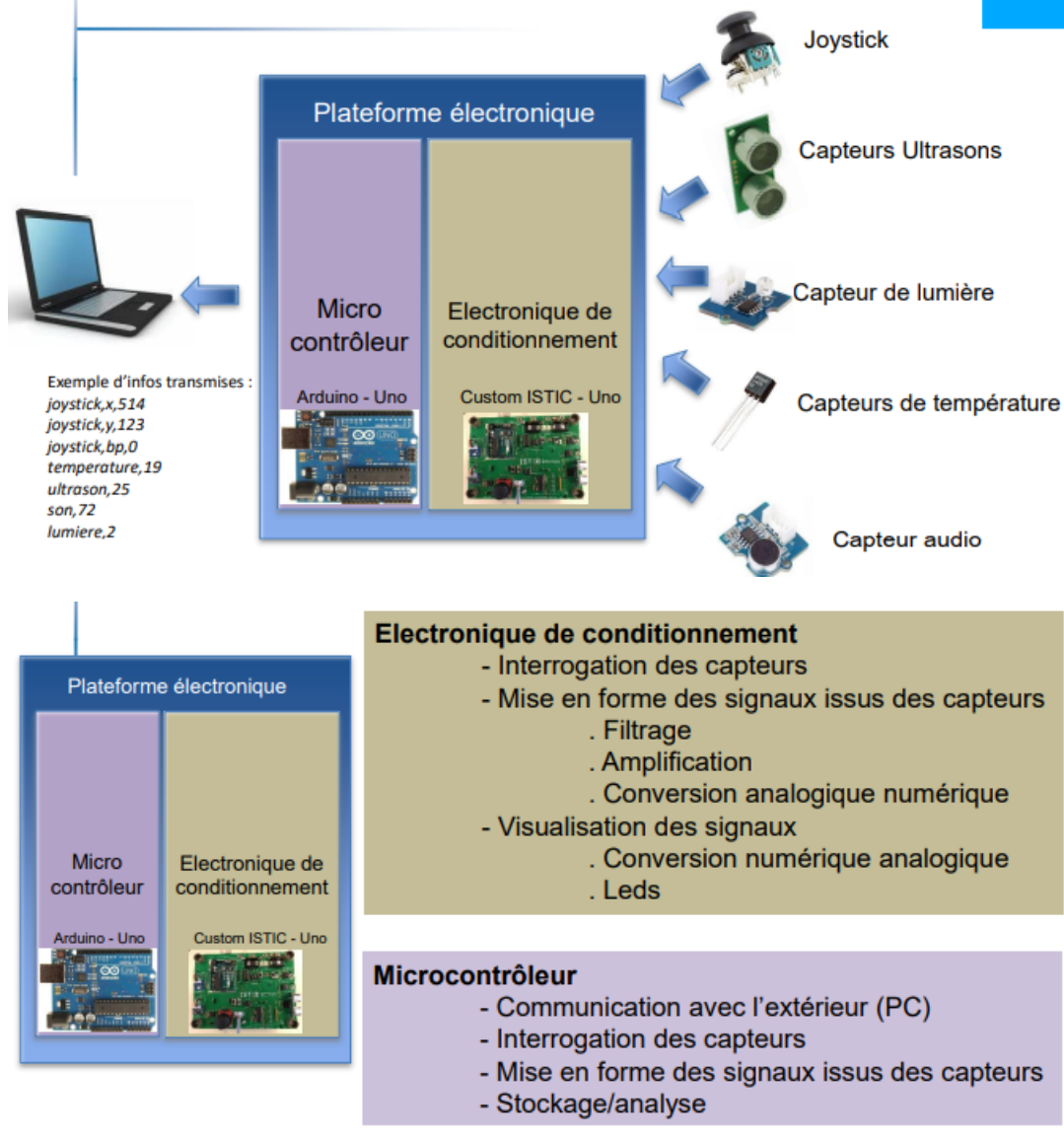
- Présentation UE
- Contexte
- Un système électronique
 - Objectifs de la conversion analogique-numérique
 - Objectifs de la conversion analogique-numérique
 - Cerveau du système
- La carte capteurs
- Le microcontrôleur
- Fonctions électroniques
 - Conversion courant/tension
 - Conversion tension/lumière
 - Filtrage
 - Applications
 - Types de filtres
 - Amplification
 - Applications
 - Comparaison
 - Applications
 - Bilan fonctions analogiques
 - Conversion analogique numérique
 - Conversion numérique analogique
- L'environnement logiciel Arduino
- Les capteurs
 - Joystick
 - Capteur ultrason
 - Capteur de température
 - Capteur de lumière
 - Capteur de son
- Fusion des données

Présentation UE

Examen après les 2 séances de TP d'électronique

1. Les objectifs du module

9



Contexte

i Informatique

- Automatiser l'exécution de tâches et applications
- Mise en œuvre de logiciels (Software)
- Information traitée par l'exécution de programmes (codes) sur des machines

i Electronique

- Traiter et gérer des signaux électriques
- Mise en œuvre de composants matériels (Hardware)
- Signaux (tension/courant) traités par des circuits constitués de composants

2 types de composants :

- **passifs** : pas de source d'énergie
- **actifs** : besoin d'une source d'énergie

Mais à quoi sert l'électronique ?

Faciliter l'interaction avec l'environnement

- . Collecter des informations
- . Recevoir des informations
- . Traiter des informations
- . Mémoriser des informations
- . Interpréter des informations
- . Exploiter des informations
- . Agir en fonction des informations
- . Transmettre des informations

Informations ?

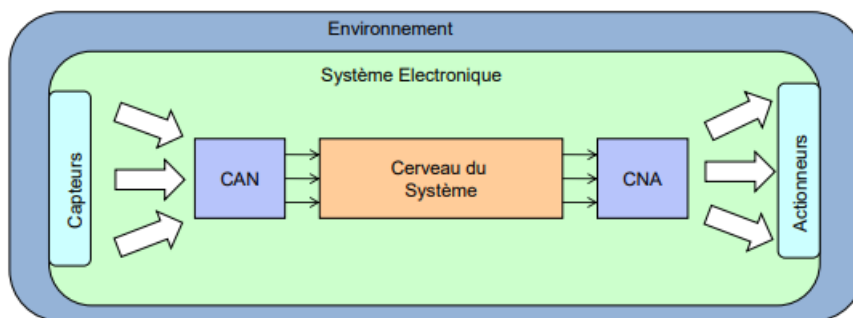
Signaux électriques

Tension et courant

=> Utilisation de la **tension** car + facile à observer que le courant

Un système électronique

Définition : un système électronique permet d'interagir avec son environnement



CAN = capteur Analogique Numérique CNA = capteur Numérique Analogique

Capteurs

Données environnementales



Signaux électriques



Température
Mouvement : déplacement,
vitesse, accélération
Gaz
Lumière
Son
Humidité
...



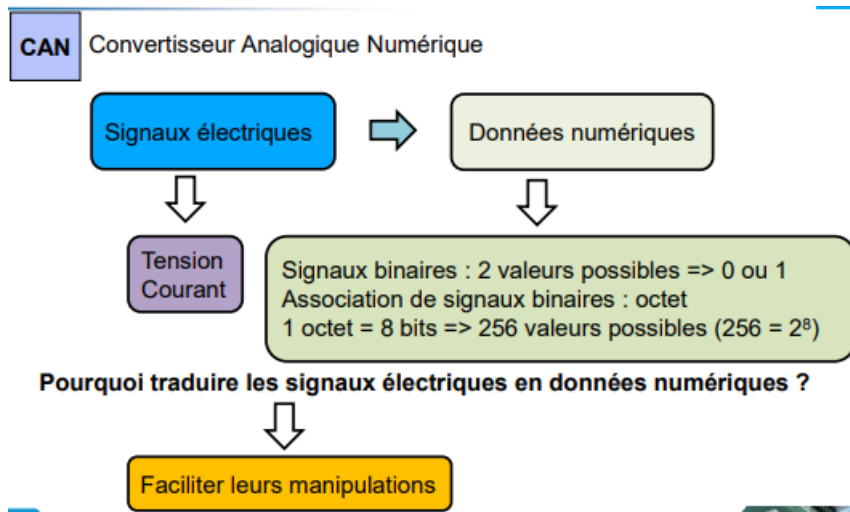
Tension
Courant



Analogique



Infinité de valeurs
(espace et temps)



