MPSI – PCSI

Sciences Industrielles de l'Ingénieur

Séquence 10

Systèmes à événements discrets

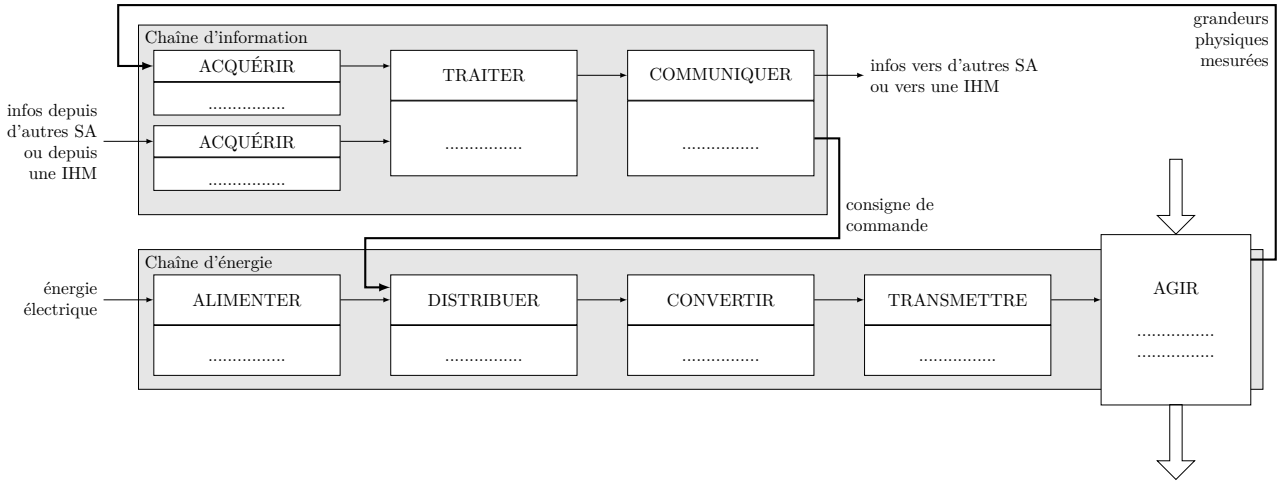
Nicolas COURRIER – Jérôme DIDIER – Sébastien GERGADIER

A	Analyser
A1	Identifier le besoin et les exigences
A1-01	Décrire le besoin
A1-02	Traduire un besoin fonctionnel en exigences
A1-03	Présenter la fonction globale
A1-04	Définir les domaines d'application, les critères technico-économiques
A1-05	Identifier les contraintes
A1-06	Identifier et caractériser les fonctions
A1-07	Qualifier et quantifier les exigences (critère, niveau)
A1-08	Évaluer l'impact environnemental (matériaux, énergies, nuisances)
A3	Appréhender les analyses fonctionnelle et structurelle
A3-01	Analyser les architectures fonctionnelle et structurelle
A3-02	Identifier les fonctions des différents constituants
A3-03	Repérer les constituants dédiés aux fonctions d'un système
A3-04	Identifier la structure d'un système asservi: chaîne directe, capteur, commande, consigne, comparateur, correcteur
A3-05	Identifier et positionner les perturbations
A3-06	Différentier régulation et poursuite
A3-08	Identifier et décrire la chaîne d'information et la chaine d'énergie du système
A3-09	Identifier les liens entre la chaîne d'énergie et la chaine d'information
A3-10	Identifier les constituants de la chaîne d'information réalisant les fonctions acquérir, coder, communiquer, mémoriser, restituer, traiter
A3-11	Identifier les constituants de la chaîne d'énergie réalisant les fonctions agir, alimenter, convertir, moduler, transmettre, stocker

Table des matières

Outre le dimensionnement du système en termes de performances (conception de la chaîne d'énergie), l'ingénieur doit de plus en plus concevoir la chaîne d'information qui aujourd'hui prend une part très importante dans les systèmes rencontrés.

Les capteurs acquièrent des grandeurs physiques qui sont ensuite traitées par la partie commande qui va ensuite communiquer des ordres à la chaîne d'énergie ou envoyer des informations à des éléments extérieurs.



Objectif

Les objectifs du cours sont :

- *de découvrir l'architecture des systèmes de commande actuels ;*
- *de manipuler l'information qui transite dans la chaîne d'information ;*
- *d'analyser voire de décrire le fonctionnement d'un système.*

I Architecture des systèmes de commande

I.1 Rôle d'un système de commande

Depuis le début des années 2000, les progrès technologiques sans cesse croissants des actionneurs (moteurs, vérins, etc.), des protocoles de communications numériques et des capteurs dits « intelligents » ont facilité le développement de *systèmes numériques de contrôle-commande* (SNCC) de plus en plus puissants.

> Un système numérique de contrôle-commande (SNCC, ou DCS pour *distributed control system* en anglais) est une structure programmable utilisée pour le pilotage d'un procédé industriel. Un SNCC est doté d'une interface homme-machine pour la supervision et d'un réseau de communication numérique pour l'interface avec les différents éléments de son environnement tels que, par exemple, les capteurs ou les autres cartes de commande associées à des sous-systèmes.

I.2 Architecture

Les SNCC sont architecturés autour d'une carte de commande principale appelée « carte mère » à laquelle sont connectées plusieurs cartes secondaires, appelées « cartes filles », gérant de manière efficace un nombre limité de tâches, par exemple l'asservissement en vitesse et position de l'arbre de sortie d'un moteur électrique.

Cette organisation nodulaire est simple à concevoir et la division du traitement des informations sur plusieurs cartes présente de nombreux avantages parmi lesquels la fiabilité, les

capacités de traitement et de stockage des données acquises par les capteurs ou transmises par les autres cartes, le faible temps de réponse à un événement d'entrée et la possibilité de gestion des communications par la carte mère comme par les cartes filles.

Exemple : Le robot à structure humanoïde NAO®, développé par l'entreprise française Aldebaran Robotics, est un exemple d'un système technique complexe. Pour gérer à la fois les fonctions de détection de l'environnement, de déplacement fluide et de préhension d'objets, le robot dispose de nombreux capteurs (sonores, visuels et tactiles), de 25 axes asservis en position et en vitesse de rotation et d'un calculateur central gérant l'ensemble des fonctions.

