



# Expérimenter

## Chaine d'information

### Communication et information

### Algorithme et Algorithme



**Traitement :**

Lire la valeur de  $N$ ;

**pour**  $n$  allant de 1 à  $N$  **faire**

**si** le reste de la division de  $n$  par 2 est nul **alors**

    Affecter  $n^2$  à  $u$

**sinon**

    Affecter  $n - 2$  à  $u$

**fin**

    Affecter  $s + u$  à  $s$ ;

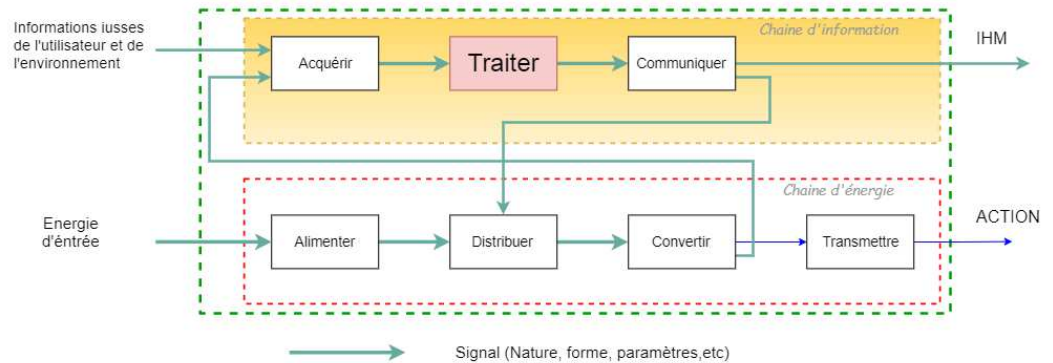
**fin**

**Sorties :** Afficher la valeur de la somme  $s$ .

<b>1.</b>	<b>TRAITEMENT DE L'INFORMATION</b>	<b>3</b>
<b>1.1.</b>	<b>Les supports techniques</b>	<b>3</b>
1.1.1.	Logique câblée	3
1.1.2.	Logique programmée	4
<b>1.2.</b>	<b>Structure d'un système programmable</b>	<b>4</b>
1.2.1.	Le microcontrôleur	4
1.2.2.	La mémoire morte (R.O.M)	5
1.2.3.	La mémoire vive (R.A.M)	5
1.2.4.	Les bus analogiques	5
1.2.5.	Les bus logiques	5
<b>1.3.</b>	<b>Exemple : Carte ARDUINO UNO</b>	<b>6</b>
<b>2.</b>	<b>ALGORITHME ET PROGRAMME</b>	<b>7</b>
<b>2.1.</b>	<b>Algorithme et algorithme</b>	<b>7</b>
2.1.1.	Algorithme	7
2.1.2.	La sémantique et la syntaxe algorithmique	7
2.1.3.	Algorithme	8
2.1.4.	Variables	9
2.1.5.	Fonctions	9
2.1.6.	Structures linéaires ou séquentielles	9
2.1.7.	Structures itératives	10
2.1.8.	Structures conditionnelles (alternatives)	12
<b>2.2.</b>	<b>Langage informatique</b>	<b>13</b>
2.2.1.	Python	13
2.2.2.	ARDUINO	14
2.2.3.	SCILAB	14
<b>2.3.</b>	<b>Comportement d'un objet à partir d'une description à événements discrets</b>	<b>14</b>
2.3.1.	Logique combinatoire	15
2.3.2.	Logique séquentielle	15
<b>3.</b>	<b>LES FONCTIONS LOGIQUES</b>	<b>16</b>
<b>3.1.</b>	<b>Mise en situation</b>	<b>16</b>
<b>3.2.</b>	<b>Symbole logique</b>	<b>17</b>
<b>3.3.</b>	<b>Schéma à contacts ou électrique</b>	<b>17</b>
<b>3.4.</b>	<b>Table de vérité</b>	<b>18</b>
<b>3.5.</b>	<b>Chronogramme</b>	<b>18</b>
<b>3.6.</b>	<b>Remarques sur la norme ISO 5784</b>	<b>18</b>
<b>4.</b>	<b>LA NORME (NF ISO 5784)</b>	<b>19</b>

4.1.	Fonctions OUI et NON	19
4.2.	Fonctions OU et ET	19
4.3.	Fonctions OU NON et ET NON	20
4.4.	Fonction OU EXCLUSIF	21
5.	FONCTION LOGIQUE : SYNTHESE	22
6.	ALGEBRE DE BOOLE	23
7.	REALISATION TECHNOLOGIQUES DES FONCTIONS LOGIQUES	23
7.1.	Technologie électronique	23
7.2.	Technologie électrique	24
7.3.	Technologie pneumatique	24

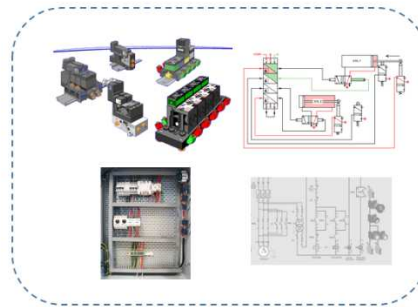
# 1. Traitement de l'information



Dans la chaîne d'information, la fonction «Traiter» reçoit des signaux en provenance de la fonction «Acquérir». Ces signaux, mis en forme ou non, sont issus de capteurs externes ou internes au système. Le traitement permettra l'analyse des informations, leur structuration et leur transmission vers la fonction «Communiquer».

## 1.1. Les supports techniques

Logique câblée



Logique programmée



### 1.1.1. Logique câblée

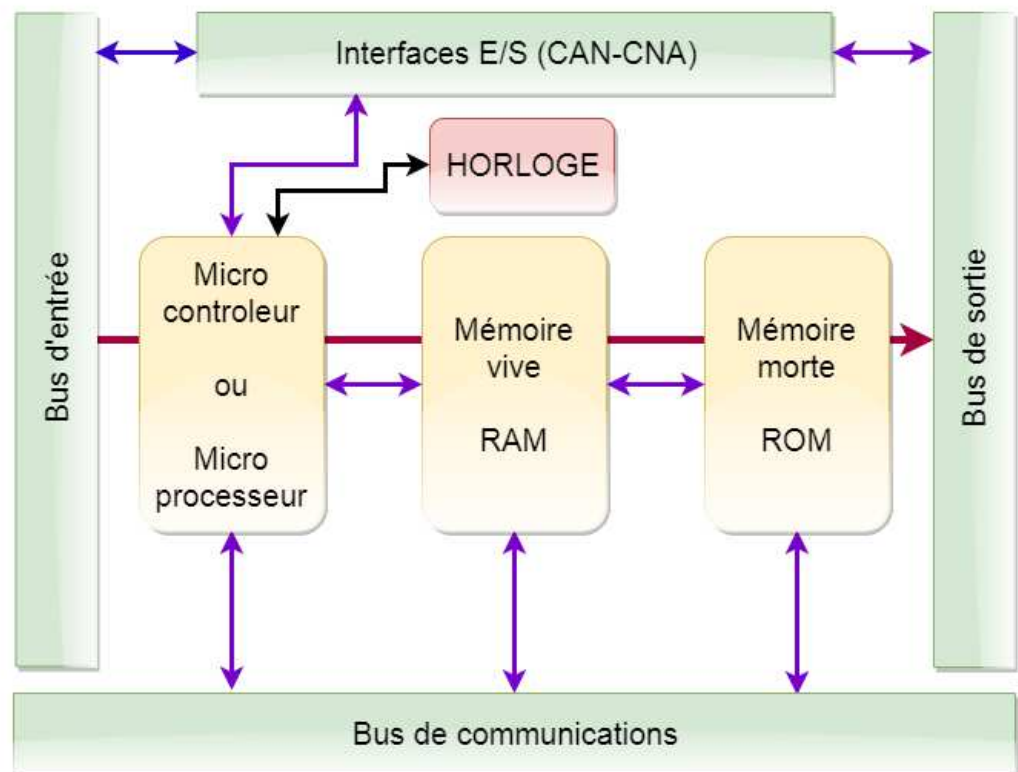
Les solutions câblées sont souvent anciennes mais elles présentent l'avantage d'être robustes et ne nécessitent pas de compétence de programmation. Elles consistent à réaliser un circuit (électrique, pneumatique, hydraulique) spécifique à l'application. Il est difficile de modifier le comportement du système une fois qu'il a été élaboré. Certains composants restent paramétrables (régulateur de vitesse, temporisations, etc.).

