



C9 : MODÉLISATION DE LA CHAINE D'INFORMATION DES SYSTÈMES

## C9-1 - Modélisation des systèmes logiques numériques

11 Juin 2019

---

### Table des matières

---

<b>I Introduction</b>	<b>1</b>
1 Rôle d'un système de commande . . . . .	1
2 Informations binaires et numériques . . . . .	3
3 Systèmes asservis numériques . . . . .	4
<b>II Manipulation de l'information binaire</b>	<b>5</b>
1 Bases de logiques combinatoires . . . . .	5
a) Algèbre de Boole . . . . .	5
b) Propriétés : manipulation de l'algèbre de Boole . . . . .	5
c) Théorème de De Morgan . . . . .	6
d) Définition d'une fonction par sa table de vérité . . . . .	6
e) Opérateurs logiques fondamentaux . . . . .	6
f) Logigrammes . . . . .	9
g) Chronogrammes . . . . .	9
h) Notions de fronts montants et descendants . . . . .	10
2 Numération . . . . .	10
a) Base de numération . . . . .	10
b) Conversion bases 2, 10 et 16 . . . . .	11
c) Codage ASCII . . . . .	12
<b>III Applications technologiques</b>	<b>12</b>
1 Réalisation Technologique . . . . .	12
a) Technologie électrique câblée . . . . .	12
b) Technologie électronique . . . . .	12
c) Micro-contrôleur et ordinateurs embarqués . . . . .	13
d) Codage, décodage, transcodage d'informations . . . . .	14
e) Commande de systèmes simples . . . . .	15

### Compétences

- **Analyser :**
  - Appréhender les analyses fonctionnelles et structurelles : structures de systèmes asservis
- **Modéliser :**
  - Identifier et caractériser les grandeurs physiques : flux d'information
  - Proposer un modèle de connaissance et de comportement : systèmes logiques
  - Valider un modèle : grandeurs influentes d'un modèle
- **Expérimenter :**
  - S'approprier le fonctionnement d'un système pluritechnologique
  - Proposer, justifier et mettre en oeuvre un protocole expérimental : chaîne d'acquisition, filtrage, échantillonnage, quantification
- **Conserver :** Architecture fonctionnelle et structurelle

# I. Introduction

## 1 Rôle d'un système de commande

### Définition 1 : Système numérique de contrôle-commande

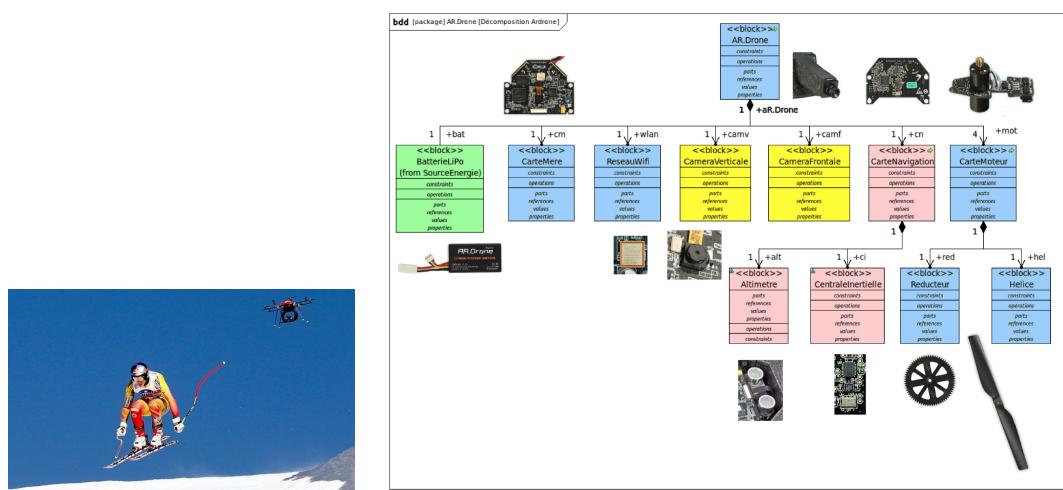
Un **système numérique de contrôle-commande** est une structure programmable (SNCC ou DCS pour *distributed control system* en Anglais) utilisée pour le pilotage d'un procédé industriel ou d'un système mécanique. Un SNCC est composé :

- d'une ou plusieurs cartes de commande qui réalisent la fonction **traiter** de la chaîne fonctionnelle;
- d'une interface homme-machine pour la supervision d'un réseau de communication numérique qui réalise la fonction **communiquer** et/ou **acquérir** de la chaîne fonctionnelle;
- d'un ensemble de capteurs permettant d'assurer une commande précise du système considéré et qui réalise la fonction **acquérir** de la chaîne fonctionnelle;

### Exemple 1 : AR Drones

L.A.R. Drone © est un robot volant quadrirotor développé par l'entreprise Française Parrot est un exemple de système technique complexe. Un exemple d'application directe de ce genre d'objet volant est son utilisation avec une caméra embarqué stabilisée pour suivre les compétitions de sports alpins comme on a pu le voir lors des derniers Jeux Olympiques d'hiver. La plus grande difficulté est la précision de la trajectoire pour assurer une bonne cadrage tout en respectant les distances de sécurité.