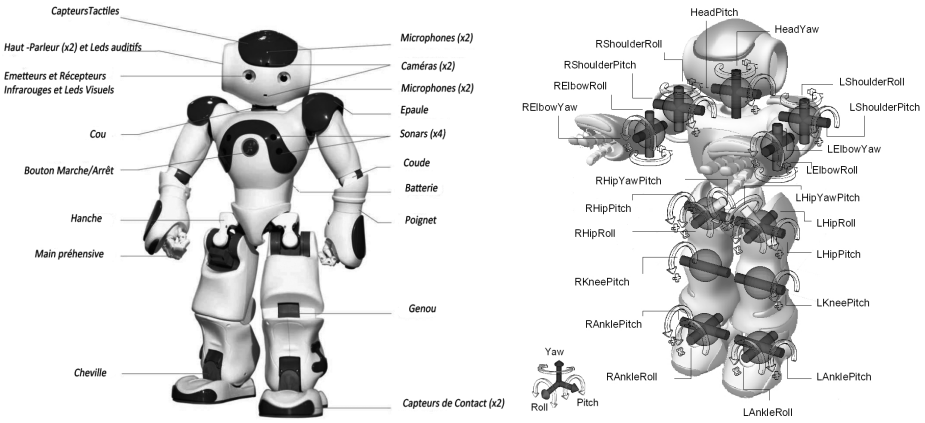


FIGURE *1 : Le robot NAO® de l'entreprise ALDEBARAN ROBOTICS.*

Lors, par exemple, du déplacement en ligne droite de ce robot humanoïde, au moins 16 axes sont sollicités (les 6 de chaque jambe + les 2 de chaque épaule afin d'assurer l'équilibre) : il est donc impossible que chacun de ces axes ait un comportement individuel non contrôlé globalement par un calculateur central. Pour améliorer les capacités du robot et gérer efficacement les flux d'informations, les 25 axes ne peuvent être gérés par un seul et même calculateur : l'utilisation d'un SNCC est donc indispensable.
Dans le cas du robot NAO®, le SNCC est organisé autour d'une carte mère à processeur Intel ATOM® et embarquant un système d'exploitation Linux (noyau 2.6 temps réel). Chacun des 25 axes est géré par une carte microcontrôleur dédiée et connectée au calculateur central par un bus de données. La programmation du robot, connecté à un ordinateur par une connexion WiFi ou Ethernet, se fait par une interface graphique reproduisant la structure d'un diagramme d'activités (voir cours SysML). Le paramétrage des blocs de cette interface se fait en langage Python. La création des blocs et de nouvelles activités se fait dans de nombreux langages parmi lesquels C, C++, Python, Java et Matlab.

La réalisation d'un système de commande s'appuie sur des technologies numériques qui évoluent très rapidement. Les performances s'améliorent chaque jour et autorisent des applications toujours plus pointues.

La commande numérique est fondamentalement constituée de portes logiques et de mémoires, qui manipulent de l'information binaire. Cette information est matérialisée électriquement, le plus souvent par un potentiel nul (0 logique) et un potentiel de quelques volts (1 logique).

I.2.1 Structure d'un FPGA

Un FPGA (*Field Programmable Gate Arrays*) est un « réseau logique programmable », c'est-à-dire une matrice de cellules logiques reliées entre elles par un réseau matriciel de connections dont les nœuds peuvent être connectés de façon programmée (figure 2). Il existe bien entendu d'autres composants de bas niveaux mais le FPGA est une solution d'avenir relativement représentative dans sa structure et sa flexibilité en termes de conception.

L'élément de base d'un FPGA est la *cellule logique* programmable (CL), capable d'implémenter des opérations logiques élémentaires : ET, OU, NON et leurs variantes. Cette cellule logique est essentiellement composée d'une table de calcul logique réalisant l'opération et d'une bascule permettant la mémorisation d'un bit. Les FPGA contiennent plusieurs centaines de milliers de cellules logiques.

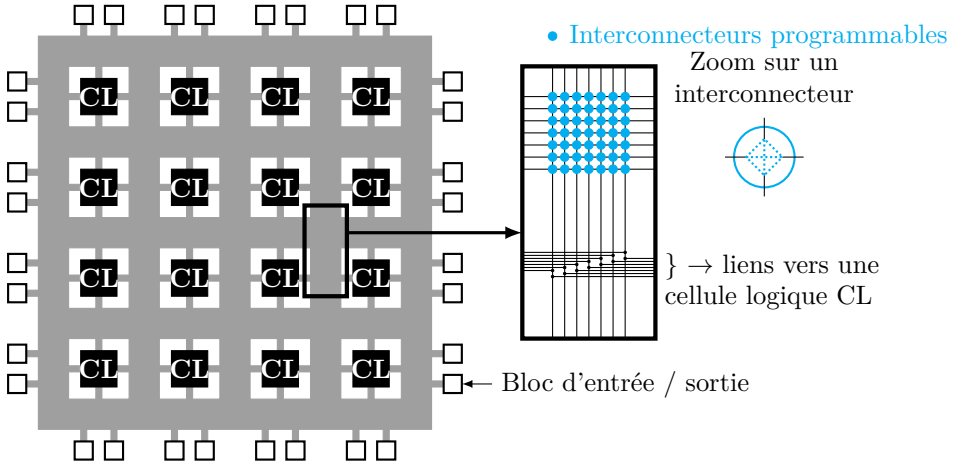


FIGURE 2 : Architecture interne d'un FPGA.

La matrice de routage connecte les cellules logiques entre elles et aux broches d'entrées-sorties. À chaque croisement de connecteurs, le signal peut être transmis dans toutes les directions de façon programmée. Il est ainsi possible de réaliser ces circuits logiques très complexes dans un seul composant, autorisant des vitesses d'exécutions exceptionnelles tout en consommant peu d'énergie.

Les FPGA modernes comportent, de plus, des composants périphériques comme de la mémoire, des horloges, des modules de communication série, des convertisseurs analogique-numérique ou numérique-analogique, ou encore des architectures optimisées pour accueillir une ou plusieurs structures de micro-contrôleurs dans la matrice du FPGA.

I.2.2 Structure d'un micro-contrôleur

Le cœur d'un micro-contrôleur est lui aussi constitué de portes logiques et de mémoires. À la différence du FPGA, la structure interne est déjà organisée pour réaliser un micro-processeur et ne peut pas être reconfigurée. Le micro-contrôleur comporte (figure 3) de la mémoire sous forme de registres et des zones mémoire dédiées aux programmes et aux données. L'utilisateur configure les fonctions du micro-contrôleur en modifiant les registres et fixe les tâches à effectuer en implantant un programme dans la mémoire « non volatile », c'est-à-dire qu'elle ne s'efface pas lorsque le composant n'est pas alimenté. Les variables utilisées au cours de l'exécution du programme sont quant à elles stockées dans la mémoire vive « volatile » du micro-contrôleur. Un bus de données permet aux informations binaires de circuler entre les mémoires, le processeur et les périphériques d'entrée-sortie.

Par ailleurs, le micro-contrôleur comporte généralement des périphériques permettant de gérer des fonctionnalités avancées telles que la communication série suivant divers protocoles (communication série RS232, USB, Ethernet, etc.), la conversion analogique-numérique, les sorties PWM utilisées pour la commande de hacheurs, etc.

Le micro-contrôleur se rapproche donc du fonctionnement de la carte mère complète d'un ordinateur en un seul composant. Par comparaison avec un ordinateur de bureau, la mémoire, la vitesse d'exécution et les capacités de calcul sont nettement inférieures mais c'est généralement largement suffisant pour des applications embarquées. Sans le savoir, chacun de nous possède

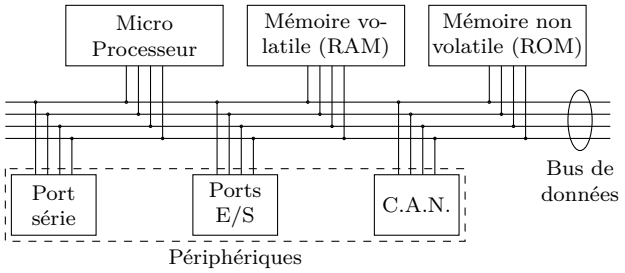


FIGURE 3 : Architecture interne d'un micro-contrôleur.

plusieurs dizaines (voire centaines) de micro-contrôleurs : une voiture en comporte déjà plus de cinquante (pour le contrôle de l'ABS, des injecteurs, des radars de recul, de la climatisation, de la fermeture centralisée, etc.), un ordinateur en comporte plus de vingt (contrôle de l'alimentation, de la ventilation, du lecteur CD, du clavier, de la souris, etc.) et la plupart des composants intelligents en comportent un ou plusieurs (télévision, appareil photo, téléphone, télécommande, jouet, box internet, ampoule basse consommation, etc.).