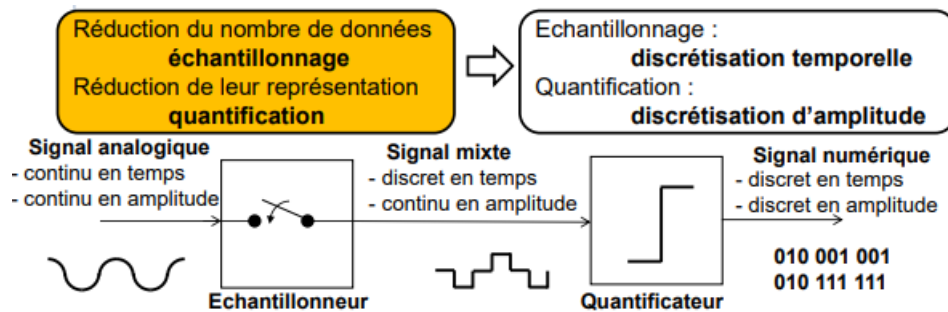
Objectifs de la conversion analogique-numérique

- réduire le nombre de données = échantillonnage (en évitant de perdre de l'information, sur le temps)

Exemple

Pour le son, on ne peut échantillonner toutes les 10 secondes => **perte d'information**

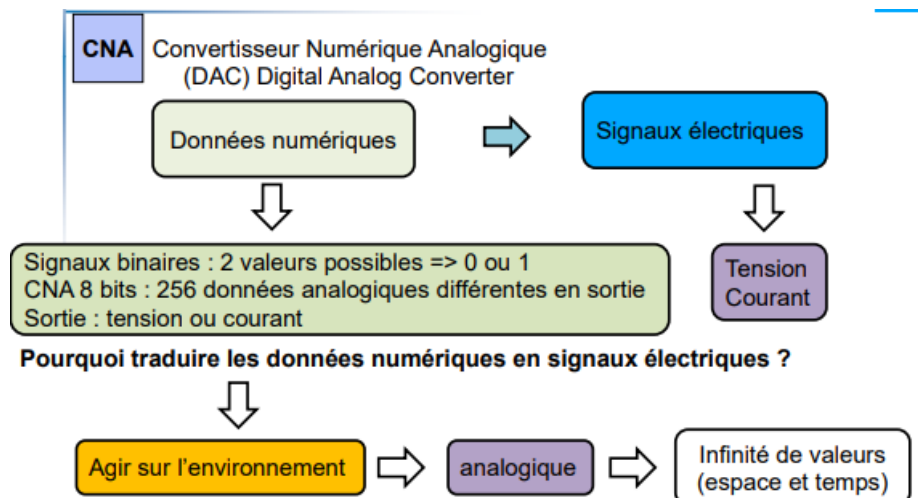
- réduire leur représentation = quantification (sur l'amplitude)



à chaque niveau ↪ une valeur numérique donc approximation donc le passage de l'analogique au numérique **perd de l'information**

Objectifs de la conversion analogique-numérique

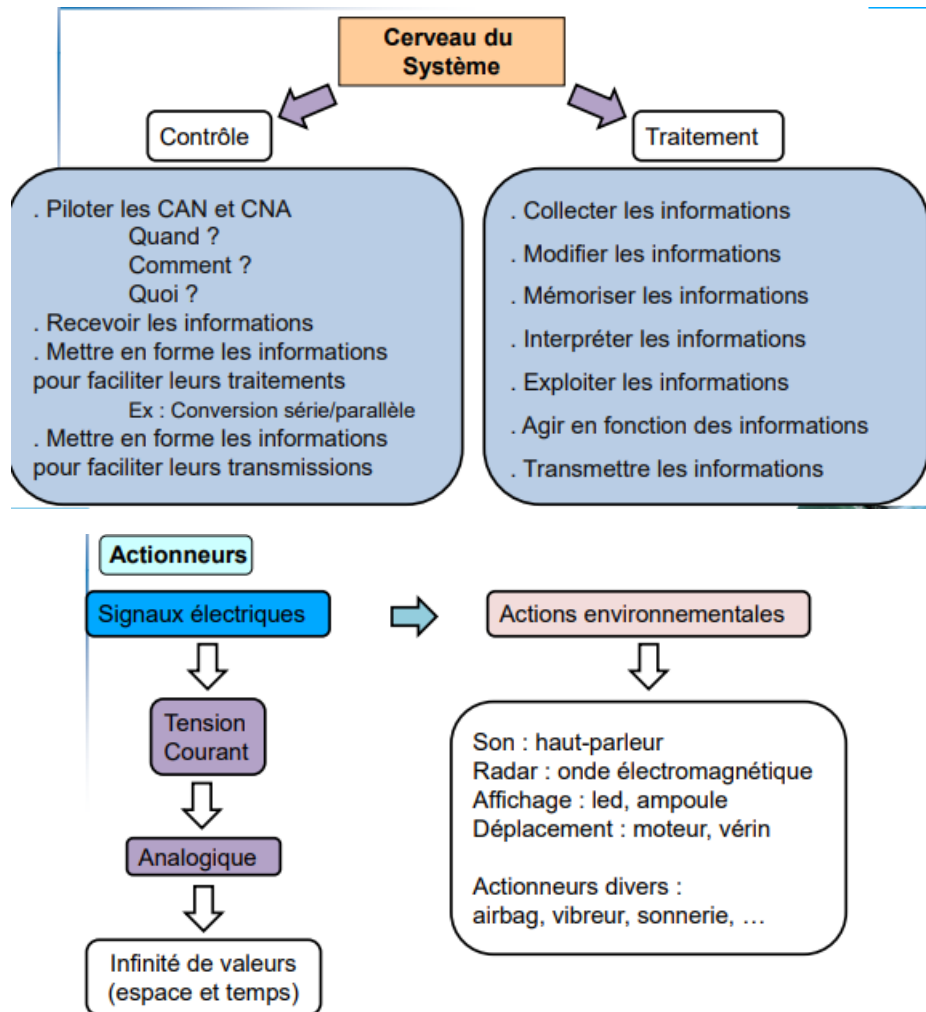
A chaque valeur = une tension analogique correspondante



Cerveau du système

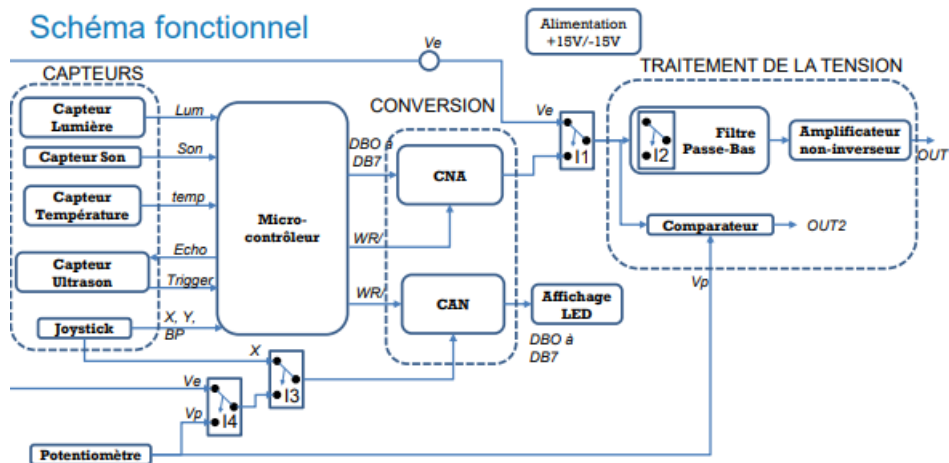
constitué de 2 parties :