### 1.1.2. Logique programmée

Le comportement du système est régi par **un programme** qui nécessite une connaissance du langage utilisé. Il existe plusieurs technologies :

- **Les Automate Programmable Industriel (API)** utilisés dans les applications industrielles sur des machines automatisées et des systèmes ne nécessitant pas de programmation complexe.
- **Les microcontrôleurs et micro-ordinateurs** sont quant à eux plus performants et permettent un nombre de calculs plus important. Ils sont très présents dans les systèmes industriels de grande diffusion. C'est aussi un des moyens utilisés pour prototyper une solution avant son industrialisation. (ARDUINO, RaspberryPi).

## 1.2. Structure d'un système programmable



### 1.2.1. Le microcontrôleur

Un microcontrôleur est un composant électronique (circuit intégré) qui regroupe toutes les fonctions d'un système minimal (Processeur, RAM, ROM, et périphériques d'entrée sortie).

Le microprocesseur ou microcontrôleur effectue les opérations élémentaires qui ont été structurées dans le code représentant la suite logique des actions que doit mener le système (Programme = code machine). Le code est stocké dans la mémoire morte. La communication avec l'extérieur s'effectue par les bus d'entrée et de sortie. D'autres bus sont disponibles en fonction des applications souhaitées. Les bus transportent l'information.

Une mémoire se comporte comme un meuble à tiroir : les informations stockées dans la mémoire sont appelées « données », et chaque tiroir possède une « adresse ».

#### **1.2.2. La mémoire morte (R.O.M)**

Les données contenues dans la R.O.M (Read Only Memory) ne peuvent être effacées ni lors du fonctionnement du système, ni lorsqu'il est mis hors tension. Elle est utilisée pour stocker le code qui sera exécuté par le microprocesseur.

#### **1.2.3. La mémoire vive (R.A.M)**

Les données contenues dans la R.A.M (Random Acces Memory) sont perdues lorsqu'elle est mise hors tension. Le microprocesseur utilise cette mémoire lors de l'exécution du programme.

#### **1.2.4. Les bus analogiques**

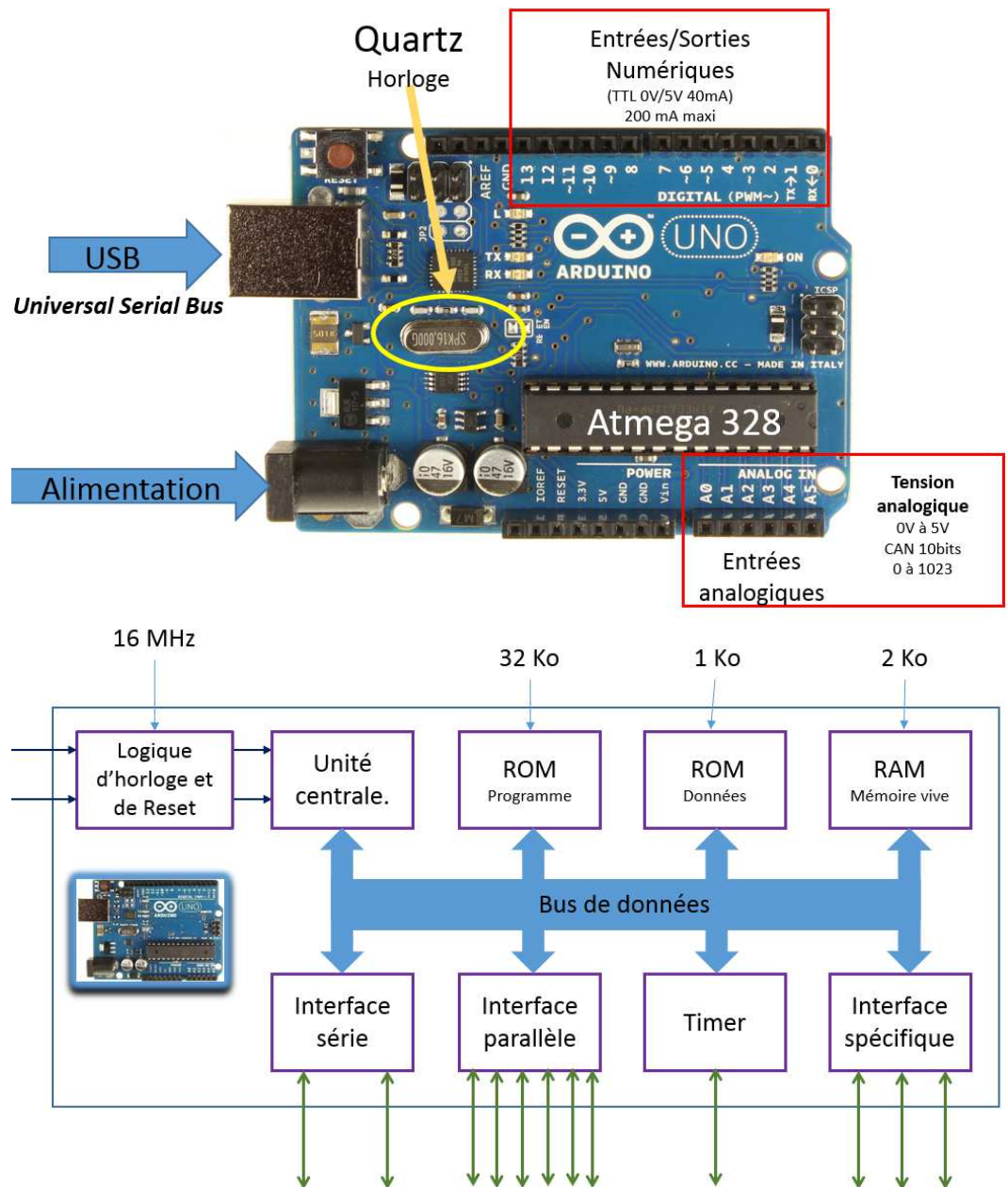
Ils permettent au système de traduire une grandeur électrique qui représente une grandeur physique. Le système dispose de convertisseurs analogiques numériques (CAN) qui convertissent la grandeur analogique en grandeur logique. En sortie il est possible de convertir le signal numérique en signal analogique par l'intermédiaire de convertisseurs numériques analogiques (CNA).

#### **1.2.5. Les bus logiques**

Ils fonctionnent en tout ou rien (TOR). Une entrée ou une sortie logique peut prendre deux valeurs : 0 ou 1, qui correspondent généralement aux tensions 0V et 5V.