- L.A.R. Drone © est composé de quatre hélices entraînées par 4 moteurs électriques Brushless asservis en vitesses afin de contrôler sa trajectoire.
- L'appareil est composé d'un ordinateur avec un processeur d'1GHz.
- Différents capteurs permettent d'assurer une trajectoire avec une grande précision (Gyroscope, accéléromètre, magnétomètre, capteur de pression, capteurs à ultrasons, ...).



Un certain nombre d'informations logiques doivent être collectées et traitées sur ce genre de système (Arrêt, détection d'obstacle, etc.). Il est alors nécessaire de manipuler ces informations pour garantir le bon fonctionnement du système. L'objet de ce cours est la présentation des bases nécessaires à la manipulation des informations logiques.

## 2 Informations binaires et numériques

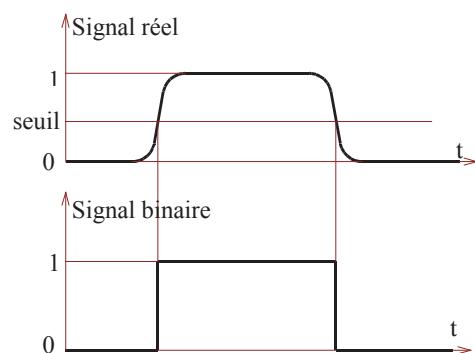


### Définition 2 : *Informations binaires*

Un signal **binnaire ou logique** est une grandeur physique qui ne peut prendre que deux états : "Vrai ou Faux".

On dit aussi qu'un signal binaire est un signal "Tout ou Rien", "présent ou absent".

Par convention, on donne le code numérique 1 au signal quand il est à l'état "vrai" et 0 quand il est à l'état "faux". En pratique, il ne peut y avoir de discontinuité au niveau d'un signal physique. Un signal, pendant le court instant de passage de vrai à faux ou de faux à vrai, prend toutes les valeurs intermédiaires. On considère alors que toute valeur du signal inférieure à un seuil donné est fausse et que toute valeur supérieure est vraie. Le passage de 0 à 1 est appelé "**front montant**" et le passage de 1 à 0 "**front descendant**". Les informations traitées par un système combinatoire sont des signaux binaires.



### Définition 3 : *Informations numériques*

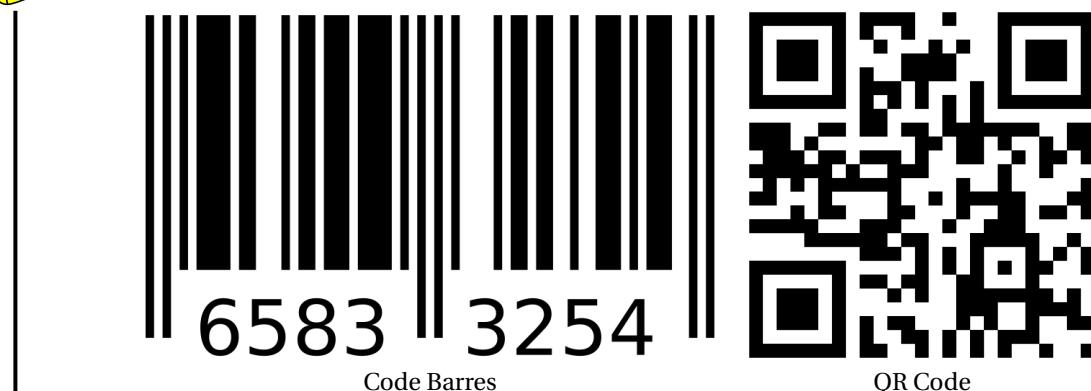
Une **information numérique** se représente à partir d'une combinaison d'**informations binaires**.

La taille d'une information numérique est directement liée au nombre d'informations binaires qui la compose.

- un *bit* ou *digit* est composé d'un nombre binaire;
- un *octet* est composé de 8 *bits*;
- un *kilo octet (ko)* est composé de 1000 *octets* et donc de 8000 *bits*.



### Exemple 2 : *Code Barres et code 2D QR*



### 3 Systèmes asservis numériques

Généralement on modélise les systèmes asservis avec les hypothèses de linéarité de comportement et de continuité des fonctions décrivant les signaux qui "transitent" dans le système. En réalité les signaux qui transitent entre les différents constituants dans la chaîne d'information sont pour la plupart numériques.