- contrôle
- traitement



La carte capteurs

Schéma fonctionnel

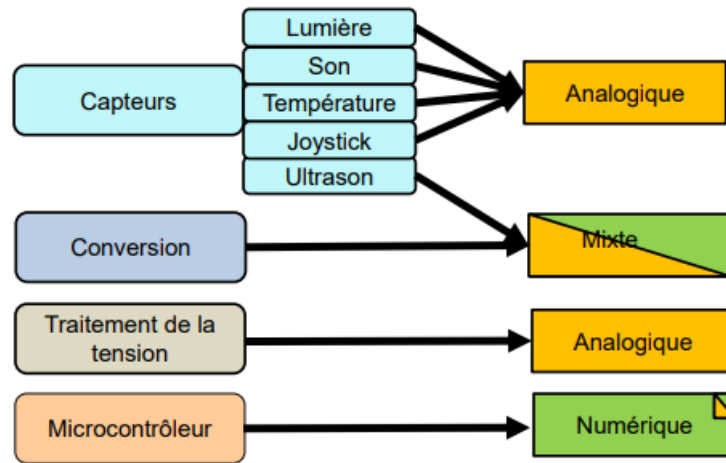


Source : sophie.ailain@univ-rennes1.fr



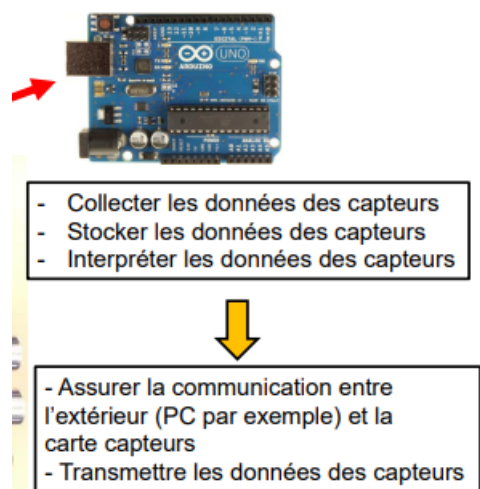
Tip

→ Faire le lien entre le schéma fonctionnel et la carte



Une petite partie du microcontrôleur est **mixte** (convertisseurs analogiques)

Le microcontrôleur

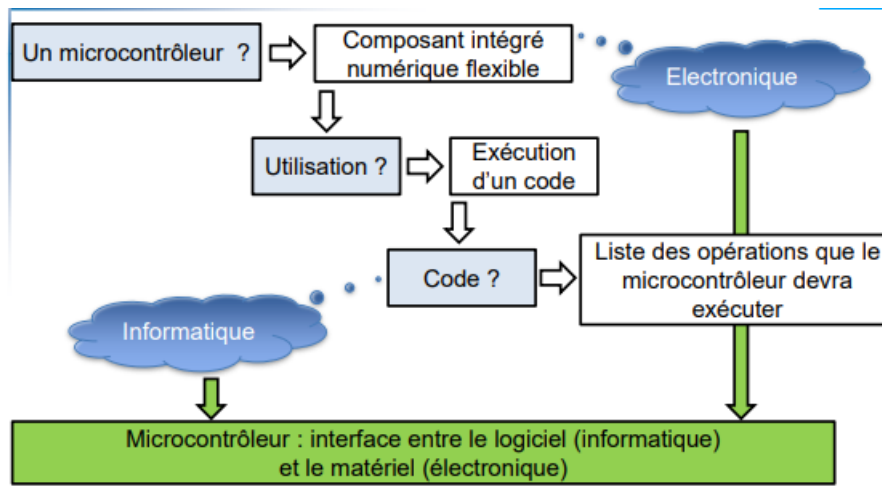


Microcontrôleur

- Système quasi-autonome
- Unité centrale avec un processeur
- Mémoire vive + morte
- Interfaces pour communiquer

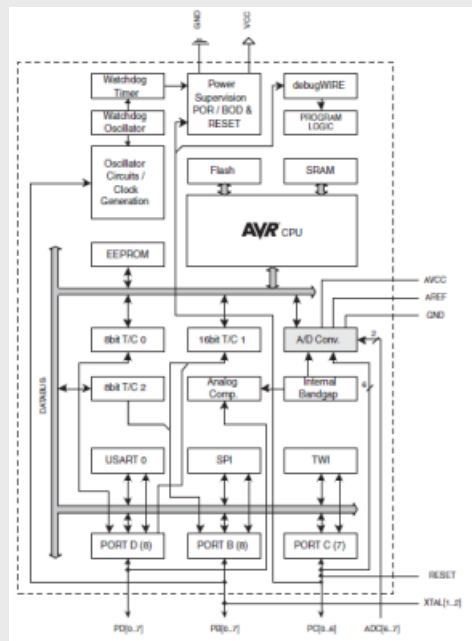
→ peut se contrôler

- **Ressources très limitées** donc \neq d'un "ordinateur" => permet de faire un pré-traitement et de gérer uniquement les données **utiles**



i en TP : microcontrôleur ATmega328

Le microcontrôleur n'est pas Arduino



- Architecture RISC (Reduced instruction set computer)
- Low Power
- In System programmable system
- 20 MIPS (Mega Instructions Per Second)
- 3 types de mémoire (Flash, SRAM, EEPROM)
- 6 A/D Converters de 10 bits
- 6 PWM (Pulse Width Modulation)

Les mémoires internes

Flash memory

- non volatile : **pas de perte d'information** quand on coupe l'alimentation (*exemple : clé USB*)
- rapide
- stockage du Bootloader (programme initial -> permet d'utiliser le microcontrôleur)
- stockage du programme application / code applicatif (programme compilé/traduit en un langage machine et sauvegardé dans la mémoire flash)

SRAM : Static Random Acces Memory

- mémoire vive : mémoire accessible en écriture et en lecture
- volatile
- stockage intermédiaire des données (*exemple : boucle for*)

EEPROM : Electrically Erasable Programmable *Read Only* Memory

- non volatile
- stockage de paramètres utiles à l'application (*exemple : données constructeurs...*)

Fonctions électroniques

7 fonctions étudiées :

1. Conversion courant tension
2. Conversion tension lumière
3. Filtrage
4. Amplification
5. Comparaison
6. Conversion analogique numérique
7. Conversion numérique analogique

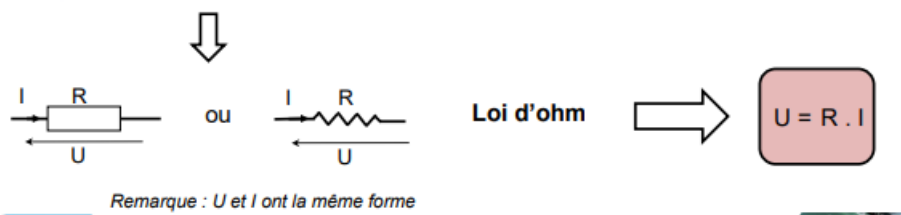
Conversion courant/tension

Conversion courant/tension et tensions/courant : Fonction de base de l'électronique

=> Courant et tension : 2 grandeurs à manipuler qui transportent l'information utile
=> Informatique : monde numérique

Objectif du convertisseur : Convertir un courant en tension et vice versa

Convertisseur : Impédance et dans notre cas une résistance



Le courant *traverse* la résistance, alors que la tension est *aux bornes* de la résistance => plus facile d'observer une tension qu'un courant

Courant

- Quantité d'électricité qui parcourt un conducteur
- Débit de charges électriques dans ce conducteur.
- Appelé aussi intensité (électrique)
- Noté I et exprimé en Ampère (A)

Analogie avec une rivière

- le lit de la rivière = le conducteur électrique
- le débit d'eau = l'intensité électrique

Tension

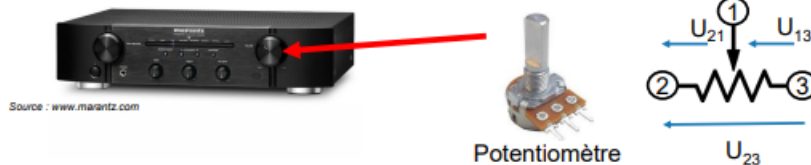
- Différence de 2 potentiels électriques
- Liée à une énergie potentiellement disponible (voir l'analogie avec la rivière)
- $U_{AB} = V_A - V_B$ et exprimée en Volt (V).