I.2.3 Les processeurs de signaux digitaux (DSP)

Alors qu'un micro-contrôleur est plutôt généraliste et d'architecture « standard », les DSP (*Digital Signal Processors*) sont des micro-contrôleurs dont l'architecture interne est optimisée pour des tâches spécifiques, comme le calcul de transformées de Fourier, le traitement d'images ou de vidéos, les protocoles de communication série de forts débits, le contrôle de moteurs brushless, etc.

I.2.4 Vers les ordinateurs embarqués et l'architecture distribuée

Les performances des micro-contrôleurs sont très variables, depuis les plus petits, peu coûteux, très sobres et dédiés à une tâche précise, jusqu'aux plus gros, disposant d'un espace mémoire confortable, de périphériques élaborés (USB, VGA, Ethernet, etc.), et pourvus d'une puissance de calcul significative (multiplication, données codées sur 32 bits, calcul flottant, etc). Sur ces derniers, il est souvent possible d'implanter un OS (*Operating System*) comme Linux.

Le micro-contrôleur propose alors la plupart des fonctionnalités de haut niveau proposées par un ordinateur (multi-tâches, gestion des fichiers, drivers pour le pilotage de matériel, affichage sur un écran, gestion du clavier et de la souris, communication TCP-IP, etc.) tout en restant très simples à programmer à l'aide de bibliothèques standards.

Cette solution présente néanmoins des inconvénients : la gestion de l'OS absorbe une part de la capacité de calcul et conduit à une plus forte consommation, l'aspect multi-tâches rend les programmes vulnérables à des retards d'exécution lorsque le processeur est occupé à d'autres tâches et les temps de réponse sont généralement beaucoup plus longs.

Un système de commande complexe présente souvent une *architecture distribuée* où plusieurs micro-contrôleurs sont reliés entre eux par des bus de données. Les plus puissants embarquent un OS, commandent le fonctionnement global du système, et transmettent leurs consignes aux plus petits, responsables d'une tâche précise comme le contrôle d'un actionneur par exemple, et susceptibles de renvoyer des informations comme des données capteurs.

I.3 Nature des informations

Les systèmes numériques ne manipulent que des grandeurs binaires logiques donc de type « tout ou rien » : les entrées et les sorties de ses composants ne peuvent prendre que deux états.

Une succession de données binaires sera nommée grandeur numérique (ou discrète) par opposition à une grandeur analogique ou continue.
Les capteurs sont toujours utilisées pour mesurer des grandeurs physiques continues (pression, température, position,...). Ces capteurs délivrent ensuite une grandeur qui doit être échantillonnée puis quantifiée pour être utilisée par un système de commande numérique.
En général, on convertit les signaux de nature non-électrique en signaux électriques, et vice-versa, à l'aide d'un transducteur comme un microphone, un haut-parleur, une caméra numérique, un écran d'ordinateur, une antenne, un thermomètre électronique, un indicateur de débit ou un accéléromètre. Ensuite, on mesure l'intensité électrique correspondant au phénomène ou à la quantité physique pour en indiquer la valeur à un moment précis. Finalement, on convertit cette intensité électrique en un nombre pouvant être traité par le système numérique. Ce processus s'appelle la numérisation d'un signal.

II Traitement des informations binaires

Pour représenter des valeurs chiffrées, il est nécessaire de disposer d'une base. Le système de numérotation le plus utilisé aujourd'hui est le système décimal. Les systèmes numériques utilisent plus naturellement le système binaire. La connaissance du système binaire et des changements de base binaire vers décimal et décimal vers binaire est donc essentielle à l'étude et à la réalisation des parties commandes.

II.1 Système de numération

II.1.1 Bases

Un nombre est représenté par la juxtaposition de symboles appelés digits pris parmi les éléments de la base B considérée.
Les bases les plus couramment utilisées sont :

- la base 10 (ou décimale) contenant 10 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- la base 2 (ou binaire) contenant 2 symboles : 0, 1
- la base 16 (ou hexadécimale) comptant 16 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

La figure ci-contre représente l'écriture des nombres de 0 à 15 dans 3 bases différentes.
Exemple : $(10)_{10}$ (lire dix) représente le nombre 10 en base 10, $(10)_2$ (ou 0b10, lire un, zéro) représente le nombre 10 écrit en base 2 et $(10)_{16}$ (ou 0x10) représente le nombre 10 en base 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

II.1.2 Changement de bases : transcodage

L'objectif est de mettre en place la correspondance entre l'expression d'un nombre dans une base (ex : base 10) et son expression dans une autre base (ex : base 2).

Passage d'une base X vers la base décimale Un nombre $(\text{abcd})_X$ (avec a, b, c, d 4 digits) dans une base X correspond au nombre suivant en base décimale : $(\text{abcd})_X = a.X^3 + b.X^2 + c.X^1 + d.X^0$.

On parle de système de numération pondéré. On appelle poids le rang de chaque digit (exemple : a digit de poids le plus fort)

Exemple :
Binaire vers décimal : $(1011)_2$:
Hexadécimal vers décimal : $(A3B)_{16}$:

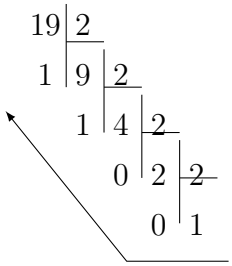
Passage de la base décimale vers une base X On utilise la méthode des divisions successives. Soit un nombre N défini sur une base X. Son écriture en base 10 est la suivante : $(N)_X = n_p.X^p + n_{p-1}X^{p-1} + \ldots + n_1.X^1 + n_0.X^0$. Effectuer le changement de base revient à déterminer n_0, n_1, $\ldots$, n_{p-1}, n_p

En mettant X en facteur il vient : $(N)_X = X(np.X^{p-1} + n_{p-1}X^{p-2} + \ldots + n_1.X^0) + n_0$ qui est de la forme : $(N)_X = X \times (quotient) + reste$. Le reste permet donc de déterminer n_0.

En reprenant le quotient précédent et en factorisant par X on trouve un nouveau reste n_1 et ainsi de suite jusqu'à n_p.

Ainsi pour écrire un nombre décimal en binaire, il faut faire des divisions successives par 2. Par exemple pour écrire 19 en binaire, on réalise les divisions ci-contre :

Le sens de lecture, et donc d'écriture, se fait selon la flèche. Le nombre décimal 19 s'écrit ainsi en binaire 10011, soit : $(19)_{10} = (10011)_2$.



Exemple : Écrire 25 en binaire et en hexadécimal.

II.2 Codage de l'information

Les ordinateurs travaillant en binaire, tout traitement informatique ou automatique nécessite de coder l'information en binaire, à l'aide des deux états 1 ou 0. Un bit (contraction de binary digit) est un chiffre binaire (1 ou 0). Un Octet (byte en anglais) correspond à 8 bits. Un ko (kilo octet) correspond à 2^{10} chiffres soit 1024 octets.

L'écriture des nombres dans cette base donnant une multitude de chiffres, les informaticiens lui préfèrent le système hexadécimal qui permet de contracter les informations en paquets de 4 bits.

Il est également intéressant de regrouper un ensemble de valeurs binaires suivant une autre organisation qu'un système de nombres. Ces autres organisations sont appelées des **codes**. Il en existe un très grand nombre et chacun peut en créer selon un besoin spécifique.

Les qualités requises pour un code sont principalement :
- la taille du codage (nombre de bits nécessaires)
- la fiabilité de lecture
- la simplicité de manipulation

Exemple : Les données stockées dans une carte vitale (nom, âge ...) sont codées selon une norme ASCII (American Standard Code For Information Interchange) qui nécessite 8 bits et permet de coder 256 caractères. Les chiffres de 0 à 9 en ASCII binaire sont obtenus en ajoutant 0011 devant le code binaire du chiffre sur 4 bits. Les lettres de A à Z sont codées en ajoutant 010 devant le code binaire des 26 lettres (A=01000001, etc...)