### 1.3.Exemple : Carte ARDUINO UNO



## 2. Algorithme et programme

### 2.1. Algorithme et algorithme

Un algorithme est un ensemble de **règles opératoires rigoureuses** ordonnant à un système d'exécuter **dans un ordre déterminé**, un nombre fini d'opérations élémentaires pour résoudre tous les problèmes d'un type donné (NF Z 61-100)

Un algorithme peut être représenté :

- soit littéralement grâce au **langage algorithmique** en respectant un formalisme d'écriture ;
- soit graphiquement à l'aide de **l'algorithme** en respectant un formalisme de symboles et de structure.

#### 2.1.1. Algorithme

*« Un algorithme est une suite de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat, et cela indépendamment des données. »*

Pour écrire un algorithme il faut :

- Analyser le problème et définir avec précision le résultat attendu ;
- Déterminer la suite d'opérations à effectuer pour obtenir le résultat attendu en étant le plus efficace possible ;
- Respecter un formalisme adéquat ;
- Traduire l'algorithme dans un langage de programmation adapté.

Une des meilleures façons de rendre un algorithme clair et compréhensible est d'utiliser **un langage de description structuré** n'utilisant qu'un petit nombre de structures **indépendantes du langage de programmation** utilisé.

Un algorithme doit :

- Etre facile à comprendre par tous ceux qui le lisent ;
- Etre d'une utilisation aisée même par ceux qui ne l'ont pas écrit (Commentaires) ;
- Etre conçu de manière afin de limiter le nombre d'opérations (capacité mémoire).

#### 2.1.2. La sémantique et la syntaxe algorithmique

##### Des mots clés

Ils définiront la structure algorithmique utilisée. Un mot clé est toujours suivi :

- D'une expression conditionnelle écrite entre guillemets ;
- D'un ou plusieurs mots instructions

Exemples de mots clés :

- SI ... ALORS... SINON... : définissent une structure ALTERNATIVE ;
- REPETER... JUSQU'A... : définissent une structure ITERATIVE



### Des mots instructions

Ils définiront les actions qui caractérisent la nature des opérations à effectuer sur une ou plusieurs données. Un mot instruction est suivi entre guillemets :

- De la désignation de l'objet sur lequel il s'applique ;
- De la description de l'opération à appliquer à l'objet.

Exemples de mots instruction :

- **LIRE** « Capteur S1 » ;
- **FAIRE** « Compteur = Compteur + 1 »

### Des mots délimiteurs

Ils fixent :

- Les bornes d'ENTRÉE et de SORTIE de l'algorithme ;
- Les bornes d'ENTRÉE et de SORTIE des différentes structures utilisées dans l'algorithme.










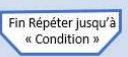
**DEBUT** et **FIN** sont les seuls mots délimiteurs et peuvent être suivis éventuellement d'un mot clé.

### 2.1.3. Algorithme

C'est la représentation d'un algorithme à l'aide de symboles normalisés qui permet une description synthétique du fonctionnement d'un système.

Les organigrammes (algorigrammes) sont largement utilisés pour décrire tous les types de problèmes de traitement de l'information et les moyens de les résoudre.

Ils sont composés de symboles ayant une signification donnée, d'un texte explicatif bref et de lignes de raccordement. Ils sont définis explicitement dans la norme ISO 5807:1985<sup>1</sup>(fr) (*Traitement de l'information— Symboles de documentation et conventions applicables aux données, aux organigrammes de programmation et d'analyse, aux schémas des réseaux de programmes et des ressources de système*).

SYMBOLE	DESIGNATION	SYMBOLE	DESIGNATION
	<b>Début, Fin</b> Début/Fin d'un algorithme, sous programme, programme		<b>Renvoi</b> Assure la continuité d'une ligne
	<b>Initialisation, préparation</b> Opération qui détermine partiellement la voie à suivre dans un embranchement ou un sous programme.		<b>Sous-Programme</b> Portion de programme résumée à son seul titre et détaillée dans un algorithme associé.
	<b>Traitement ou Action</b> Opération ou groupe d'opérations sur des données, instruction.		<b>Test logique</b> Choix à effectuer sur un état avec une réponse binaire : Vrai ou Faux Le point signifie « Faux ».
	<b>Entrée ou Sortie</b> Lecture d'une entrée Ecriture d'une sortie		<b>Commentaire</b> Permet d'apporter des précisions sur n'importe quel symbole.
	<b>Début d'une boucle</b> Utilisé pour signaler le début d'une boucle conditionnelles. Il faut noter la condition de réalisation de la boucle.		<b>Fin d'une boucle</b> Utilisé pour signaler le début d'une boucle conditionnelles. Il faut noter la condition de réalisation de la boucle.

Extrait de la norme ISO 5807:1985

<sup>1</sup> <https://www.iso.org>

#### 2.1.4. Variables

Un programme est composé d'instructions qui travaillent sur des **données** qui sont stockées dans **des variables**. Dit autrement, une variable est un espace de stockage où le programme peut mémoriser une donnée. Une variable doit être initialisée. Une variable peut contenir *toutes sortes de données différentes* (nombre, texte, etc.) il est alors nécessaire de définir le type de la variable.

Une variable est définie par un nom, une adresse, un type, une valeur.

Dans de nombreux langage, (C/C++, Java, Pascal) une variable ne peut contenir *qu'une seule sorte de données*. C'est le type de la variable;

**Une affectation** consiste à donner une valeur à une variable. Elle peut être une constante, le résultat d'un calcul, ou le contenu d'une autre variable. En langage algorithmique la variable à affecter est placée à gauche d'une flèche pointant vers elle. La valeur à donner à la variable est placée à droite de la flèche.

**Syntaxe** : nomDeVariable ← expression

#### 2.1.5. Fonctions

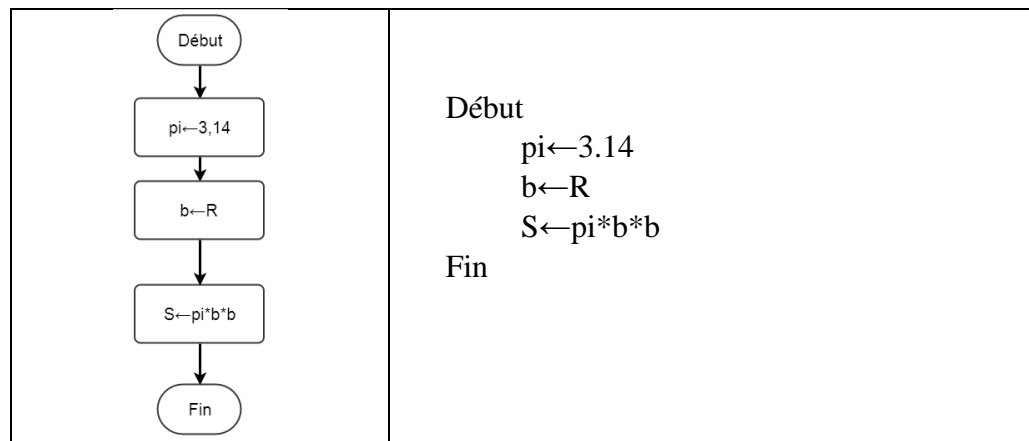
Une fonction exécute des actions et renvoie un résultat. C'est un **morceau de code (programme)** qui sert à faire quelque chose de précis. Comme les variables, les fonctions ont un type, un nom, voire des paramètres.

Elles sont très utiles pour réaliser plusieurs fois la même opération au sein d'un programme. Elles rendent également le code plus lisible et plus clair en le fractionnant en blocs logiques. Elles sont parfois nommées « sous-programmes ».