#### Définition 4 : Représentation physique et modèle numérique

La figure 1 illustre la réalité des grandeurs physiques utilisées tout au long d'une chaîne asservie (illustrée ici par l'asservissement de l'angle de roulis d'un drone). Le rôle d'une carte informatique est d'assurer les fonctions de calcul de l'écart et de la valeur de commande. Il faut donc convertir les grandeurs physiques continues en grandeurs numériques :

- dans un sens avec un **Convertisseur Analogique Numérique** : CAN;
- dans l'autre sens avec un **Convertisseur Numérique Analogique** : CNA.

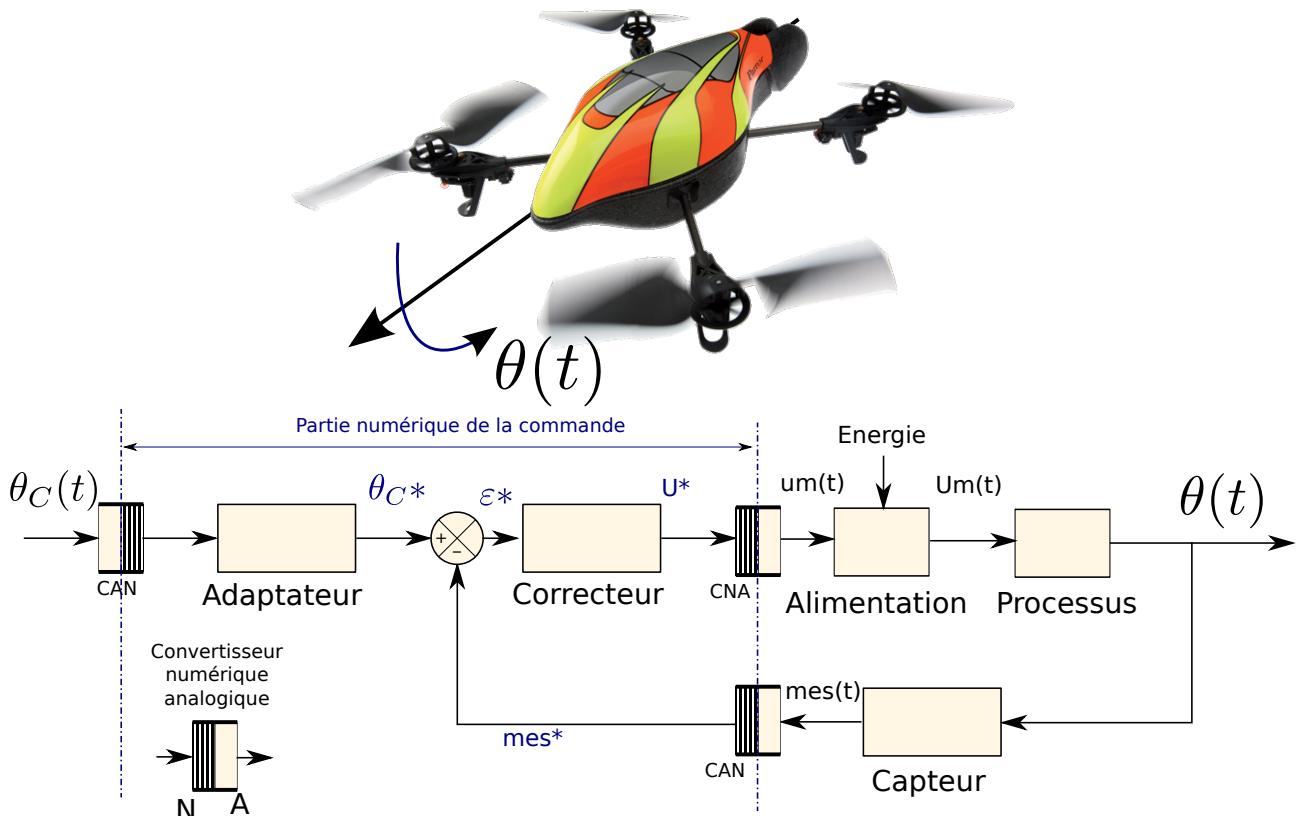


FIGURE 1 – Représentation physique et modèle numérique

## II. Manipulation de l'information binaire

### 1 Bases de logiques combinatoires

#### a) Algèbre de Boole

Les systèmes numériques ne manipulent que des grandeurs binaires logiques. Les relations entre les entrées et les sorties sont représentées par des fonctions logiques qui associent un état logique à une ou plusieurs variables logiques. L'**algèbre de Boole** est une logique mathématique permettant de formaliser ces relations.

#### Définition 5 : Algèbre de Boole

Soit un ensemble  $B\{0,1\}$  constitué de deux éléments représentants les deux états logiques vrai ou faux. Cet ensemble est muni d'une structure d'algèbre avec les trois lois suivantes :

- Complémentarité : **NON**

$$\begin{aligned} B &\longrightarrow B \\ a &\longmapsto \text{NON}(a) = \bar{a}. \end{aligned} \tag{1}$$

$\bar{a}$  est à 1 si  $a$  est à 0 et réciproquement.