Exemple : Écrire la séquence suivante en ASCII binaire : 19ans. Combien d'octets sont nécessaires pour coder cette séquence ?

On étudie dans la suite les principaux codes à connaître. On distingue les codes pondérés pour lesquels chaque chiffre possède un poids, des codes non pondérés qui nécessite une table de correspondance pour décoder un nombre.

II.2.1 Code binaire naturel

Ce code pondéré correspond à donner un nombre selon sa valeur en système de numérotation binaire.

C'est le seul code qui permet de réaliser des opérations (additions, soustractions, ...) très simplement.

Inconvénient : il introduit des erreurs lors d'un changement de code. Pour passer de $(01)_2$ à $(10)_2$ (donc de 1 à 2), il faut modifier les deux digits et afficher (même brièvement) $(11)_2$ (soit 3) ou $(00)_2$ (soit 0), ce qui provoque des perturbations, ou erreurs.

0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

II.2.2 Code décimal codé binaire (Code DCB)

C'est un code pondéré basé sur le code binaire naturel, mais qui est adapté à la représentation des nombres en base 10. En effet le code binaire pur n'associe pas de bits spécifiques aux unités, dizaines, centaines,.... La propriété du code DCB est d'associer quatre bits différents à chaque puissance de 10. Ainsi 1664(DCB) s'écrit 0001.0110.0110.0100 (voir tableau code binaire pur).

Ce code est utilisé sur les afficheurs 7 segments. Chaque afficheur reçoit le chiffre codé en binaire sur 4 bits.

Inconvénient : Cette représentation adaptée à la représentation binaire des nombres décimaux utilise un nombre de bits supérieur à celui du binaire naturel, et donc une place plus importante en mémoire de l'ordinateur.

Remarque : on peut réaliser la même typologie pour le code hexadécimal. Ainsi $(0C3F)_{16}$ s'écrit 0000.1100.0011.1111 (cette fois la représentation est optimale en mémoire).

II.2.3 Code binaire réfléchi ou code Gray

Ce code non pondéré est un arrangement du système binaire. Le passage d'un nombre à l'autre se fait en changeant l'état d'un seul bit, et celui qui change est le bit le plus à droite ne provoquant pas un nombre déjà écrit. Il est très utilisé pour décrire des automatismes (un changement d'état d'un composant correspond à un bit qui change), en particulier dans les codeurs de position absolue (FIGURE 4.

Avantage : Il apporte une garantie d'interprétation avec une erreur maximale d'une incrémentation. Inconvénient : ils sont très chers car plus difficile à réaliser que leurs homologues les codeurs incrémentaux

Pour obtenir le code Gray, il faut faire toutes les deux 2^1, quatre 2^2, huit 2^3,... lignes, une symétrie en commençant par le bit de droite et changer la valeur du bit de gauche.

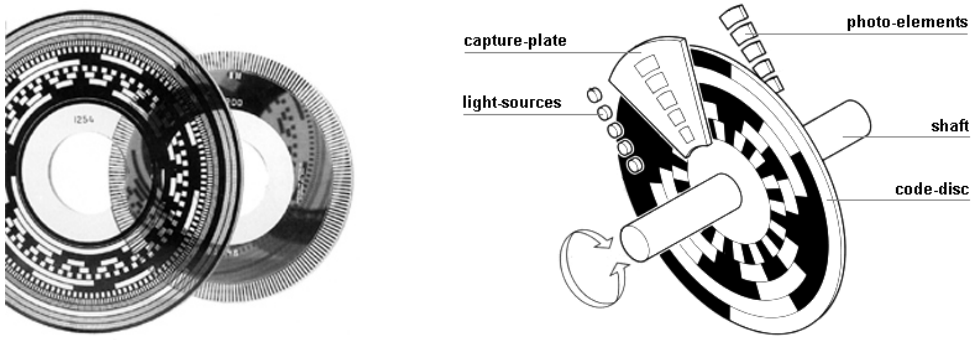
Application : Écrire en code Gray le nombre 56.

Méthode :

- Ecrire 56 en binaire : 0111000.
- Multiplier le nombre par 2 (décalage vers la gauche de chaque bit) : 1110000.
- Faire un OU exclusif (somme bit à bit, sachant que 1+1=0 et 0+0=0) : 0111000+1110000 = 1001000.
- Diviser par 2 (décalage à droite de chaque bit) : soit 0100100

0	0	0	0	0	0	0	0	0	
1	0	0	0	1	0	0	0	1	1ère symétrie
2	0	0	1	0	0	0	1	1	
3	0	0	1	1	0	0	1	0	2ème symétrie
4	0	1	0	0	0	1	1	0	
5	0	1	0	1	0	1	1	1	
6	0	1	1	0	0	1	0	1	
7	0	1	1	1	0	1	0	0	3ème symétrie
8	1	0	0	0	1	1	0	0	
9	1	0	0	1	1	1	0	1	
10	1	0	1	0	1	1	1	1	
11	1	0	1	1	1	1	1	0	
12	1	1	0	0	1	0	1	0	
13	1	1	0	1	1	0	1	1	
14	1	1	1	0	1	0	0	1	
15	1	1	1	1	1	0	0	0	

Exemple : Écrire 26 en code Gray :



(a) Disque du codeur absolu (b) Principe du codeur absolu

FIGURE 4 : Codeur absolu.

II.2.4 Code p parmi n

Le code p parmi n consiste à choisir parmi n bits, p bits égaux à 1 et $n-p$ bits égaux à 0. Le nombre de combinaisons répondant à cette définition est égal à $C_n^p = \dfrac{n!}{p!(n_p)!}$.

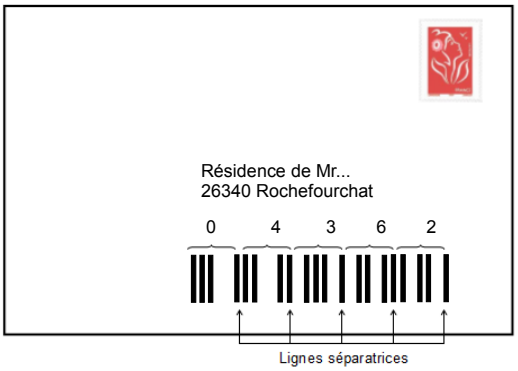
Par exemple, le tableau ci-contre présente un code 3 parmi 5 et met en correspondance la valeur décimale et son écriture en code 3 parmi 5. Ce code comporte trois « 1 » et deux « 0 » et permet 10 combinaisons : $C_5^3 = \dfrac{5!}{3!(5-3)!} = 10$.

Décimal	Code 3 parmi 5				
0	0	0	1	1	1
1	0	1	0	1	1
2	0	1	1	0	1
3	0	1	1	1	0
4	1	0	0	1	1
5	1	0	1	0	1
6	1	0	1	1	0
7	1	0	0	0	1
8	1	1	0	1	0
9	1	1	1	0	0

L'avantage du code p parmi n est qu'il peut être personnalisé. En effet il existe $C_n^p!$ arrangements de la codification. Ainsi le code 3 parmi 5 procure $C_5^3! = 10!$ arrangements différents, soit 3 628 800 possibilités.

L'inconvénient de ce code non pondéré est qu'il est nécessaire d'avoir la table de correspondance pour décoder un nombre.