#### 2.1.6. Structures linéaires ou séquentielles

C'est une suite d'actions à exécuter *successivement* dans l'ordre de leur énoncé. Les mots instructions sont écrits au même rang sur des lignes successives.

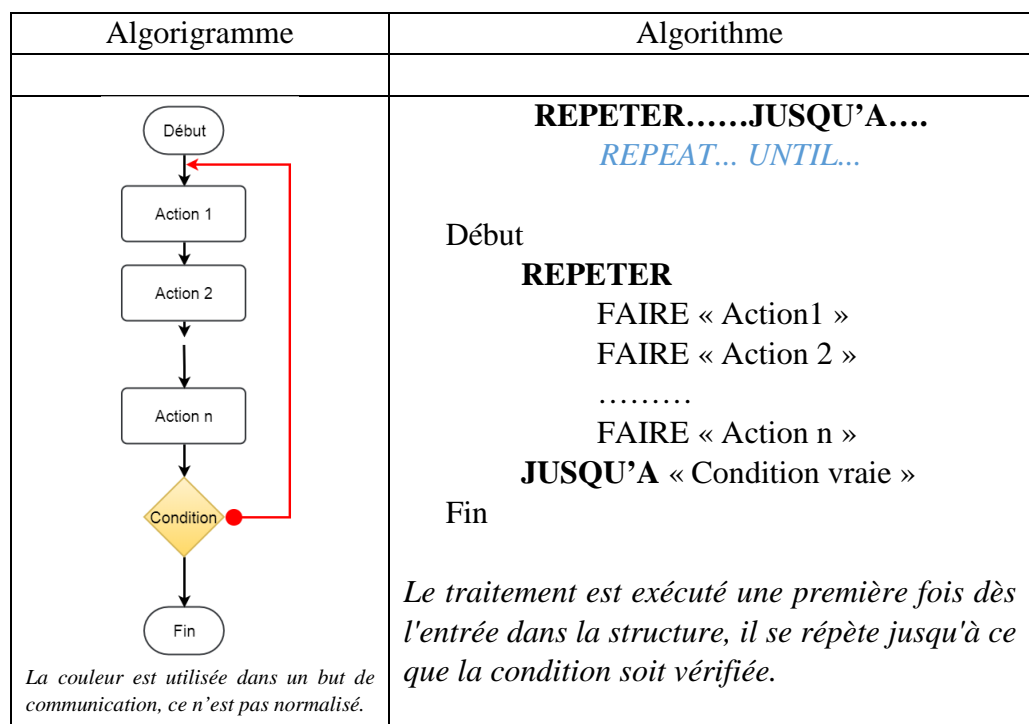
Algorithme	Algorithme
<pre> graph TD     A((Début)) --&gt; B[Action 1]     B --&gt; C[Action 2]     C --&gt; D[Action n]     D --&gt; E((Fin))           </pre>	Début FAIRE « Action1 » FAIRE « Action 2 » ..... FAIRE « Action n » Fin



### 2.1.7. Structures itératives

ITERATION : Répétition de l'exécution d'un traitement. Il existe trois types de structures itératives :

- **REPETER.... JUSQU'A**, le nombre d'itération n'est pas connu à l'avance, il dépendra d'un ou plusieurs éléments extérieurs;
- **REPETER... TANT QUE**, le nombre d'itération n'est pas connu à l'avance, il dépendra d'un ou plusieurs éléments extérieurs;
- **POUR... A... REPETER**, le nombre d'itération est connu, c'est une consigne.



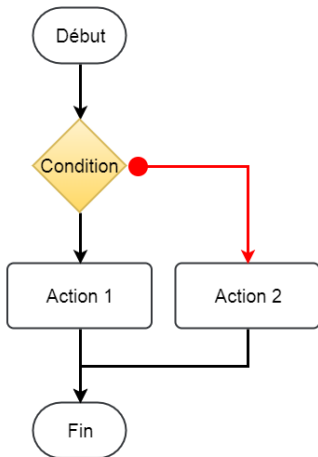
Algorithme	Algorithme
<pre> graph TD     Debut([Début]) --&gt; A1[Action 1]     A1 --&gt; A2[Action 2]     A2 --&gt; An[Action n]     An --&gt; Cond{Condition}     Cond --&gt; A1     Cond --&gt; Fin([Fin]) </pre> <p><i>La couleur est utilisée dans un but de communication, ce n'est pas normalisé.</i></p>	<p><b>REPETER.....TANT QUE....</b>  <i>REPEAT... WHILE.....</i></p> <p>Début</p> <p><b>REPETER</b></p> <p>FAIRE « Action1 »</p> <p>FAIRE « Action 2 »</p> <p>.....</p> <p>FAIRE « Action n »</p> <p><b>TANT QUE</b> « Condition vraie »</p> <p>Fin</p> <p><i>Le traitement est exécuté une première fois dès l'entrée dans la structure, il se répète tant que la condition est vérifiée. Les deux structures itératives « REPETER... JUSQU'A... » et « REPETER... TANT QUE... » sont strictement équivalentes</i></p>

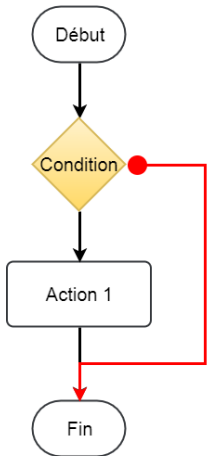
Algorithme	Algorithme
<pre> graph TD     Debut([Début]) --&gt; Vinit[V ← Vinitiale]     Vinit --&gt; Cond{V = Vfinale}     Cond --&gt; Fin([Fin])     Cond --&gt; Vinc[V = V++]     Vinc --&gt; An[Action n]     An --&gt; Cond </pre> <p><i>La couleur est utilisée dans un but de communication, ce n'est pas normalisé.</i></p>	<p><b>POUR....A....REPETER...</b>  <i>FOR... TO... STEP... NEXT...</i></p> <p>Début</p> <p><b>POUR</b> V=Vinitiale <b>A</b> V=Vfinale <b>Pas 1</b></p> <p><b>REPETER</b></p> <p>.....</p> <p>FAIRE « Action n »</p> <p><b>FIN POUR</b></p> <p>Fin</p> <p><i>Dans cette structure la sortie de la boucle d'itération s'effectue lorsque le nombre de répétitions est atteint. La variable « V » de contrôle d'itération est définie par :</i></p> <ul style="list-style-type: none"> <li>– sa VALEUR INITIALE : Vinitiale</li> <li>– sa VALEUR FINALE : Vfinale</li> <li>– son PAS DE VARIATION : P</li> </ul>

### 2.1.8. Structures conditionnelles (alternatives)

Ces structures désignent toute situation n'offrant que deux issues possibles s'excluant mutuellement. Il existe deux types de structures alternatives :

- La structure alternative COMPLETE
- La structure alternative REDUITE

Algorithme	Algorithme
 <p><i>La couleur est utilisée dans un but de communication, ce n'est pas normalisé.</i></p>	<p><i>Alternative complète</i>  <b>SI...ALORS....SINON....FIN SI...</b>  <i>IF....THEN....ELSE...</i></p> <p>Début  <b>SI</b> « Condition Vraie »              <b>FAIRE</b> « Action n »              <b>SINON</b>                  <b>FAIRE</b> « Action 2 »              <b>FIN SI</b></p> <p>Fin</p> <p><i>L'exécution d'un des deux traitements dépend du résultat d'un test :          Si le test est VRAI le premier traitement est exécuté.          Si le test est FAUX c'est le deuxième traitement qui s'effectue.          Structure nommée aussi « Divergence en OU » ou « Sélection de séquence »</i></p>