- Produit booléen : **ET**

$$\begin{aligned} B \times B &\longrightarrow B \\ (a, b) &\longmapsto a \text{ ET } b = a \cdot b. \end{aligned} \tag{2}$$

$a \cdot b$  est à 1 si est seulement si  $a$  est à 1 ET  $b$  est à 1.

- Somme booléenne : **OU**

$$\begin{aligned} B \times B &\longrightarrow B \\ (a, b) &\longmapsto a \text{ OU } b = a + b. \end{aligned} \tag{3}$$

$a + b$  est à 1 si est seulement si  $a$  est à 1 OU  $b$  est à 1.

#### b) Propriétés : manipulation de l'algèbre de Boole

	Loi OU	Loi ET
Commutativité	$a + b = b + a$	$a \cdot b = b \cdot a$
Distributivité	$a + (b \cdot c) = (a + b) \cdot (a + c)$	$a \cdot (b + c) = (a \cdot b) + (a \cdot c) = a \cdot b + a \cdot c$
Associativité	$a + (b + c) = (a + b) + c = a + b + c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c = a \cdot b \cdot c$
Élément neutre 0	$a + 0 = a$	$a \cdot 0 = 0$
Élément neutre 1	$a + 1 = 1$	$a \cdot 1 = a$
Complémentarité	$a + \bar{a} = 1$	$a \cdot \bar{a} = 0$
Idempotence	$a + a = a$	$a \cdot a = a$

D'autres propriétés peuvent venir compléter ce tableau :

- Double complément :  $\bar{\bar{a}} = a$ .
- Absorption :  $a + a \cdot b = a \cdot (1 + b) = a \cdot 1 = a$ .
- Inclusion :  $a + \bar{a} \cdot b = a \cdot (1 + b) + \bar{a} \cdot b = a + a \cdot b + \bar{a} \cdot b = a + (a + \bar{a}) \cdot b = a + 1 \cdot b = a + b$ .

c) Théorème de De Morgan

 **Théorème 1 : de De Morgan**

- Le complément d'un **OU** est le **ET** des compléments :

$$\overline{(a + b)} = \bar{a} \cdot \bar{b}. \quad (4)$$

- Le complément d'un **ET** est le **OU** des compléments :

$$\overline{(a \cdot b)} = \bar{a} + \bar{b}. \quad (5)$$

d) Définition d'une fonction par sa table de vérité

 **Définition 6 : Table de vérité**

Une **table de vérité** permet de représenter le comportement logique d'une fonction de plusieurs variables logiques. Pour une fonction de  $n$  variables la table comporte :

- $n$  colonnes plus une pour le résultat.
- $2^n$  lignes pour les différentes combinaisons des  $n$  variables écrites en **binaire naturel** ou **binaire réfléchi** à partir de 0.

Ci-contre les tables de vérité, en **binaire naturel** ou **binaire réfléchi**, définissant une fonction S de trois variables :  $S(a, b, c)$ .

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Code binaire naturel

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
0	0	0	0
0	0	1	1
0	1	1	1
0	1	0	1
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	0

Code binaire réfléchi

e) Opérateurs logiques fondamentaux

Les deux tableaux suivants donnent les représentations des différentes opérations logiques fondamentales. Pour chacune d'elle, on donne la table de vérité ainsi que le symbole permettant de les utiliser dans des logigrammes.

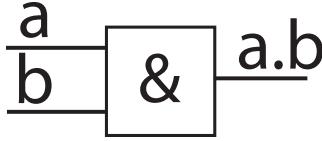
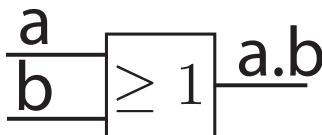
Opérateur logique NON																	
Informations	Table de vérité	Symbolisation															
C'est l'opérateur qui transforme une variable logique $a$ en son complément $\bar{a}$ (a barre).	<table border="1"> <thead> <tr> <th></th> <th><math>a</math></th> <th><math>\bar{a}</math></th> </tr> </thead> <tbody> <tr> <td>0</td><td>1</td><td></td></tr> <tr> <td>1</td><td>0</td><td></td></tr> </tbody> </table>		$a$	$\bar{a}$	0	1		1	0								
	$a$	$\bar{a}$															
0	1																
1	0																
Opérateur logique ET																	
Le résultat R de l'opération "ET" entre deux variables binaires $a$ et $b$ est égal à 1 si et seulement si les deux variables sont vraies. On note $a \cdot b = R$ . On lit "a ET b égale R"	<table border="1"> <thead> <tr> <th><math>a</math></th> <th><math>b</math></th> <th><math>a \cdot b</math></th> </tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</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>1</td></tr> </tbody> </table>	$a$	$b$	$a \cdot b$	0	0	0	0	1	0	1	0	0	1	1	1	
$a$	$b$	$a \cdot b$															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
Opérateur logique OU																	
Le résultat R de l'opération "OU" entre deux variables binaires $a$ et $b$ est égal à 1 si au moins une des deux variables est vraie. On note $a + b = R$ . On lit "a OU b égale R"	<table border="1"> <thead> <tr> <th><math>a</math></th> <th><math>b</math></th> <th><math>a + b</math></th> </tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</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>1</td></tr> </tbody> </table>	$a$	$b$	$a + b$	0	0	0	0	1	1	1	0	1	1	1	1	
$a$	$b$	$a + b$															
0	0	0															
0	1	1															
1	0	1															
1	1	1															