Application : Tri postal. Le tri automatique des plis postaux nécessite la codification numérique du bureau distributeur. Le code numérique est lu automatiquement (lecture O.C.R. : Reconnaissance Optique de Caractères) ou par un opérateur. Il est ensuite traduit en un code à barres qui est matérialisé sous la forme de bâtonnets déposés sur le pli postal. Les « 1 » sont codés par des bâtonnets foncés, les « 0 » sont matérialisés par des bâtonnets pratiquement transparents, ces « 1 » et ces « 0 » étant séparés par des espaces. Le code 3 parmi 5 retenu pour le code postal est celui donné précédemment.



Le tri automatique met donc un code barre tel que celui représenté ci-contre :

Exemple : Indiquer le code postal correspondant au code barre suivant : 

III Logique combinatoire

III.1 Définitions

III.1.1 Variables binaires

De nombreux composants utilisés en automatisme ne peuvent normalement prendre que deux états différents : lampe allumée ou éteinte, bouton-poussoir actionné ou relâché, moteur tournant ou à l'arrêt, vérin pneumatique sorti ou rentré...

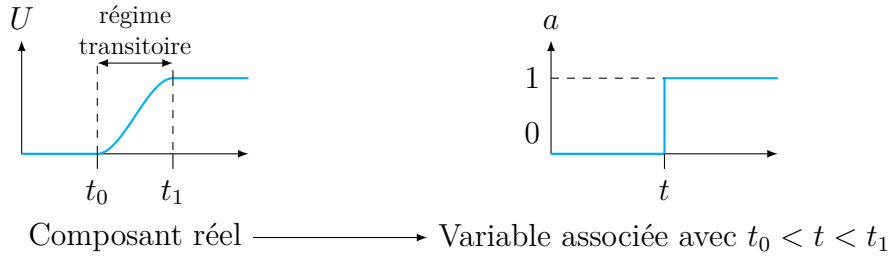
A chacun de ces composants, on peut associer une **variable binaire** (ou logique, ou Tout Ou Rien) qui ne peut prendre que deux valeurs notées 0 ou 1 (vrai ou faux, oui ou non).

> Une variable a est binaire si et seulement si elle ne peut prendre, à chaque instant, qu'une seule valeur parmi un ensemble de 2 valeurs possibles.

Exemple : Lister les variables binaires du support de cours

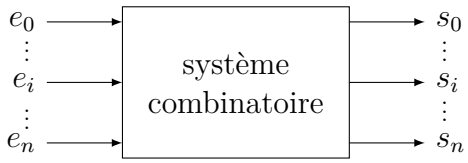
Remarque :

- le comportement « Tout ou Rien » (TOR) ne correspond qu'au comportement normalement prévu en régime stabilisé et en l'absence de tout dysfonctionnement.
- l'association d'une variable binaire à un composant ne peut pas rendre compte des états transitoires apparaissant entre deux états stables. C'est donc une simplification du comportement réel.



III.1.2 Système binaire

Un système est dit binaire ou logique si les variables d'entrée et de sortie sont binaires.



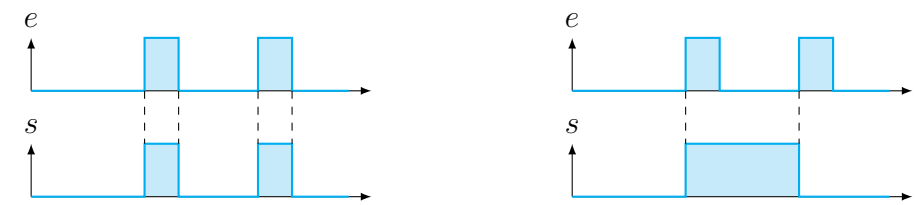
III.1.3 Systèmes combinatoire et séquentiel

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

La Figure 5(a) illustre l'appuie sur un interrupteur à deux positions :
— sur la position 0 (entrée à 0), la lumière est éteinte (sortie à 0),
— sur la position 1 (entrée à 1), la lumière est allumée (sortie à 1).

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

La Figure 5(b) illustre l'appuie sur un interrupteur de type bouton poussoir :
— sur la position 0 (entrée à 0), la lumière est éteinte (sortie à 0),
— sur la position 1 (entrée à 1, appui sur le bouton), la lumière s'allume (sortie à 1),
— sur la position 0 (entrée à 0, relâchement du bouton), la lumière reste allumée (sortie à 0),
— sur la position 1 (entrée à 1, appui sur le bouton), la lumière s'éteint (sortie à 0).



(a) Système logique combinatoire

(b) Système logique séquentielle

Figure 5 : Différence entre système logique combinatoire et séquentiel.

III.2 Algèbre de Boole

Pour traiter les systèmes combinatoires, on fait appel à l'algèbre de Boole.

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

— complémentarité : NON

$$\begin{aligned} B &\longrightarrow B \\ a &\longmapsto \text{NON}(a) = \overline{a} \end{aligned}$$

$\overline{a}$ est à 1 si a est à 0 et réciproquement (voir table 1(a)).

— produit booléen : ET

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

$a \cdot b$ est à 1 si et seulement si a est à 1 ET b est à 1 (voir table 1(b)).

— somme booléenne : OU

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

$a + b$ est à 1 si et seulement si a est à 1 OU b est à 1 (voir table 1(c)).

Deux expressions logiques A et B équivalentes seront notés avec le symbole « = » : $A = B$.

Propriétés Le tableau suivant donne les propriétés usuelles des lois ET et OU.

	Loi OU	Loi ET
Commutativité	$a + b = b + a$	$a \cdot b = b \cdot a$
Distributivité	$a \cdot (b + c) = a \cdot b + a \cdot c$	$(a \cdot b) + c = (a + c) \cdot (b + c)$
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	$a + \overline{a} = 1$	$a \cdot \overline{a} = 0$
Idempotence	$a + a = a$	$a \cdot a = a$

Tableau 1: Propriétés usuelles de l'algèbre de Boole.

D'autres propriétés algébriques permettent d'aider à la simplification des expressions :

— double complément : $\overline{\overline{a}} = a$
— absoption : $a + a \cdot b = a$
— inclusion : $a + \overline{a} \cdot b = a + b$

Théorèmes de De Morgan
Le complément d'un OU est le ET des compléments : $\overline{a + b} = \overline{a}.\overline{b}$
Le complément d'un ET est le OU des compléments : $\overline{a \cdot b} = \overline{a} + \overline{b}$

Exemple : Déterminer les expressions complémentaires $\overline{P}$ et $\overline{Q}$ des expressions logiques suivantes : $P = x \cdot y \cdot (z + \overline{t})$ et $Q = \overline{x} \cdot \overline{y} + y + x \cdot t$.

III.3 Représentation des fonctions logiques

Pour définir une fonction logique associée à un processus, il existe plusieurs représentations possibles :

— représentation algébriques : $f(a, b, c, d) = a \cdot b \cdot \overline{c} + b \cdot c \cdot \overline{d}$;
— représentation graphique : table de vérité, logigramme, chronogramme.

Remarque : Il existe d'autres représentations graphiques non traitées dans ce paragraphe.

III.3.1 Table de vérité

La table de vérité d'une fonction logique est un tableau comportant une colonne par variable logique et une colonne pour la fonction.
Le nombre de lignes est donné par le nombre de variables de la fonction et représente le nombre de combinaisons possibles de l'état de ces variables. Si la fonction dépend de n variables logiques, alors la table de vérité comportera 2^n lignes.

Le tableau 2 donne les tables de vérités des fonctions logiques de l'algèbre de Boole.

(a) Table de la loi NON

a	$\overline{a}$
0	1
1	0

(b) Table de la loi ET

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

(c) Table de la loi OU

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

Tableau 2: Table de vérité des lois de l'algèbre de Boole.