Analogie avec une rivière

- l'altitude = le potentiel électrique
- la différence d'altitude entre le début et la fin de la rivière (le dénivelé) = la différence de potentiels (la tension)

Applications

- Gérer le volume audio sur un ampli



$$U_{23} = U_{21} + U_{13} \quad \Rightarrow \quad U_{23} = \text{constante}$$

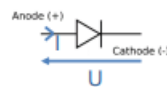
Déplacement du point 1 vers la gauche : U_{21} diminue et U_{13} augmente
 Déplacement du point 1 vers la droite : U_{21} augmente et U_{13} diminue

Conversion tension/lumière

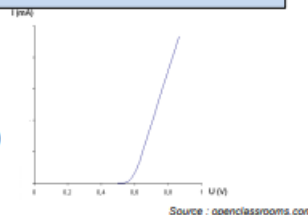
LED ? => Light-Emitting Diode (ou DEL : Diode Electroluminescente)



Diode ? => Composant électronique : dipôle asymétrique

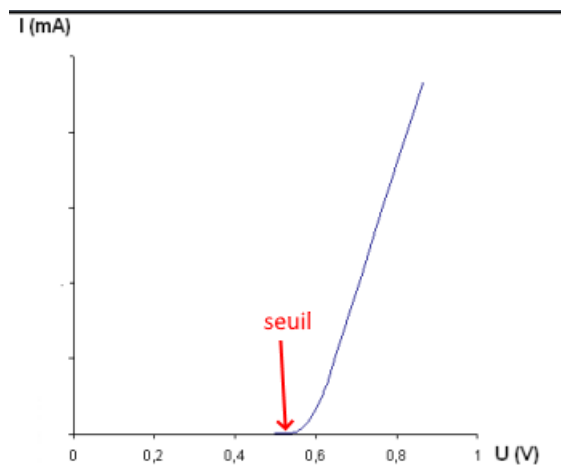


Si $U >$ Tension de seuil, la diode conduit ($I > 0$ A)
 Sinon la diode est dite bloquée ($I = 0$ A)...



Une LED = une diode => Si elle est conductrice => elle éclaire !!!

Le courant est bloqué jusqu'à un certain seuil :



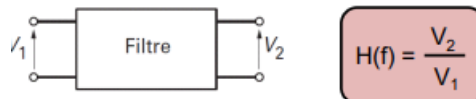
Filtrage

i Filtrage : Fonction très utilisée dans les systèmes électroniques

- traitement du signal
- modifications des caractéristiques spectrales d'un signal électrique
- **Objectif du filtrage** : extraire une information utile : « *Le but du filtrage est d'extraire une partie de l'information, afin de restituer un signal plus intelligible* »

i Filtre

Ensemble de composants électroniques permettant de modifier la forme d'un signal électrique (spectre de fréquence et/ou phase) ; peut être analogique et numérique



💧 Fonction de transfert $H(f)$

Rapport entre la tension de sortie et la tension d'entrée

Rappels :

signal sinusoïdal => amplitude, fréquence, phase

$$S(t) = A \sin(\omega t + \varphi)$$

avec

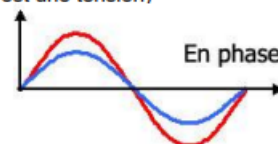
A : Amplitude du signal (en volts si S est une tension)

ω : pulsation en $\text{rad}^{-1}/\text{s} = 2\pi f = 2\pi/T$

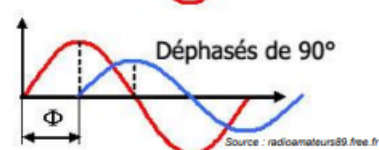
F : fréquence en Hz

T : période en s

φ : phase à l'origine en rad



Déphasage entre 2 signaux : retard



Source : radioamateurs89.free.fr



Caractérisation d'un filtre

deux courbes en fonction de la fréquence permettent d'expliciter les modifications de forme entre V_2 et V_1

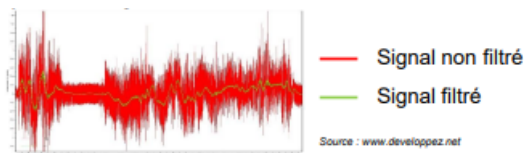
- amplitude/fréquence => gain en fonction de la fréquence
- phase/fréquence => déphasage en fonction de la fréquence

Fonction de transfert $H(f)$:
Gain = $|V_2/V_1|$
Phase = $\text{Arg}(V_2/V_1)$

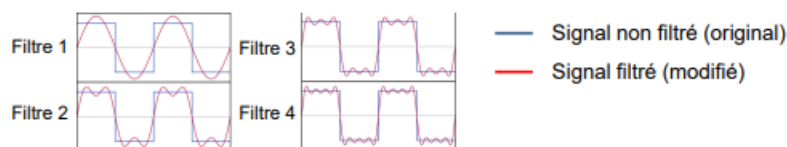
$$H(f) = \frac{V_2}{V_1}$$

Applications

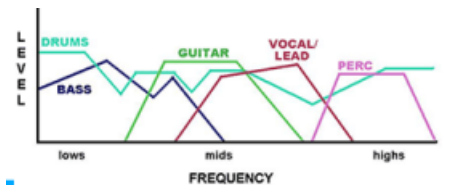
Nettoyer un signal bruité



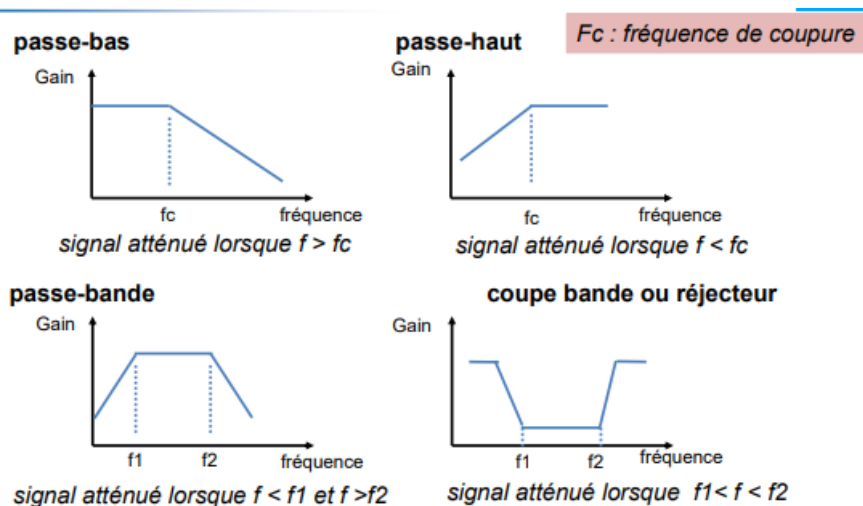
Modifier la forme d'un signal



Mixer de la musique



Types de filtres



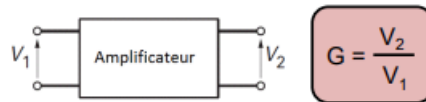
filtre de base : filtre passe-haut

Amplification

Objectifs : Augmenter l'amplitude d'un signal sans en modifier la forme

Amplificateur

Composants électroniques permettant de multiplier un signal par une constante, appelée gain

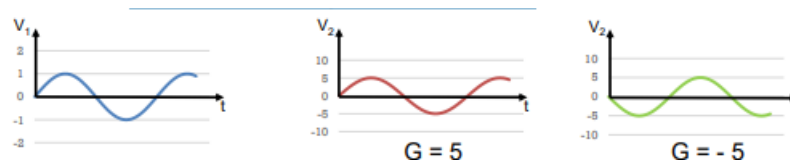


Gain

$G = \text{Constante}$

Si $G > 0 \Rightarrow$ Amplificateur non inverseur

Si $G < 0 \Rightarrow$ Amplificateur inverseur



Applications

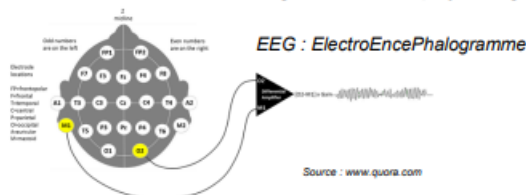
Applications :

- Amplificateur de guitare



Source : fr.audiorzine.com

- Amplificateur biomédical \Rightarrow Signaux électrophysiologiques

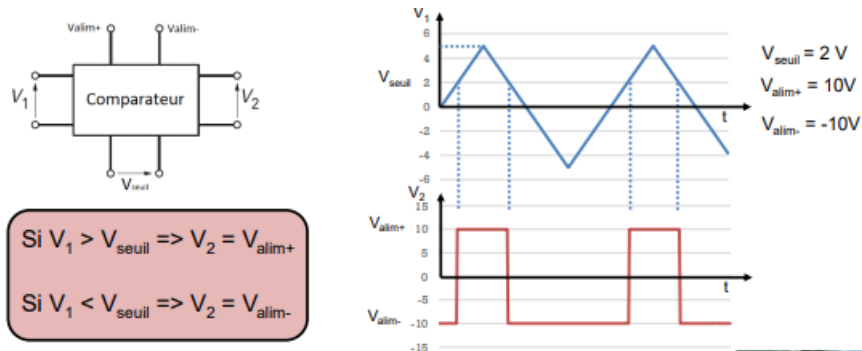


Comparaison

Objectif de la comparaison : Comparer l'amplitude d'un signal à une amplitude de référence appelée seuil