TABLE 1 – Opérateurs logiques fondamentaux

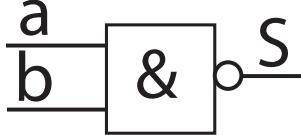
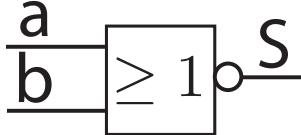
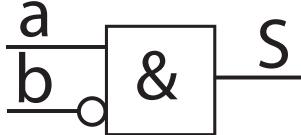
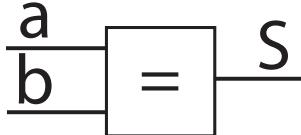
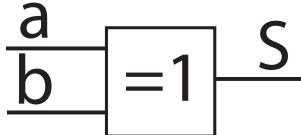
Opérateur NON-ET (ou NAND)																		
Règle	Table de vérité		Symbolisation															
La sortie est à 1 si au moins une des entrées est à 0. $S = \overline{a \cdot b} = \overline{a} + \overline{b}$		<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>S</th></tr> </thead> <tbody> <tr><td>0</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> </tbody> </table>	a	b	S	0	0	1	0	1	1	1	0	1	1	1	0	
a	b	S																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
Opérateur NON-OU (ou NOR)																		
Règle	Table de vérité		Symbolisation															
La sortie est à 1 si toutes les entrées sont à 0. $S = \overline{a + b} = \overline{a} \cdot \overline{b}$		<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>S</th></tr> </thead> <tbody> <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> </tbody> </table>	a	b	S	0	0	1	0	1	0	1	0	0	1	1	0	
a	b	S																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Opérateur INHIBITION																		
Règle	Table de vérité		Symbolisation															
La sortie est à 1 si l'entrée inhibition est à 0 et les autres à 1. $S = a \cdot \overline{b}$		<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	a	b	S	0	0	0	0	1	0	1	0	1	1	1	0	
a	b	S																
0	0	0																
0	1	0																
1	0	1																
1	1	0																
Opérateur IDENTITÉ																		
Règle	Table de vérité		Symbolisation															
La sortie est à 1 si toutes les entrées sont au même état. $S = a \otimes b = a \cdot b + \overline{a} \cdot \overline{b}$		<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>S</th></tr> </thead> <tbody> <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>1</td></tr> </tbody> </table>	a	b	S	0	0	1	0	1	0	1	0	0	1	1	1	
a	b	S																
0	0	1																
0	1	0																
1	0	0																
1	1	1																
Opérateur OU EXCLUSIF																		
Règle	Table de vérité		Symbolisation															
La sortie est à 1 si une et une seule des entrées est à 1. $S = a \oplus b = a \cdot \overline{b} + \overline{a} \cdot b$		<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</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> </tbody> </table>	a	b	S	0	0	0	0	1	1	1	0	1	1	1	0	
a	b	S																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

TABLE 2 – Fonctions logiques de base

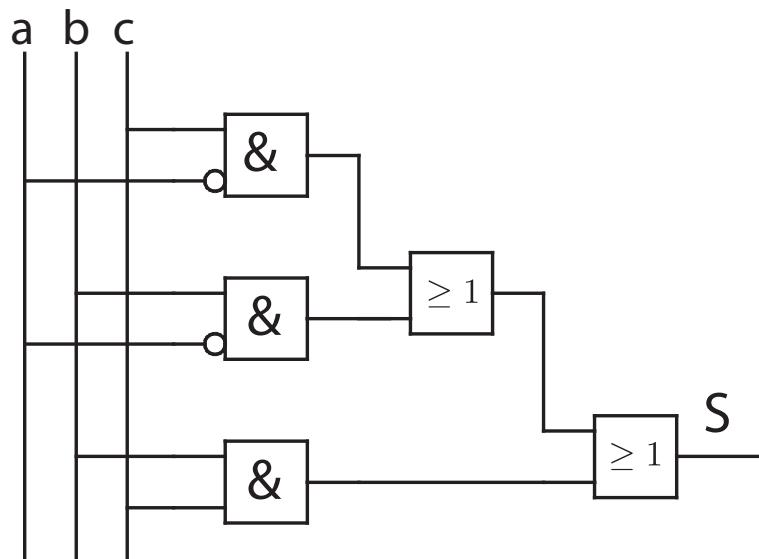
**Remarque 1 :**

- $\overline{a \oplus b} = a \otimes b$ ;
- $a \otimes b = a \oplus b$ ;
- $a \oplus b \oplus c$  ne veut pas dire une seule variable à 1, il faut lire  $a \oplus (b \oplus c)$  ou  $(a \oplus b) \oplus c$ .