Algorithme	Algorithme
 <p><i>La couleur est utilisée dans un but de communication, ce n'est pas normalisé.</i></p>	<p><i>Alternative incomplète</i>  <b>SI...ALORS....FIN SI...</b>  <i>IF....THEN...</i></p> <p>Début  <b>SI</b> « Condition Vraie »              <b>ALORS</b> « Action n »              <b>FIN SI</b></p> <p>Fin</p> <p><i>La situation correspondant à la validation de la condition entraîne l'exécution du traitement dans le cas où la condition n'est pas satisfaite, le traitement n'est pas exécuté et la structure est abandonnée.</i></p>





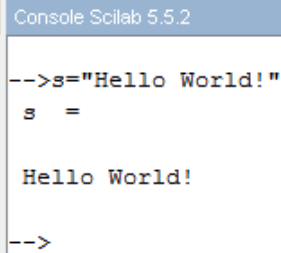
### 2.2.2. ARDUINO

Le langage Arduino est très proche du C et du C++. Cette application est particulièrement utile pour le développement de systèmes nécessitant une logique programmée.

*(Voir applications dans la séquence sur les capteurs)*

*L'IDE Arduino est gratuit, sous **licence libre**.*

### 2.2.3. SCILAB



```

Console Scilab 5.5.2

-->s="Hello World!"
s =

Hello World!

-->

```

**Exemple :**

A l'invit' :

--> s=« Hello World ! »

s=

Hello World !

-->

*L'application SCILAB est gratuite, sous **licence libre**.*

## 2.3. Comportement d'un objet à partir d'une description à événements discrets

« Dans un certain nombre de systèmes conçus par l'homme, tels que les réseaux de communication et d'ordinateurs, les systèmes informatiques, les unités centrales d'ordinateurs elles-mêmes, l'essentiel de **l'enchaînement dynamique des tâches** provient de **phénomènes de synchronisation**, et d'exclusion mutuelle ou de compétition dans l'utilisation de ressources communes, ce qui nécessite une politique d'arbitrage ou de priorité, questions généralement désignées sous le terme générique d'ordonnancement. De tels systèmes, dans lesquels **les transformations sont déclenchées par des événements ponctuels**, typiquement **l'arrivée d'un signal ou l'achèvement d'une tâche**, sont appelés depuis le début des années 80 « **Systèmes à Événements Discrets (SED)** ».<sup>2</sup>

Le mot «discret» ne signifie ni «temps discret», ni «état discret», mais réfère au fait que la dynamique est composée d'événements, qui peuvent être des *débuts* et *fins* de tranches d'évolution continue mais qui seuls nous préoccupent dans la mesure où les fins conditionnent de nouveaux débuts.

<sup>2</sup> J.-M. Delosme –université Evry

### 2.3.1. Logique combinatoire

Un système logique combinatoire est un système binaire pour lequel à un état des variables d'entrée correspond **un unique état** des variables de sorties (La réciproque n'est pas vraie).

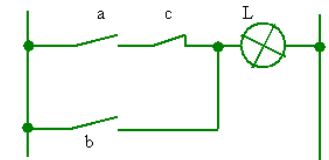
Les problèmes de logique combinatoire conduisent à l'établissement de pures combinaisons dans lesquelles la notion de temps n'intervient pas. Les états des variables d'entrée sont seuls à considérer.

Les états des conditions (a, b et c dans le circuit ci-contre) permettra de mettre en équation le fonctionnement ou non de la lampe.

Pour qu'elle soit allumée, nous pouvons agir sur l'interrupteur « b » OU sur l'interrupteur « a » ET ne pas agir sur l'interrupteur « c ».

Cela se traduit sous forme d'équation avec l'algèbre de Boole<sup>3</sup> :

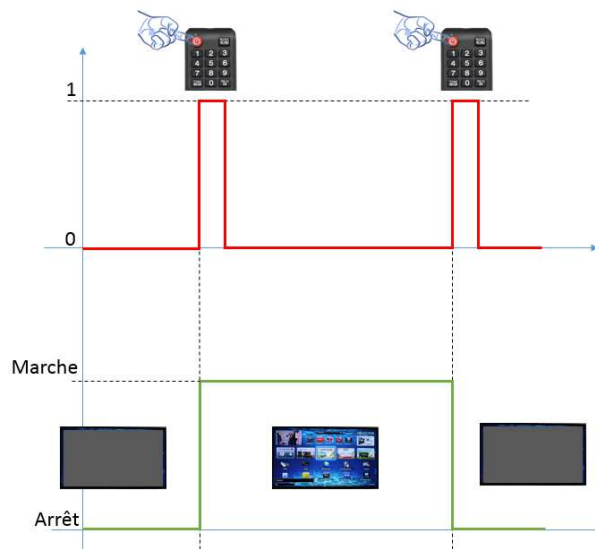
$$L = (a \cdot \bar{c}) + b$$



### 2.3.2. Logique séquentielle

Un système logique est dit séquentiel si les sorties ne dépendent pas uniquement des entrées mais aussi de l'évolution antérieure du système (historique). En effet, à un état des entrées peuvent alors correspondre plusieurs états des sorties. Il y a alors nécessité de faire apparaître de nouvelles variables internes (stockées dans des mémoires).

L'exemple ci-dessous montre que l'appui sur **un seul bouton** peut générer **deux actions** sur le système.



Trame : Adresse MAC (Média Acces Control), adresse liée à la carte réseau.

Paquet : Adresse IP (Internet Protocol), adresse liée au périphérique.

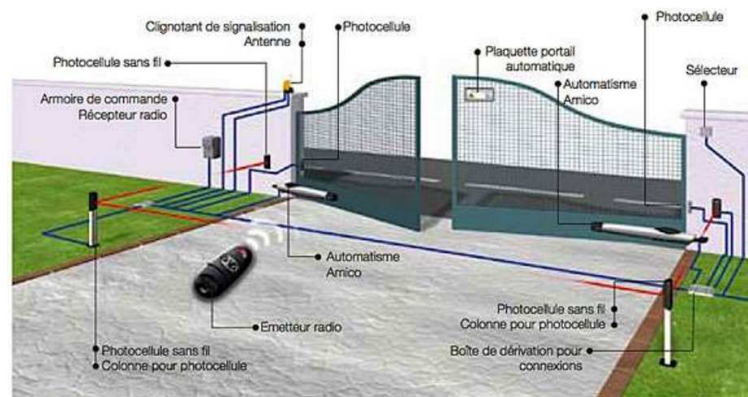
<sup>3</sup> Algèbre de Boole et théorème de de Morgan : Mathématiques adaptées aux équations logiques

### 3. Les fonctions logiques

#### 3.1. Mise en situation

Prenons comme exemple de système un portail motorisé. Pour provoquer son ouverture les conditions suivantes sont nécessaires :

- Le portail est fermé  
ET
- La présence d'une commande filaire  
OU
- La présence d'une commande non filaire  
ET
- Qu'il n'y ait pas d'obstacle à son ouverture



Ces différentes conditions, ET, OU, doivent être traduites dans la programmation du dispositif, mais aussi avec du matériel. Les informations seront binaires pour que le système puisse piloter les actionneurs.

Une **variable binaire** est un élément qui ne peut prendre que 2 états logiques. Par convention à l'un des états de la variable binaire est la valeur "0", et à l'autre la valeur "1".

Les systèmes programmés et/ou automatisés utilisent des **variables booléennes**. Ces variables sont représentatives des informations issues du système (pression, tension, etc.).