Exemple : Compléter les tables de vérité du théorème de De Morgan

a	b	$\overline{a}$	$\overline{b}$	$\overline{a \cdot b}$	$\overline{a} + \overline{b}$	$\overline{a + b}$	$\overline{a} \cdot \overline{b}$
0	0						
0	1						
1	1						
1	0						

III.3.2 Équations logiques

Il est possible à l'aide de la table de vérité de définir une équation donnant les variables de sortie en fonction des variables d'entrée.

Il existe 2 modes d'écriture particuliers :

— somme canonique. (somme de produits) : $(a \cdot b) + (c \cdot d)$

— produit canonique (produit de sommes) : $(a + b) \cdot (c + d)$

Méthode de détermination de la somme canonique : pour chaque ligne où la sortie vaut 1, déterminer la combinaison d'entrées correspondante à l'aide de l'opérateur ET puis sommer ces combinaisons.

a	b	S	
0	0	0	
0	1	1	$S = 1$ quand $a = 0$ ET $b = 1$ soit quand on a l'expression $\overline{a} \cdot b$ vraie.
1	0	0	
1	1	1	$S = 1$ quand $a = 1$ ET $b = 1$ soit quand on a l'expression $a \cdot b$ vraie.

Au final, on a $\boxed{S = \overline{a} \cdot b + a \cdot b}$.

Exemple : Déterminer les expressions des fonctions logiques du support de cours.

III.3.3 Logigrammes

> Un logigramme est une représentation graphique d'une fonction logique utilisant des composants élémentaires appelés « portes logiques ». Ces représentations sont particulièrement utilisées dans les documentations techniques des composants électroniques.
> La table 3 donne les principaux symboles logiques dans les normes européennes et américaines.

Exemple : Réaliser les logigrammes associées aux fonctions logiques du support de cours.

III.3.4 Chronogramme

Une autre représentation classique des variables et fonctions logiques est le chronogramme.

> Un chronogramme est une représentation de l'évolution temporelle d'une variable ou une fonction logique en fonction du temps.

Exemple : Soit la fonction logique $f(a, b) = a + b$. La figure suivante donne l'évolution de f en fonction du temps.

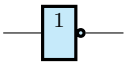
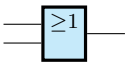
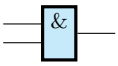
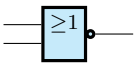
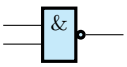
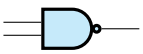
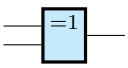
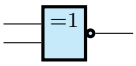
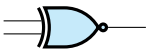
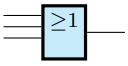
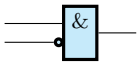
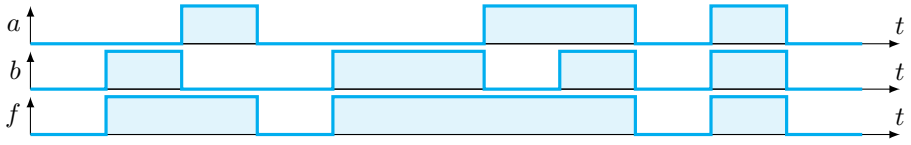
Fonction logique	Symbole européen	Symbole américain
NON $\overline{a}$		
OU $a+b$		
ET $a \cdot b$		
NON OU $\overline{a+b}$		
NON ET $\overline{a \cdot b}$		
OU EXCLUSIF $a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$		
ET EXCLUSIF / IDENTITÉ $a \otimes b = \overline{a \oplus b} = \overline{a} \cdot \overline{b} + a \cdot b$		
Il peut y avoir plus de deux entrées sur une porte logique :		
OU à 3 entrées		
Il peut y avoir des entrées inhibées (l'entrée est complémentée) :		
ET à 2 entrées la 2nde est complémentée		

TABLEAU 3: Tableau des portes logiques.



Exemple : Compléter les chronogrammes du support de cours.

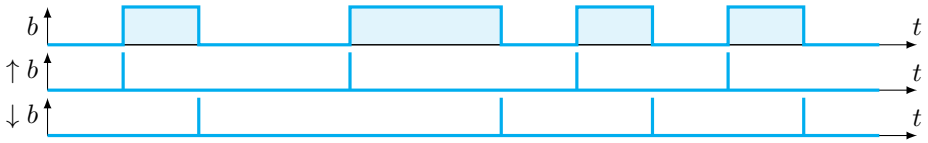
Remarque : On s'aide d'un chronogramme pour savoir si un système est combinatoire ou séquentiel. Il suffit de regarder si pour un même état des variables d'entrée, les sorties sont toujours les mêmes (cf. exemple de l'interrupteur).

Notion de fronts montant et descendant

Les fronts permettent de représenter le changement d'état d'une variable. Cette notion est très importante dans le cadre de la synchronisation des processus mais aussi la prise en compte de l'évolution du système à partir de capteurs (exemple appui sur un bouton poussoir, codeur incrémental...).

Le front montant d'une variable 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 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.

Exemple : Soit la variable logique b. La figure suivante donne l'évolution de b en fonction du temps et l'évolution des fronts montant et descendant de b.

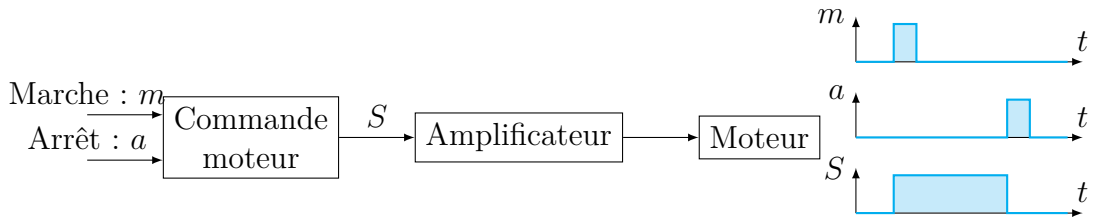


IV Logique séquentielle

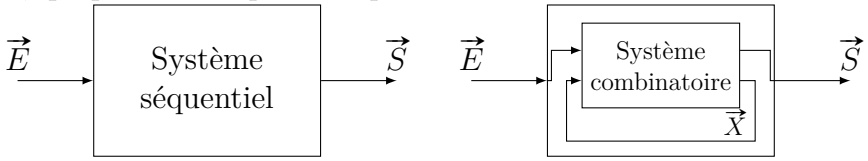
IV.1 Introduction

Un système séquentiel est un système dont les sorties S_i dépendent de l'histoire des entrées $E_i(t)$.

Exemple : commande d'un moteur électrique par un système séquentiel. On remarque sur le chronogramme de droite que la sortie S peut présenter une valeur différente (0 ou 1) pour une configuration identique des entrées m et a.



Le système est capable de mémoriser de l'information. Cette information mémorisée est l'état du système, qui peut être représenté par un vecteur d'état booléen : $\vec{X} = (x_1, x_2, \ldots, x_n)$.



Connaissant $\vec{X}$ et $\vec{E}$ (le vecteur d'état booléen des entrées), la sortie $\vec{S}$ peut être déterminée comme une fonction booléenne de $\vec{X}$ et de $\vec{E}$: $\vec{S} = f(\vec{X}, \vec{E})$ (donc via un système combinatoire).

L'état interne $\vec{X}$ à l'instant t dépend de $\vec{E}(t)$ et de l'état interne immédiatement précédent : $\vec{X}(t) = g(\vec{E}(t), \vec{X}(t - \Delta t))$.