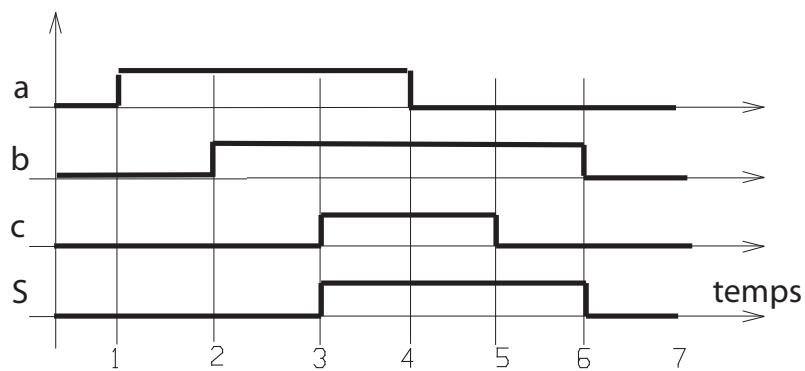
**f) Logigrammes****Définition 7 : Logigramme**

| Les **logigrammes** utilisent les symboles des opérateurs et des fonctions de base.

Ci-contre le logigramme de la fonction :

**Exemple 3 : Exemple de logigramme pour  $S = \overline{a} \cdot c + \overline{a} \cdot b + b \cdot c$** **g) Chronogrammes****Définition 8 : Chronogramme**

| Le **chronogramme** est un graphe qui permet de définir l'état d'une variable de sortie à partir de l'état des variables d'entrée définies chronologiquement.

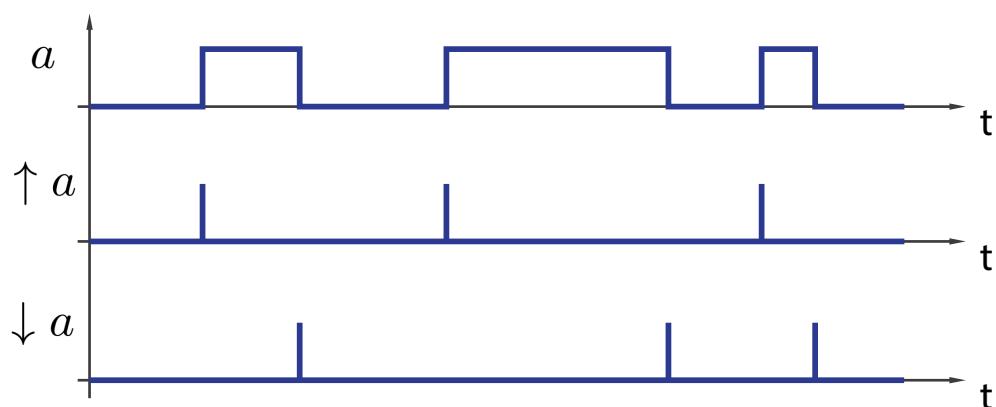

**Exemple 4 : Exemple de chronogramme pour  $S = \bar{a} \cdot c + \bar{a} \cdot b + b \cdot c$** 

**Remarque 2 :**

deux variables ne peuvent pas changer d'état **simultanément**.

### h) Notions de fronts montants et descendants


**Définition 9 : Fronts montants et descendants**

- Le **front montant** d'une variable  $a$  est vrai à l'instant pour lequel la variable passe de son état logique 0 à son état logique 1. Il est noté  $\uparrow a$ .
- Le **front descendant** d'une variable  $a$  est vrai à l'instant pour lequel la variable passe de son état logique 1 à son état logique 0. Il est noté  $\downarrow a$ .
- Un front est d'une durée théorique nulle. Il sera représenté par une impulsion de Dirac sur un chronogramme.



## 2 Numération

### a) Base de numération

Nous avons l'habitude de manipuler des valeurs numériques en base 10. Les microprocesseurs et les ordinateurs travaillent en base 2. Cette base ne comporte que deux éléments : 0 et 1. Pour l'ordinateur ces deux symboles, ou "bit", correspondent respectivement à l'absence et à la présence d'un courant électrique.

**Définition 10 : Écriture sur une base N**

L'écriture d'un nombre  $A$  dans une base  $N$  est :

$$A = \sum_{i=0}^k c_i N^i, \quad (6)$$

où les  $c_i$  sont des éléments de la base  $N$ .

**Exemple 5 :**

$$| \quad 12345 = 1 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

**Remarque 3 :**

Pour reconnaître dans quelle base est écrit un nombre, on note en indice la base ( $12345_{10}$ ). La notation en langage C utilise un préfixe (0b ou 0x) pour préciser la base avant d'indiquer le nombre. Par exemple :

- 0b0111010 pour un nombre binaire,
- 0xF3 pour un nombre en hexadécimal.