**Les niveaux** de ces variables physiques sont appelés niveau Bas (L : low) et Niveau Haut (H : high). En électronique, un niveau est une plage de tension (0 à +5V).

**Une fonction logique** traduit la relation qui existe entre les états logiques des variables **d'entrée et de sortie**.

La fonction logique sera réalisée par un opérateur binaire. A cette expression peuvent être associé d'autre mode de représentation :

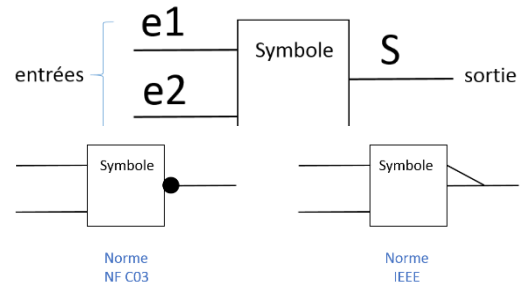
- symbole logique,
- schéma à contact ou électrique,
- table de vérité,
- chronogramme.

### 3.2. Symbole logique

C'est une représentation schématique de l'opérateur. Selon les normes, Norme NFC03 (Norme Française C03) et IEEE, le symbole représentatif est constitué d'un rectangle, dans lequel est placé un signe distinctif de la fonction logique.

L'entrée ou les entrées de l'opérateur se situent à gauche et la sortie à droite.

Un rond (NFC03) ou un triangle (CEI) sur une entrée ou une sortie indique sa négation logique.

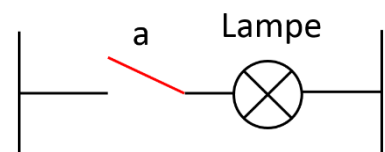


La norme NF ISO 5784 spécifie que le symbole se trouve dans le tiers supérieur du cadre.

### 3.3. Schéma à contacts ou électrique

Il est possible d'effectuer les fonctions en logique « câblée ». Pour un contact :

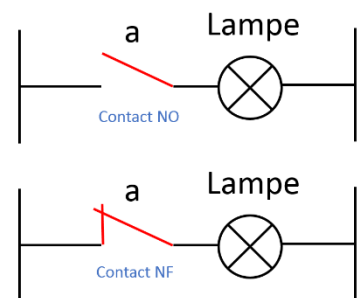
- l'état logique "1" correspond au contact actionné (action physique),
- l'état logique "0" correspond au contact non actionné (pas d'action physique).



Dans l'exemple, si « a » est actionné, c'est-à-dire à l'état « 1 », la lampe est allumée, elle a l'état logique "1" ; Si « a » n'est pas actionné, c'est-à-dire à l'état « 0 », la lampe est éteinte, elle a l'état logique "0".

Il existe deux types de contact (voir cours sur les capteurs) :

- contact à fermeture ou contact "normalement ouvert" au repos (NO). Dans l'exemple ci-contre, la lampe s'allume si le contact est actionné, c'est-à-dire à l'état 1.
- contact à ouverture ou contact "normalement fermé" au repos (NF). Dans l'exemple ci-contre, la lampe s'éteint si le contact est actionné, c'est-à-dire à l'état 1.



Norme NF EN 60617 – CEI 61082

### 3.4. Table de vérité

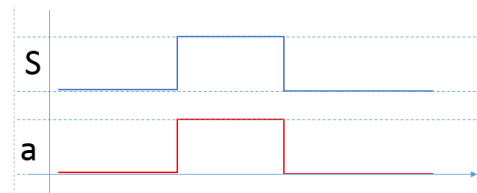
La table de vérité précise toutes les relations possibles entre les différents états des entrées en relation avec l'état de la sortie, elle s'appuie sur la fonction logique traitée.

Lorsqu'il y a "n" variables d'entrées (n colonnes sur la table de vérité), alors il y a "2<sup>n</sup>" combinaisons possibles (2<sup>n</sup> lignes sur la table de vérité).

e1	e2	S
0	0	
0	1	
1	0	
1	1	

### 3.5. Chronogramme

Le chronogramme est une représentation graphique qui permet de visualiser, en fonction du temps, toutes les combinaisons d'états logiques des entrées avec l'état correspondant de la sortie.



### 3.6. Remarques sur la norme ISO 5784

NF ISO 5784-1 Décembre 1988

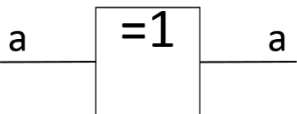
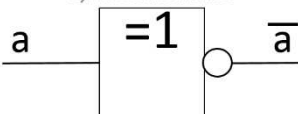
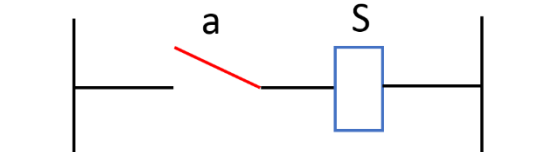
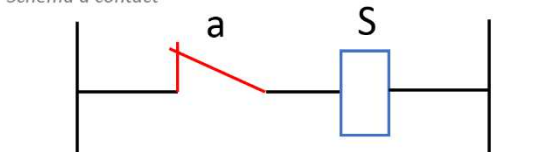
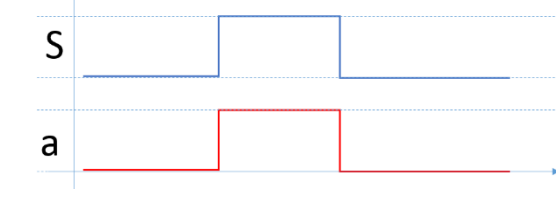
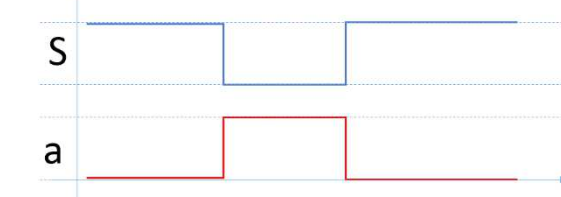
**Transmissions hydrauliques et pneumatiques -  
Logique par les fluides - Partie 1 : symboles pour  
fonctions logiques binaires et connexes.**



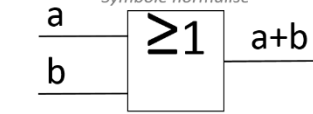
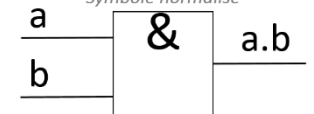
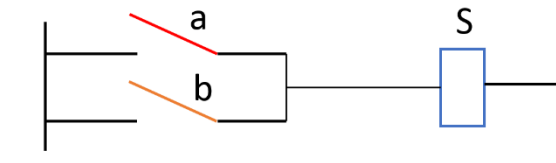
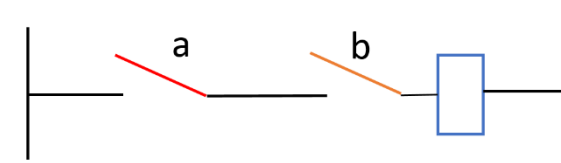
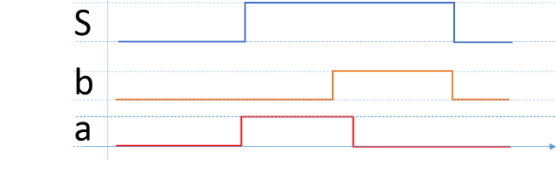
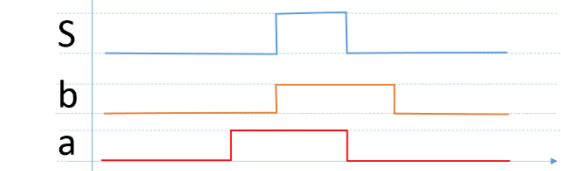
*La présente partie de l'ISO 5784 définit les symboles graphiques pour fonctions logiques binaires et connexes et prescrit certaines règles concernant leurs utilisations dans les schémas. La symbolisation définie par la présente partie de l'ISO 5784 doit être utilisée pour l'établissement de tous plans et schémas relatifs aux fonctions logiques et connexes du traitement de l'information, notamment par les fluides.*