Les tables de vérité ou logigrammes ne permettent plus de décrire l'évolution de S en fonction de E. On peut utiliser alors les chronogrammes (aussi appelés diagrammes de Gantt).

Pour décrire le fonctionnement global du système, de la partie logicielle, on utilise plutôt des outils de description du SysML : les diagrammes de séquence, d'états ou d'activité.

IV.2 Diagramme de séquence – SysML

Ce diagramme permet de définir la chronologie des interactions entre les acteurs (utilisateurs) et le système. Il permet de décrire un scénario correspondant à un cas d'utilisation.

Le principe est de décrire les messages envoyés entre les acteurs dans l'ordre chronologique (le temps s'écoule vers le bas). Le comportement interne des constituants n'est pas détaillé dans ce diagramme.

Les éléments constitutifs du diagramme sont (Figure 6) :

— les lignes de vie qui représentent les acteurs (du diagramme de cas d'utilisation) participant à la communication. Elles peuvent être actives ou inactives (bandes verticales).

— les messages : éléments de communication unidirectionnels entre les lignes. Il existe plusieurs types de messages :
  - Message synchrone : l'émetteur est bloqué en attente de réponse
  - Message asynchrone : pas de réponse attendue
  - Message réflexif : comportement interne

— les opérateurs qui permettent de réaliser des tests, des opérations en parallèle, des boucles...
  - **loop** (boucle) : plusieurs exécutions successive d'un même fragment,
  - **opt** (optionnel) : exécution si la condition fournie est vraie
  - **alt** (fragments alternatifs) : exécution du fragment si une condition parmi plusieurs est vraie
  - **ref** : appel d'un autre diagramme de séquence défini ailleurs
  - **par** : exécution en parallèle

Exemple : Lire le diagramme de séquence du support de cours.

IV.3 Diagramme d'états – SysML

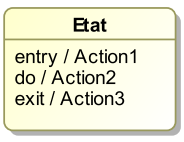
Ce diagramme permet de décrire les différents états que peut traverser le système en fonction des évènements perçus.

Il est utilisé essentiellement pour décrire les modes de marche, d'arrêt, d'attente d'un système ou sous-système.

Il utilise plusieurs symboles particuliers :

— un état représenté par un bloc qui représente une situation d'une durée finie. Pendant cet état, il peut se produire des actions. L'état initial dans lequel est placé le diagramme au départ est représenté par un symbole particulier ●. On peut également utilisé un symbole pour représenter le ou les états finals (le diagramme ne peut plus évoluer) ◉.

— les événements représentés par des flèches permettent de décrire comment passer d'un état à un autre. On peut associer une condition booléenne (appelée condition de garde) à l'événement ou utiliser des mots clés particuliers :
  - `after x s` : après x secondes on passe à l'état suivant,
  - `at h` : à l'heure indiquée, on passe à l'état suivant
  - `when x=xc` : on passe à l'état suivant quand une valeur atteint une valeur donnée
  - si aucun texte n'est noté sur la flèche (on parle de transition de complétion), ceci indique que l'on passe à l'état suivant dès que les actions associées à un état sont terminées.



Il peut être utile de spécifier à l'intérieur d'un état des actions qui sont effectuées. On utilise pour cela le mot clé `do` suivi d'une ou plusieurs actions. On peut spécifier plus précisément à quel moment se feront les actions à l'intérieur d'un état :

— `entry` suivi d'une action indique que l'action est faite en entrant dans l'état

— `exit` suivi d'une action précise que l'action est faite en sortie de l'état

Un état peut contenir lui même un diagramme d'état. On parle d'état composé ou composite. Comme tout diagramme d'état, l'état composite contiendra une étape initale qui sera activée

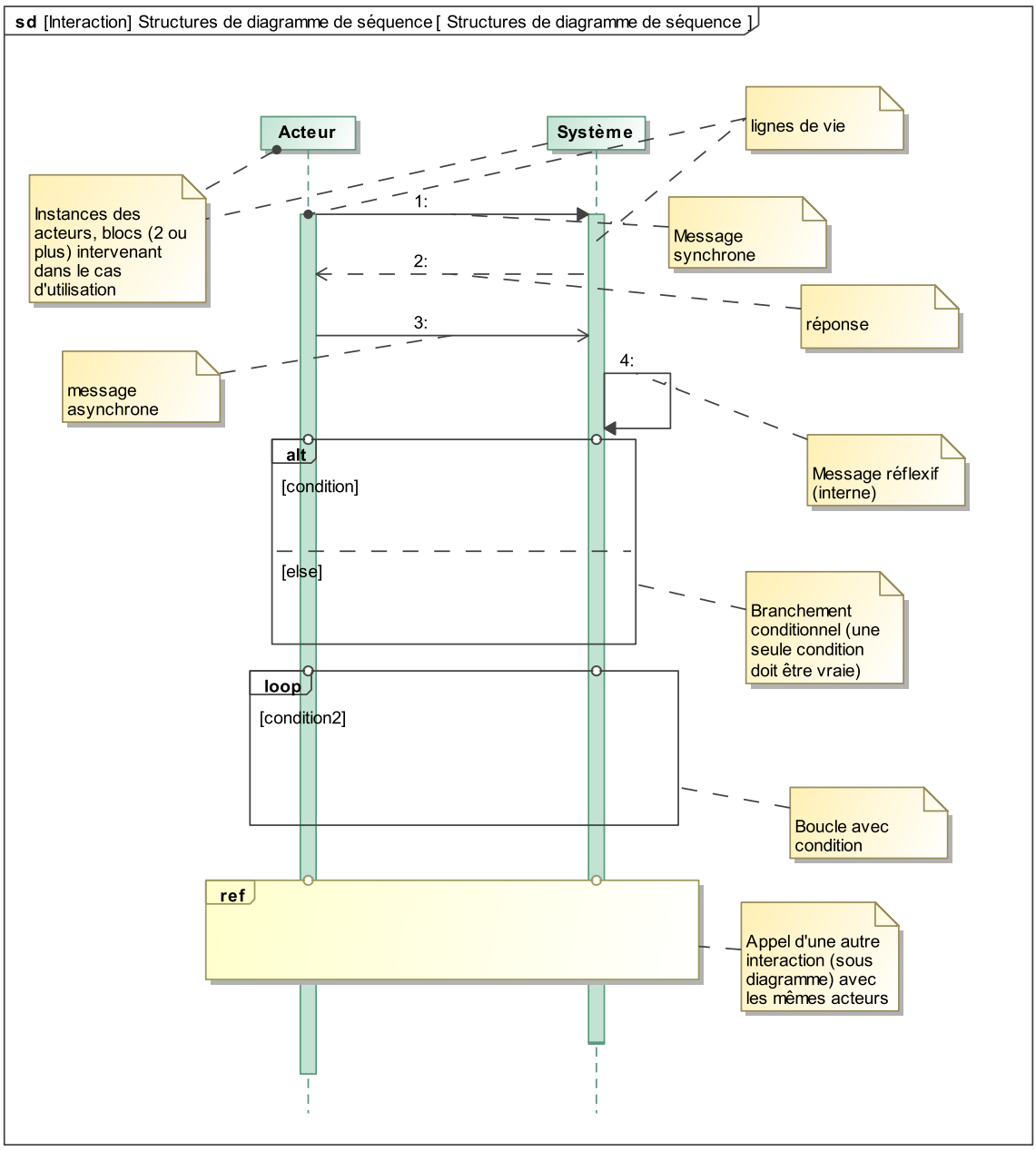


FIGURE 6 : Structure d'un diagramme de séquence.

lorsqu'on entrera dans l'état composite. On le représente à l'intérieur d'un bloc ou bien on utilise un symbole pour faire référence au diagramme d'état.