La base décimal n'a pas de notation particulière puisqu'il s'agit de la base par défaut.

**b) Conversion bases 2, 10 et 16**

La table suivante donne la conversion entre les bases 2, 10 et 16 :

Base 10	Base 2	Base 16
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Pour convertir un nombre on peut utiliser des divisions euclidiennes ou des factorisations successives Conversion d'un nombre décimal en binaire par la méthode des divisions successives.



### Exemple 6 : conversion du nombre 47

$$(47)_{10} = (?)_2$$

- Par **factorisations successives** par 2 :

$$47 = 23 \times 2 + 1$$

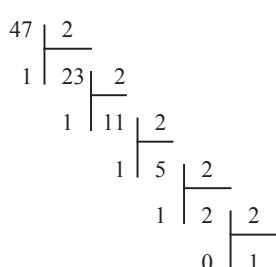
$$47 = ((11 \times 2 + 1) \times 2 + 1)$$

$$47 = (((5 \times 2 + 1) \times 2 + 1) \times 2 + 1)$$

$$47 = (((((2 \times 2 + 1) \times 2 + 1) \times 2 + 1) \times 2 + 1) \times 2 + 1)$$

$$47 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0.$$

- Par série de **divisions euclidiennes** :



On obtient :

$$(47)_{10} = (101111)_2$$

### c) Codage ASCII



#### Définition 11 : Code ASCII

Le **code ASCII** (American Standard Code for Information Interchange) est un code défini sur 8 bits permettant de définir 256 caractères distincts. Chaque caractère possède un nombre associé en hexadécimal.

## III. Applications technologiques

### 1 Réalisation Technologique

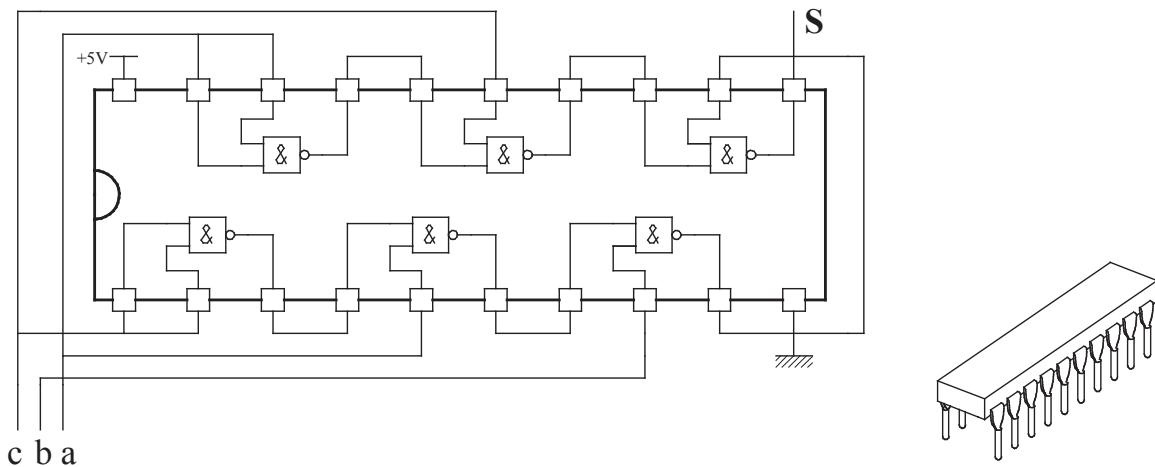
#### a) Technologie électrique câblée

- En logique électrique l'élément technologique de base est le contact à établissement de circuit (contact NO : normalement ouvert).
- La variable complémentée est matérialisée par le contact à interruption de circuit (contact NF : normalement fermé).
- L'opérateur ET est obtenu par la mise en série (câblage) des contacts, l'opérateur OU par la mise en parallèle. (*Voir l'exemple du "va-et-vient" ci-dessous.*)

#### b) Technologie électronique

Cette technologie de commande est réservée à des produits fabriqués en grande série. Les composants sont soudés sur un circuit imprimé. La figure ci-contre représente un composant regroupant 6 opérateurs NON-ET à deux entrées.