## 4. La norme (NF ISO 5784)

### 4.1. Fonctions OUI et NON

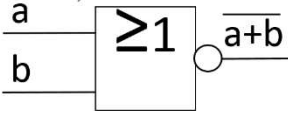
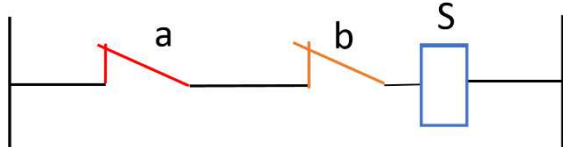

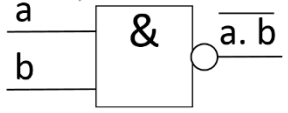
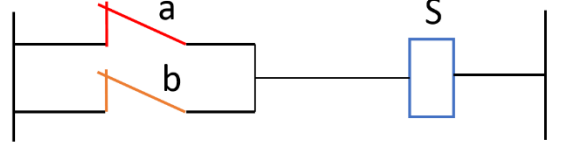
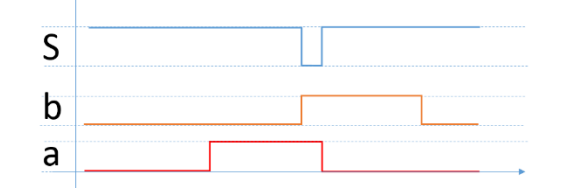
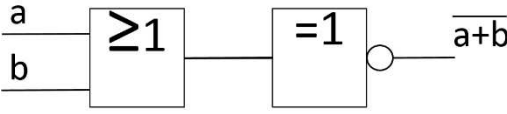
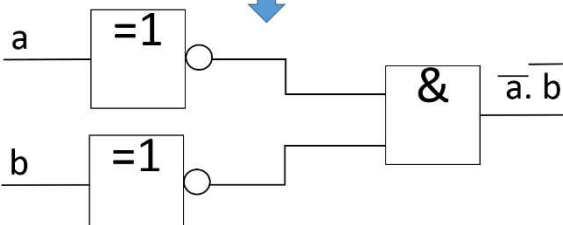
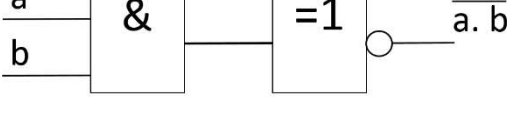
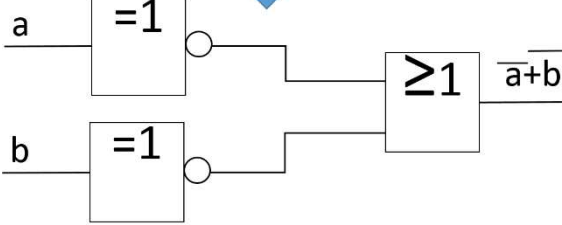
<p><i>Symbole normalisé</i></p>  <p><i>Equation</i></p> $S = a$	<p><i>Symbole normalisé</i></p>  <p><i>Equation</i></p> $S = \bar{a}$
<p><i>Schéma à contact</i></p> 	<p><i>Schéma à contact</i></p> 
<p><i>Chronogramme</i></p> 	<p><i>Chronogramme</i></p> 
<p align="center"><b>FUNCTION OUI</b></p>	<p align="center"><b>FUNCTION NON</b></p>

### 4.2. Fonctions OU et ET

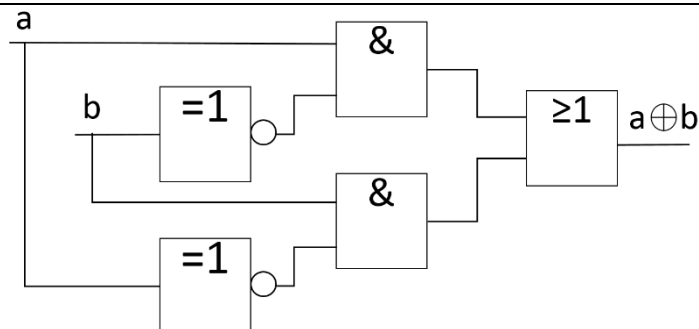
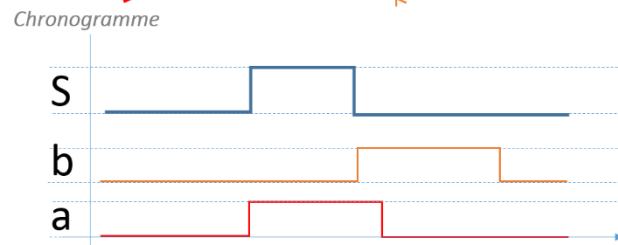
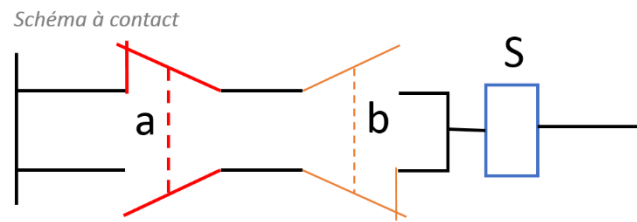
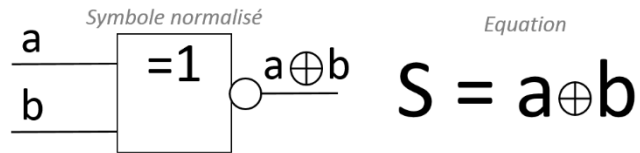
<p align="center"><b>FUNCTION OU</b></p>	<p align="center"><b>FUNCTION ET</b></p>
<p><i>Symbole normalisé</i></p>  <p><i>Equation</i></p> $S = a + b$	<p><i>Symbole normalisé</i></p>  <p><i>Equation</i></p> $S = a . b$
<p><i>Schéma à contact</i></p> 	<p><i>Schéma à contact</i></p> 
<p><i>Chronogramme</i></p> 	<p><i>Chronogramme</i></p> 



## 4.3. Fonctions OU NON et ET NON

FONCTION OU NON NOR	FONCTION ET NON NAND																														
<div><div><div>Symbole normalisé</div><div></div></div><div><div>Equation</div><div><math>S = \overline{a+b}</math></div></div></div> <div><div>Schéma à contact</div><div></div></div> <div><div>Chronogramme</div><div></div></div>	<div><div><div>Symbole normalisé</div><div></div></div><div><div>Equation</div><div><math>S = \overline{a \cdot b}</math></div></div></div> <div><div>Schéma à contact</div><div></div></div> <div><div>Chronogramme</div><div></div></div>																														
<div><div><div></div><div><math>\overline{a+b}</math></div></div><div><div><math>\updownarrow</math></div><div>Théorème de De MORGAN</div></div><div><div></div><div><math>\overline{a} \cdot \overline{b}</math></div></div></div>	<div><div><div></div><div><math>\overline{a \cdot b}</math></div></div><div><div><math>\updownarrow</math></div><div>Théorème de De MORGAN</div></div><div><div></div><div><math>\overline{a+b}</math></div></div></div>																														
<table><tr><th>a</th><th>b</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table> <div>Table de vérité</div>	a	b	S	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>a</th><th>b</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table> <div>Table de vérité</div>	a	b	S	0	0	1	0	1	1	1	0	1	1	1	0
a	b	S																													
0	0	1																													
0	1	0																													
1	0	0																													
1	1	0																													
a	b	S																													
0	0	1																													
0	1	1																													
1	0	1																													
1	1	0																													