Comparateur

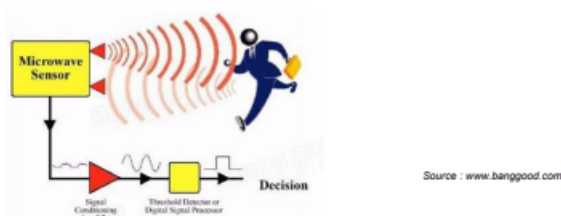
Composants électroniques permettant de générer une sortie avec uniquement 2 états (haut et bas)



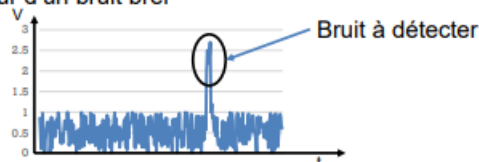
=> transformation d'un signal triangulaire en signal carré (mais les 2 sont des signaux analogiques)

Applications

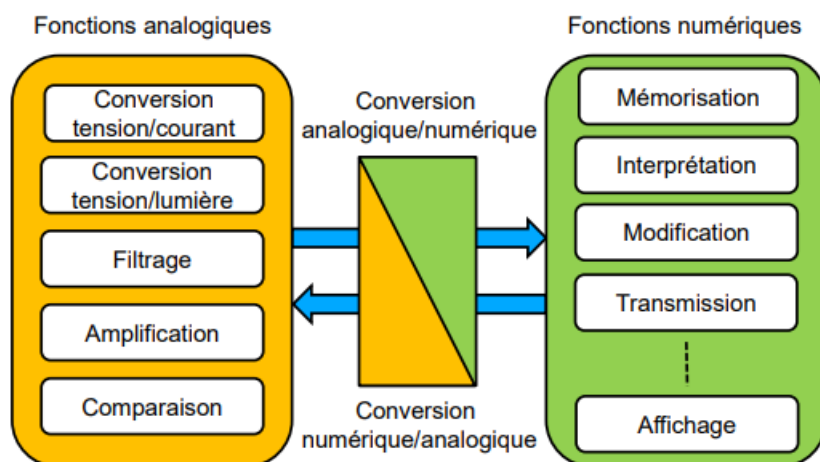
- Détecteur de mouvement



- Détecteur d'un bruit bref



Bilan fonctions analogiques



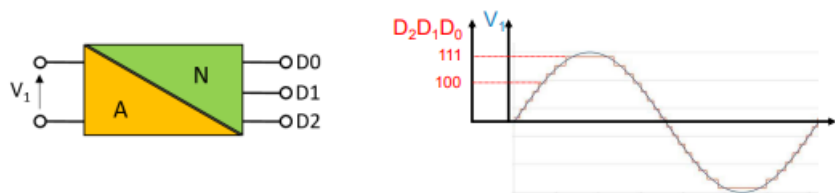
Conversion analogique numérique

Conversion analogique numérique : Fonction indispensable pour interagir avec l'environnement

- Environnement : monde analogique
- Informatique : monde numérique

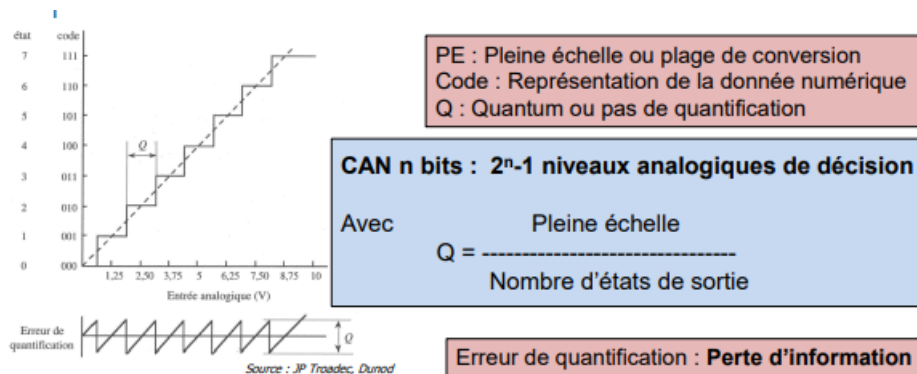
Objectif du convertisseur : Convertir un signal analogique (tension ou courant) en une donnée numérique

Convertisseur : Ensemble de composants analogiques et numériques



Ici, en sortie $2^3 = 8$ valeurs possibles stockées sur 3 bits (V_1 est l'entrée **analogique** et D0, D1, D2 sont les sorties **numériques**)

Fonction de transfert d'un CAN :



Caractérisation de transfert d'un CAN

3 bits : $2^3 = 8$ états possibles
Pleine échelle : 10 V



$$Q = 10/8 = 1,25 \text{ V}$$

Quand on choisit un composant --> choisir les bonnes caractéristiques de ce composant

Paramètres à choisir

- **PE** : pleine échelle ou plage de conversion => plage qui peut être convertie en numérique (en TP : on peut convertir de 0 à 5V)
- **Code** : représentation de la donnée numérique (par exemple on peut coder différemment un nombre négatif ou positif)
- **Q** : quantum ou pas de quantification $Q = \frac{\text{pleine échelle}}{\text{nombre d'états de sortie}}$

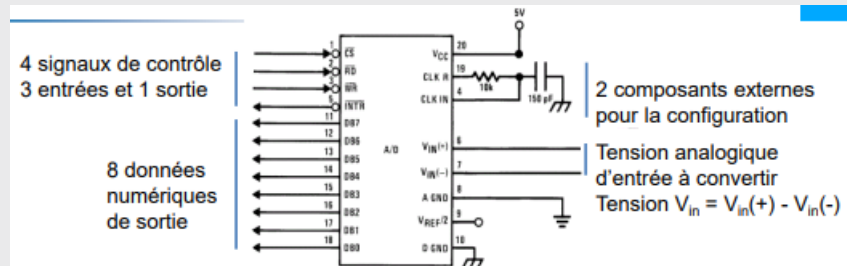
Toujours une perte d'information mais qui peut être négligeable

Pour la réduire (donc pour réduire Q) :