Reprenons la fonction :  $S = \bar{a} \cdot c + \bar{a} \cdot b + b \cdot c$  et cherchons à l'implanter avec ce composant. Il faut pour cela la mettre sous la forme de produits complémentés de deux termes.



$$S = \bar{a} \cdot c + \bar{a} \cdot b + b \cdot c$$

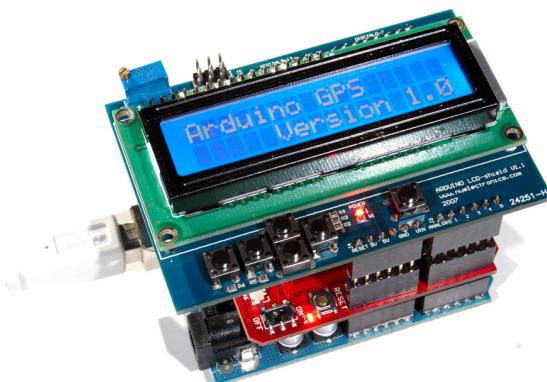
$$S = \bar{a} \cdot c + b \cdot (\bar{a} + c)$$

$$S = \bar{a} \cdot c + b \cdot \overline{a \cdot \bar{c}}$$

$$S = \overline{\overline{\bar{a} \cdot c} \cdot \overline{b \cdot \overline{a \cdot \bar{c}}}}$$

### c) Micro-contrôleur et ordinateurs embarqués

Les micro-contrôleurs sont constitués de portes logiques et de mémoires mais à la différence des FPGA, leur structure interne se rapproche de celle des ordinateurs avec un microprocesseur. De plus ils comportent des périphériques permettant des fonctionnalités avancées (communication série RS232, USB, Ethernet, etc...). Une application courante est la commande des hacheurs (variateur électronique de puissance dédiés aux moteurs électriques). Ils se rapprochent des ordinateurs à la différence près que leur performances sont optimisées en fonction de leur utilité dans un système (moins de mémoire et de puissance de calcul). La plupart des systèmes intelligents en possèdent un ou plusieurs (voitures modernes, téléphones portables, AR drones, etc...).

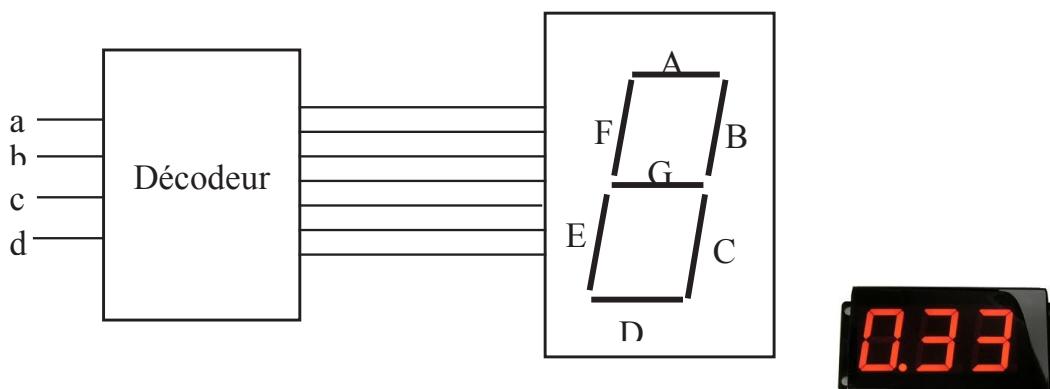


**d) Codage, décodage, transcodage d'informations**



**Exemple 7 : Commande d'un afficheur 7 segments**

Un afficheur permet d'inscrire un chiffre décimal par l'allumage d'une combinaison de diodes électroluminescentes ou de cristaux liquides. Les variables d'entrées représentent le chiffre décimal codé en DCB (décimal codé binaire).



**Exemple 8 : Construction de la table de vérité de chaque segment**

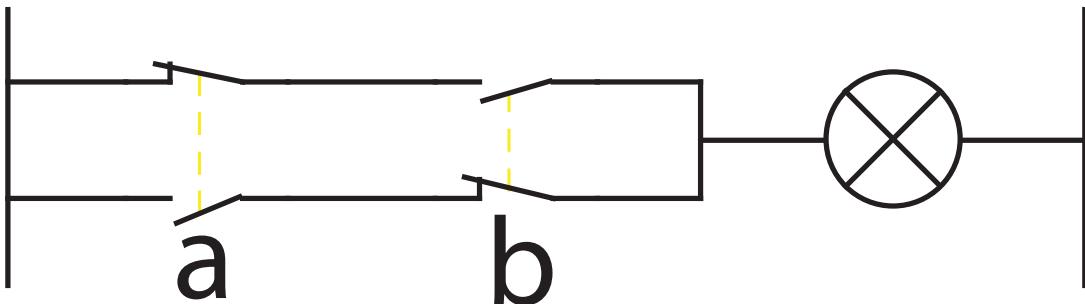
	d	c	b	a	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

e) **Commande de systèmes simples**



**Exemple 9 :**

Commande d'une lampe par "va-et-vient". Deux interrupteurs bistables  $a$  et  $b$  (deux positions stables) sont situés respectivement aux deux entrées d'une pièce. On doit pouvoir allumer ou éteindre la lampe à l'une quelconque des deux entrées.



Les deux contacts reliés par un pointillé basculent simultanément.

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

$$S = \bar{a} \cdot b + a \cdot \bar{b} = a \oplus b.$$