## 4.4.Fonction OU EXCLUSIF

FONCTION  
OU EXCLUSIF

a	b	S
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité

## 5. Fonction logique : Synthèse

TYPE	SYMBOLE EUROPEEN	SYMBOLE AMERICAIN	TAB. DE VERITE	EQUIVALENCE SCHEMA ELECTRIQUE												
NON (NOT)			<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><td>A</td><td><u>A</u></td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	Entrée	Sortie	A	<u>A</u>	0	1	1	0					
Entrée	Sortie															
A	<u>A</u>															
0	1															
1	0															
ET (AND)			<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><td>A</td><td>B</td></tr><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Entrée	Sortie	A	B	0	0	0	1	1	0	1	1	
Entrée	Sortie															
A	B															
0	0															
0	1															
1	0															
1	1															
ET-NON (NAND)			<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><td>A</td><td>B</td></tr><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Entrée	Sortie	A	B	0	0	0	1	1	0	1	1	
Entrée	Sortie															
A	B															
0	0															
0	1															
1	0															
1	1															
OU (OR)			<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><td>A</td><td>B</td></tr><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Entrée	Sortie	A	B	0	0	0	1	1	0	1	1	
Entrée	Sortie															
A	B															
0	0															
0	1															
1	0															
1	1															
OU-NON (NOR)			<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><td>A</td><td>B</td></tr><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Entrée	Sortie	A	B	0	0	0	1	1	0	1	1	
Entrée	Sortie															
A	B															
0	0															
0	1															
1	0															
1	1															
OU EXCLUSIF (XOR)			<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><td>A</td><td>B</td></tr><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Entrée	Sortie	A	B	0	0	0	1	1	0	1	1	
Entrée	Sortie															
A	B															
0	0															
0	1															
1	0															
1	1															
OU-NON EXCLUSIF (XNOR)			<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><td>A</td><td>B</td></tr><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Entrée	Sortie	A	B	0	0	0	1	1	0	1	1	
Entrée	Sortie															
A	B															
0	0															
0	1															
1	0															
1	1															

## 6. Algèbre de Boole

C'est un outil mathématique qui permet de manipuler des variables binaires pouvant prendre deux valeurs, 0 et 1.

Il existe deux opérateurs

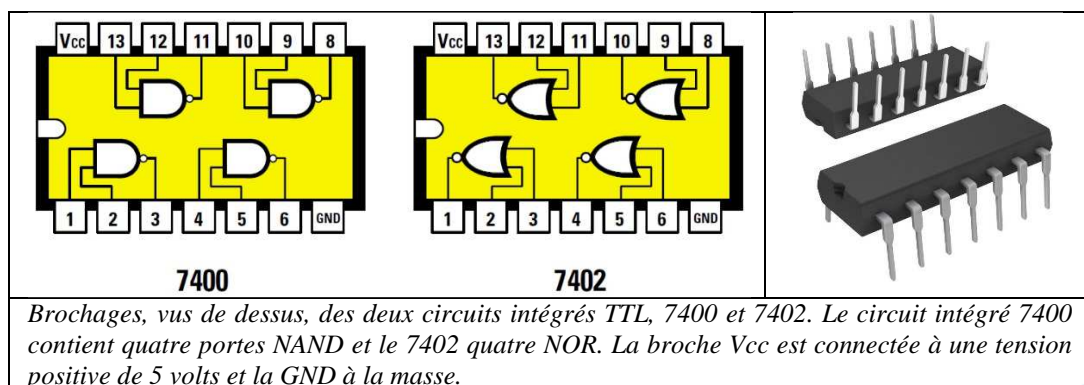
- L'opérateur « + », il faut dire « OU »
- L'opérateur « . », il faut dire « ET »
- L'opérateur ET est prioritaire sur l'opérateur OU

Somme		Produit		Involution
$0+0=0$	$a+1=a$	$0.0=0$	$a.1=a$	$\bar{0}=1$
$0+1=1$	$a+0=a$	$0.1=0$	$a.0=0$	$\bar{1}=0$
$1+0=1$	$a+a=a$	$1.0=0$	$a.a=a$	$\bar{\bar{a}}=a$
$1+1=1$	$a+\bar{a}=1$	$1.1=1$	$a.\bar{a}=0$	
Associativité		Commutativité		Distributivité
$a.(b.c)=(a.b).c$ $a+(b+c)=(a+b)+c$		$a.b=b.a$ $a+b=b+a$		$a.(b+c)=a.b+a.c$ $a+(b.c)=(a+b).(a+c)$
Absorption		Identité		De Morgan
$a+a.b=a$ $a.(a+b)=a$		$a+\bar{a}b=a+b$ $(a+b).(\bar{a}+c)=a.c+\bar{a}.b$		$\overline{a.b}=\bar{a}+\bar{b}$ $\overline{a+b}=\bar{a}.\bar{b}$

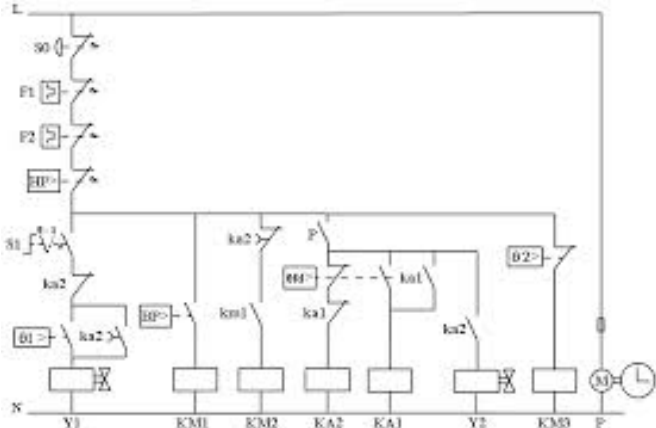
## 7. Réalisation technologiques des fonctions logiques


### 7.1. Technologie électronique

Ces composants se retrouvent dans les systèmes avec partie commande embarquée, tels que des ordinateurs, des robots nettoyeurs, des stores automatiques, des portails, etc. Ces circuits intégrés contiennent des portes logiques.



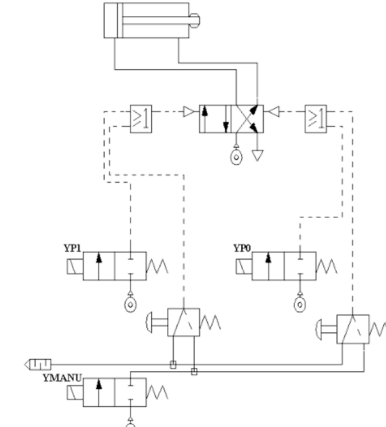
## 7.2. Technologie électrique






*Les circuits en série provoquent des fonctions logiques « ET », les circuits en dérivation provoquent des fonctions logiques « OU ».*

## 7.3. Technologie pneumatique





*Dans les circuits pneumatiques, les fonctions logiques sont garanties par des composants qui assurent mécaniquement les états désirés.*