- augmenter le nombre de bits du convertisseur
- réduire la pleine échelle

Résolution, nombre d'états et valeur du LSB en fonction de la pleine échelle

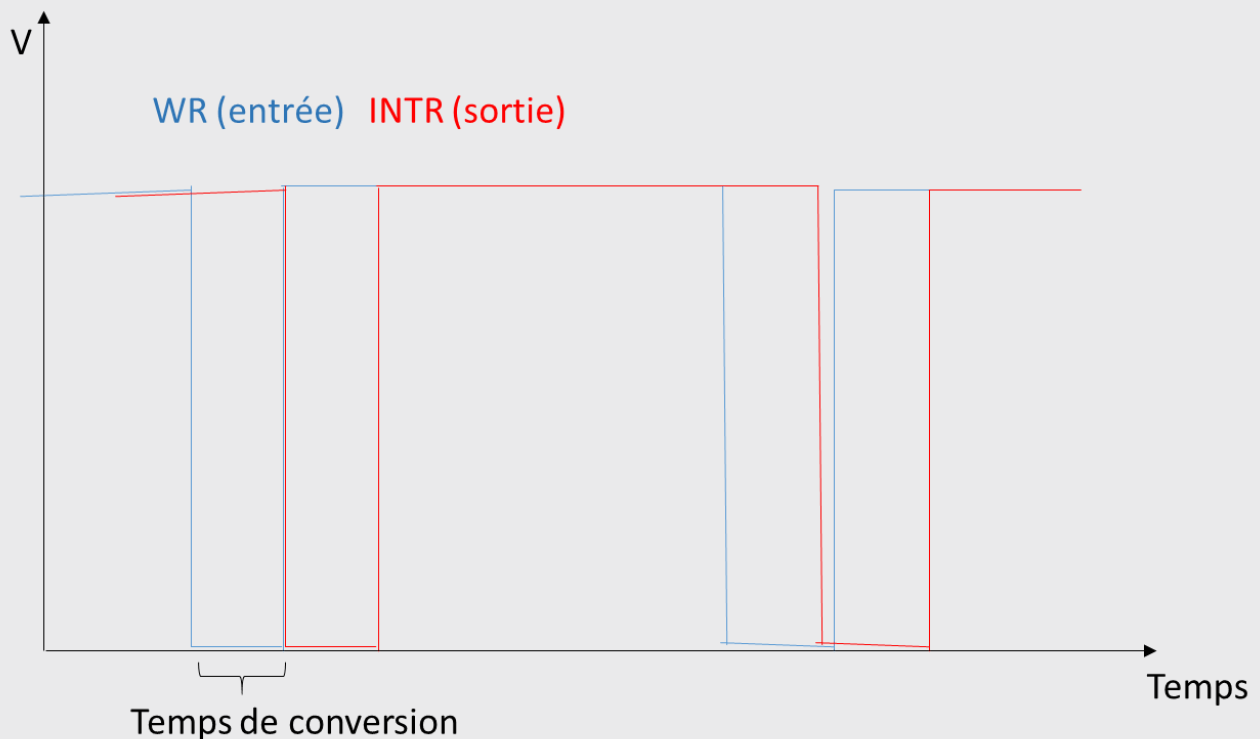
Résolution N	Nombre d'états 2^N	Valeur du LSB (% de la pleine échelle)	Valeur du LSB avec 10 V de pleine échelle (mV)
8	256	0,390 625	39,062
10	1 024	0,097 656	9,765
12	4 096	0,024 414	2,441
14	16 384	0,006 104	0,610
16	65 536	0,001 526	0,152
18	262 144	0,000 381	0,038
20	1 048 576	0,000 095	0,009
22	4 194 304	0,000 024	0,002
24	16 777 216	0,000 006	0,000 596
26	67 108 864	0,000 001	0,000 149



sortie du CAN associée à des LEDs : Exemple si $DB7=0$ alors la LED L7 est allumée, si $DB7=1$ alors la LED L7 est éteinte

Entrées et sorties :

- entrées de contrôle /CS et /RD mises à 0 => Connectées au GND (0 volt)
- entrée /WR : signal généré par un composant externe au CAN (le microcontrôleur)
 - ordre de conversion (écriture) si /WR passe de 1 à 0 => **début conversion**
- sortie /INTR : signal **généré** par le CAN
 - information de fin de conversion si /INTR = 0 => **conversion terminée**



Observation en TP : il y a des LEDs qui clignotent

=> car le pas de quantification est petit (pour une tension donnée, on a le signal idéal + le bruit et si le bruit est supérieur à 19,5mV (plus petite tension convertie) → LED qui clignote)

Les composants ne sont pas *parfaits* donc il y a **toujours** une incertitude sur les valeurs converties


On ne peut convertir que des valeurs positives de tension avec le CAN utilisé

TP - Carte capteurs : utilisation d'un ADC0804

Caractéristiques

- Résolution : 8 bits
 - 256 données différentes possibles en sortie
- Plage de conversion : 5 volts
 - possibilité de convertir une tension U avec $0 < U < 5 \text{ V}$
- Temps de conversion : $100 \mu\text{s}$
 - possibilité de convertir une tension toutes les $100 \mu\text{s}$
 - fréquence de conversion = $1/100 \mu\text{s} = 10 \text{ kHz}$
- Boîtier (chip) : 20 broches (pins)
 - 4 signaux de contrôle
 - 8 signaux de données \Rightarrow 8 bits
 - 2 signaux d'entrée \Rightarrow Tension à convertir
 - 2 alimentations
 - 3 signaux de configuration

i Temps de conversion

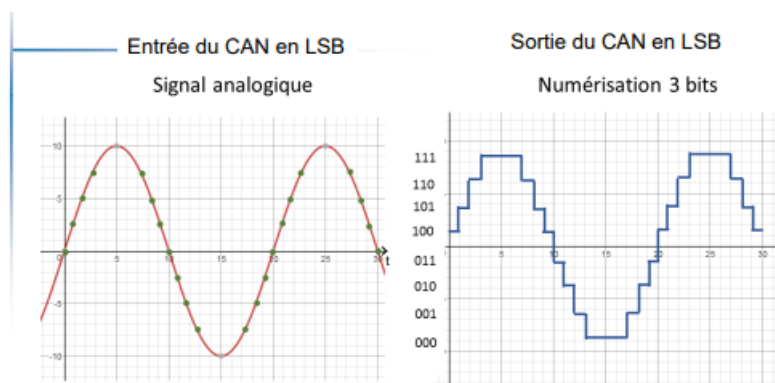
Le **temps de conversion** est le temps nécessaire pour effectuer une action  important dans les caractéristiques d'un composant

\Rightarrow on ne **peut pas** utiliser le convertisseur avec une fréquence supérieure à 10 kHz

Temps de conversion \neq temps d'échantillonnage

🔥 Pour convertir un signal avec des valeurs négatives

On peut utiliser un **additionneur** pour avoir des valeurs uniquement positives (si le CAN ne peut convertir que des données positives)





Source : <http://electroniqueamateur.fr>

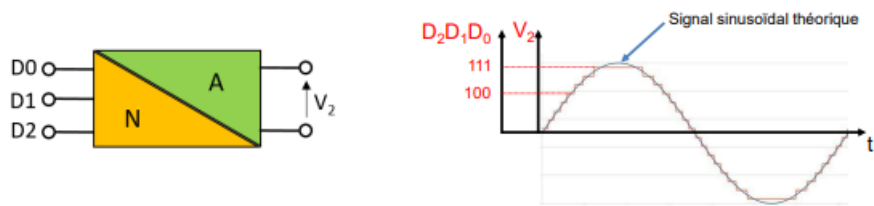
Signal numérique \neq Signal analogique \rightarrow Perte d'information

Conversion numérique analogique

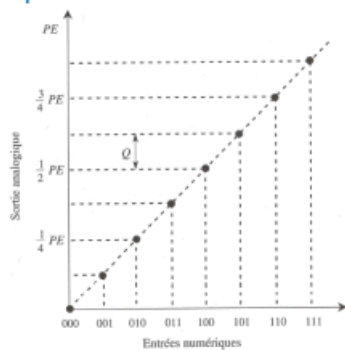
i Conversion numérique analogique : Fonction indispensable pour interagir avec l'environnement

-  Environnement : monde analogique
-  Informatique : monde numérique

Objectif du convertisseur : Convertir un signal numérique (tension ou courant) en une donnée analogique



Caractéristique de transfert d'un CNA



PE : Pleine échelle ou plage de conversion
Code : Représentation de la donnée numérique
Q : Quantum ou pas de quantification

Une combinaison d'entrée génère une valeur analogique correspondant à une fraction de la PE

Pas de perte d'information

Source : JP Troadec, Dunod

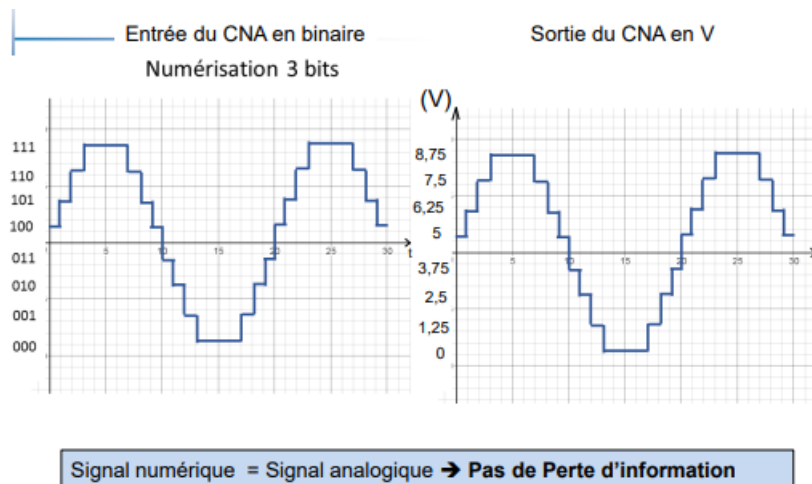
Caractérisation de transfert d'un CNA 3 bits

Pas de perte d'information => au contraire on "fabrique" de l'information car on crée les données qui se trouvent entre les paliers (on passe de données discrètes à continues)

⚠ Fonction de transfert du CNA

Ne pas relier les points entre eux quand on trace la fonction de transfert d'un CNA — sinon on crée de l'information

Le CNA ne prend en entrée que des données discrètes



💧 Principe du codage

En numérique : utilisation de la base 2 pour représenter un signal numérique car on utilise des composants qui utilisent 2 états (passant/bloquant)

Tout entier N positif peut être représenté en numération binaire :

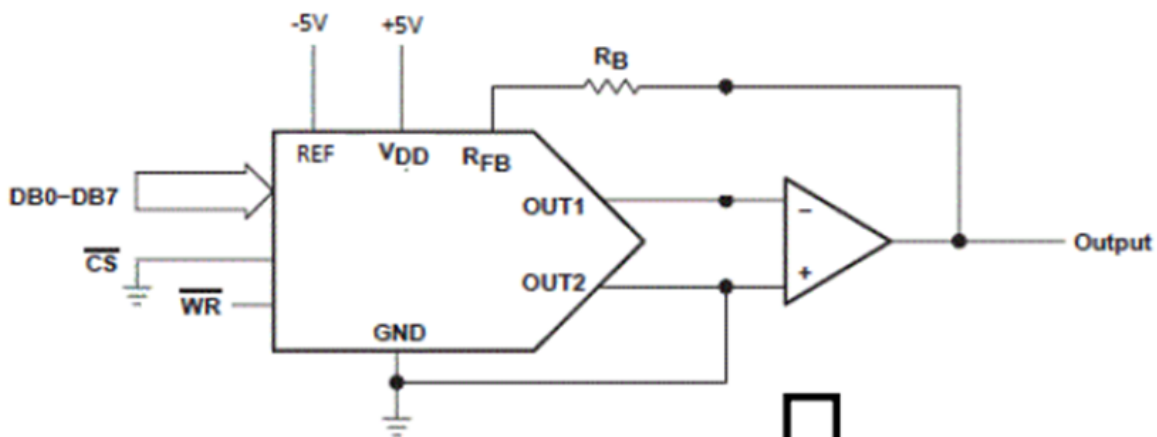
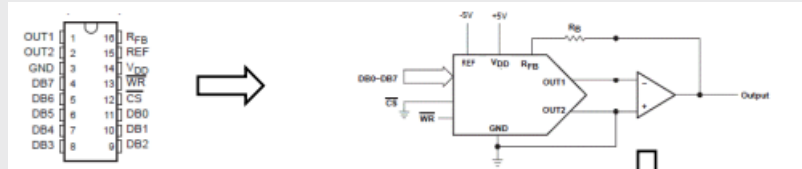
$$N = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_1 2^1 + a_0 2^0 \text{ avec } (a_i = 0 \text{ ou } 1)$$

☰ Exemple sur 3 bits

$$N_{10} = 6 = 12^2 + 12^1 + 0 \cdot 2^0$$

$$N_2 = 110$$

i en TP : utilisation d'un TLC7524



DIGITAL INPUT (see Note 3)		ANALOG OUTPUT
MSB	LSB	
1	1	$-V_{ref} (255/256)$
1	0	$-V_{ref} (129/256)$
1	0	$-V_{ref} (128/256) = -V_{ref}/2$
0	1	$-V_{ref} (127/256)$
0	0	$-V_{ref} (1/256)$
0	0	0

NOTE 3: $LSB = 1/256 (V_{ref})$

chaque donnée numérique (entrée) est connectée à une sortie du microcontrôleur

/CS : entrée de contrôle mise à 0 => Connectée au GND

/WR : entrée et signal généré par un composant externe (microcontrôleur) au CNA ordre de conversion (écriture)
si /WR passe de 1 à 0 => début conversion

L'environnement logiciel Arduino

i IDE : Integrated development environment

Fonctions de l'IDE :

- compilation du code programme
- transfert du code compilé dans la mémoire flash du microcontrôleur

Voir [Arduino.cc](https://www.arduino.cc)

Un programme = un « sketch »

Un programme : 2 fonctions obligatoires
Setup : Instructions d'initialisation
Loop : Instructions répétées indéfiniment

```
// Déclarations des constants, variables, bibliothèques,...  
void setup()  
{  
  // Instructions exécutées une fois  
}  
  
void loop()  
{  
  // Instructions exécutées indéfiniment  
}
```

L'ordre d'écriture des instructions a une importance : les instructions sont exécutées l'une après l'autre (car les ressources sont limitées)

Quelques fonctions

- **Pause** : `delay(durée du délai)`

Exemple pause de 100 millisecondes : `delay(100);`

Le temps est mesuré de manière relative (par rapport à une horloge qui envoie un signal tous les x temps)

- **Définition des entrées/sorties numériques**

`pinMode(nom de la broche, mode de la broche)`

Exemple : définition d'une sortie :

```
int enable = 14 ;  
pinMode(enable,output);
```

- **Durée d'une impulsion** : `durée = pulseIn(entrée, type, délaimax)`

Exemple : `duration = pulseIn(in1, HIGH);`

- **Modification des sorties numériques** : `digitalWrite(nom de la broche, valeur)`

Exemple : mise à l'état haut d'une sortie enable : `digitalWrite(enable,HIGH)`

- **Lecture des entrées numériques** : `Valeur = digitalRead(nom de la broche)`

Exemple : lecture de l'entrée in1 `Valeur = digitalRead(in1);`

- **Lecture des entrées analogiques** : présence de CAN `Valeur = analogRead(nom de la broche)`

Exemple : lecture de l'entrée ina1 : `Valeur = analogRead(ina1);`

- **Modification des sorties analogiques** : Absence de CNA - utilisation de PWN : `analogWrite(nom de la broche, rapport cyclique)`

Exemple : mise à VCC/2 d'une sortie outa1 (filtrage en aval nécessaire `analogWrite(outa1, 50);`

Communication avec le PC

Utilisation du port série

La carte Arduino UNO met à disposition un port série asynchrone (indépendant de l'horloge) => plus difficile à gérer qu'en synchrone

- Broche 0 « réception » : Rx
- Broche 1 « émission » : Tx

La carte Arduino UNO dispose d'une interface USB pour le chargement du soft

- Création d'un port COM virtuel
- Utilisation possible d'un serialMonitor dans l'IDE ou d'un hyperterminal

Initialisation du port Serial :

```
Serial.begin(vitesse)
```

Fonction à prévoir dans le `setup`

vitesse = Vitesse standard de transmission

Exemple : initialisation du port série asynchrone à 9600 bauds

```
Serial.begin(9600); // 9600 bits/sec
```

Désactivation du port Serial

```
Serial.end(); // broches 0 et 1 peuvent être utilisées
```

Lecture du port série

```
caractère = Serial.read();
```

Ecriture dans le port série

```
Serial.print(données, format);
```

La fonction assure les conversions

Exemples

```
Serial.print(17); //affichage de 17
```

```
Serial.print(17,BIN); //affichage de 10001
```

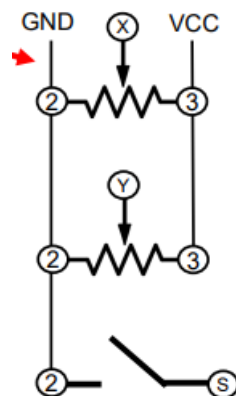
```
Serial.print(« licence »); // affichage de licence
```

```
Serial.println() // avec un retour à la ligne
```

Les capteurs

Joystick

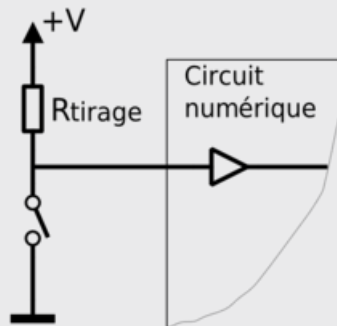
Permet de récupérer la position en X et en Y, et une troisième information (interrupteur)



- Les tensions aux points X et Y, V_X et V_Y sont proportionnelles à la position des points X et Y
- Si l'interrupteur est fermé : $V_S = \text{GND}$
- Si l'interrupteur est ouvert : $V_S = \text{VCC}$ (si on rajoute une résistance dite de pull-up)

La tension aux bornes de la résistance quand l'interrupteur est ouvert est de 0V
--> en ajoutant la résistance de pull-up : cela impose un potentiel égal à 5V sur l'interrupteur

Quote



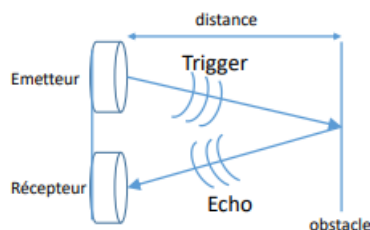
title: Résistance de pull-up/tirage avec un interrupteur

Lorsqu'un simple **interrupteur** mécanique est utilisé comme entrée à un circuit logique, une résistance de tirage est nécessaire pour ne pas laisser flottante l'entrée du circuit. La résistance de tirage sert dans ce cas-ci à imposer le niveau haut tandis que l'interrupteur peut encore imposer un niveau bas. Sans résistance de tirage, l'entrée serait flottante et ainsi donc à un niveau logique indéfini ce qui pourrait causer un fonctionnement erratique du circuit.

Capteur ultrason

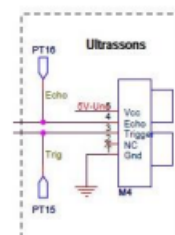
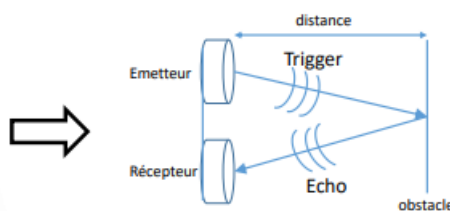
Objectif : Calcul de la distance entre le capteur et un obstacle

Principe : Calcul du temps entre l'instant d'émission d'un signal ultrason et l'instant de réception de ce signal après réflexion sur un obstacle



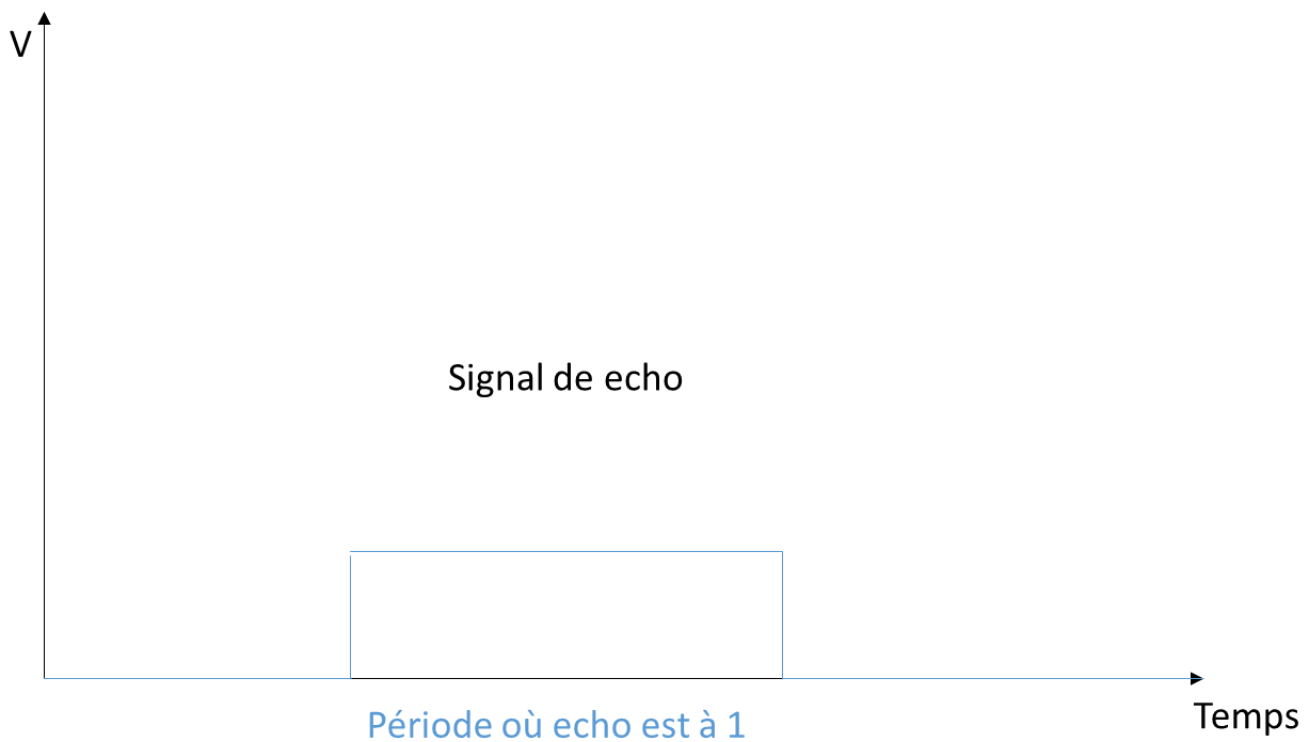
Applications :

- radar de recul
- robotique : capteur de positionnement



Trigger : Entrée du capteur – ordre d'émission généré par le microcontrôleur

Echo : Sortie du capteur – information récupérée par le microcontrôleur

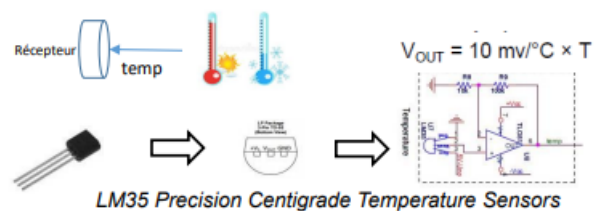


Rôle du microcontrôleur :

- Contrôler le capteur => Trigger
- Récupérer l'information => Echo
- Interpréter l'information
- Calculer la distance (grâce à la vitesse du son, car le calcul est simple à effectuer sinon on aurait pu effectuer le calcul avec le PC)
- Transmettre la distance

Capteur de température

Objectif : Calcul de la température ambiante
Principe : Mesure d'une tension qui varie en fonction de la température ambiante



V_{out} : Tension analogique

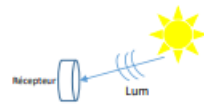
Rôle du microcontrôleur :

- Récupérer l'information => V_{out}
- Convertir la tension analogique en numérique
- Interpréter l'information
- Transmettre la température

Capteur de lumière

Objectif : mesurer le taux de luminosité ambiant (éclairage)

Principe : utilisation d'une cellule photoconductrice dont la résistance varie en fonction de la luminosité => Tension analogique



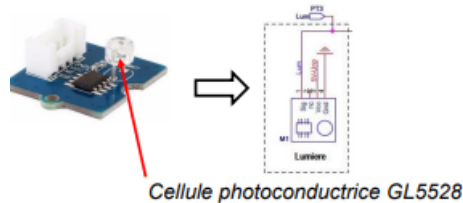
$$1 \text{ lux} = 1 \text{ lumen/m}^2$$

Activité ou lieu concerné	Éclairement moyen
Sensibilité d'une caméra	0,001 lux
Nuit de pleine lune	0,5 lux
Rue de nuit bien éclairée	20 à 70 lux
Local de vie	100 à 200 lux
Appartement bien éclairé	200 à 400 lux
Local de travail	200 à 3 000 lux

Source : wikipedia

Applications :

- Adaptation de l'éclairage à la luminosité ambiante
- Automobile : allumage de phare automatiquement



Lum : tension analogique

Rôle du microcontrôleur :

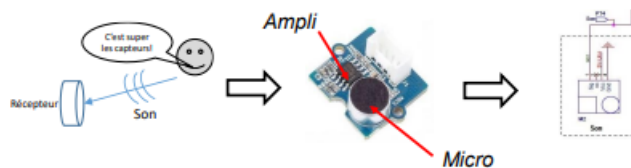
- Récupérer l'information => Lum
- Convertir la tension analogique en numérique
- Interpréter l'information
- Transmettre la luminosité

Capteur de son

Capteur de son

Objectif : mesurer l'ambiance sonore

Principe : utilisation d'un microamplifié => Tension analogique



Son : Tension analogique

Rôle du microcontrôleur

- Récupérer l'information => son
- Convertir la tension analogique en numérique
- Traitement de l'information => filtrage, moyennage, détection de max, ...
- Extraire et transmettre l'information

Fusion des données

Pour transmettre des données "capteurs" à une machine (PC)

- Utilisation de la liaison série du microcontrôleur et USB du PC
- Définition d'un protocole : langage commun
- Ecriture du code