Exemple : Lire le diagramme d'état du support de cours.

Exemple : Réaliser le diagramme d'états demandé.

IV.4 Diagramme d'activités – SysML

Ce diagramme permet de décrire le déroulement séquentiel d'un processus en spécifiant les actions, activités ou tâches réalisées. Il peut aussi préciser ce qui est produit, consommé ou transformé au cours de l'activité. Il est très utile pour décrire précisément ce qui se passe dans un état donné de diagrammes d'états.

Dans sa forme la plus restreinte ce diagramme correspond à un algorigramme et peut directement être traduit en programme informatique.

Il utilise les mêmes symboles de base que le diagramme d'état (blocs et flèches). Cependant,

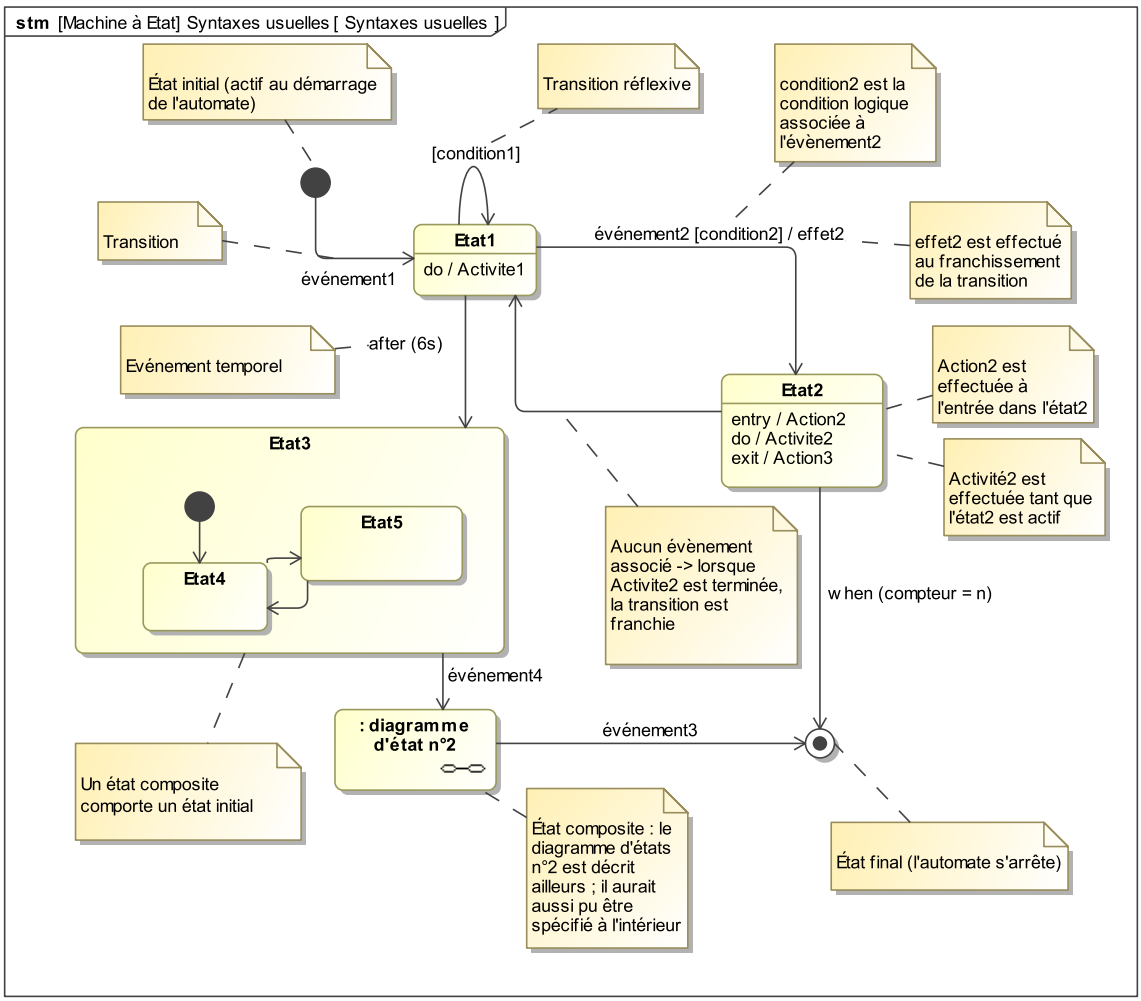
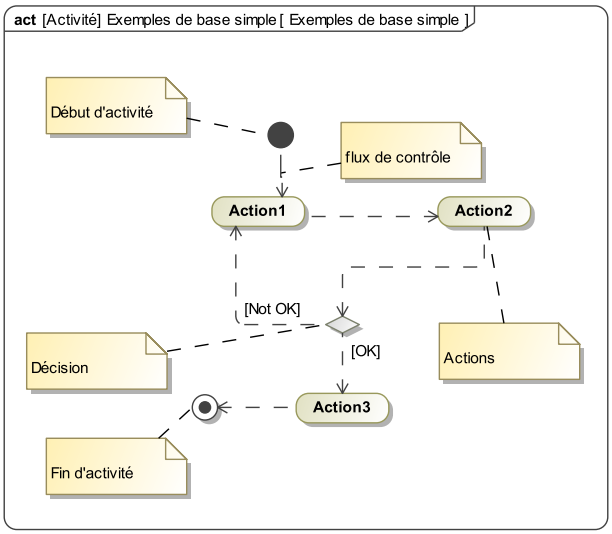


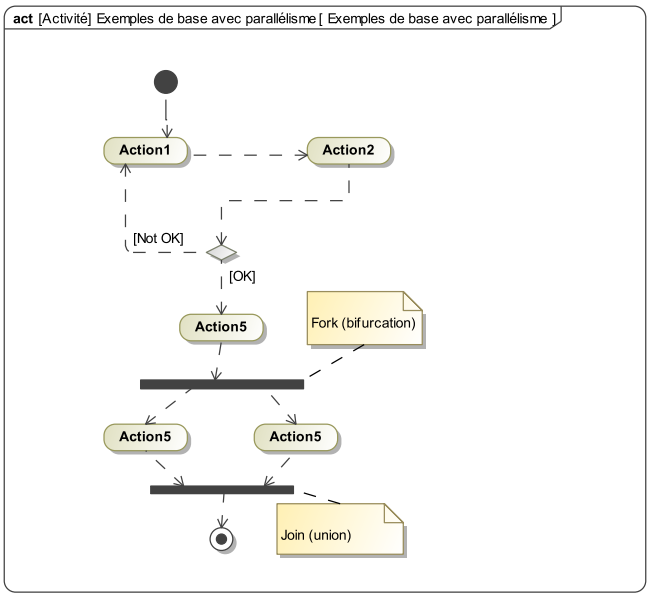
FIGURE 7 : Structures usuelles d'un diagramme d'état.

aucune condition n'est indiqué sur la flèche car celle-ci représente un signal émis dès que les actions associées à un bloc sont terminées.

Pour représenter un choix à faire entre plusieurs conditions exclusives, on utilise le symbole de branchement conditionnel. Les conditions sont notées entre crochets en sortie du branchement conditionnel (FIGURE 8(a)).



(a) Exemple avec 1 choix



(b) Exemple avec parallèlisme

FIGURE 8 : Diagramme d'activité – Algorigramme

Si des actions doivent se faire en parallèle ou doivent être synchronisées, on utilise des symboles particuliers appelés Fork et Join (FIGURE 8(b)).

Remarque : Le diagramme d'activité est plus général qu'un algorigramme car non seulement il présente les flux de contrôle mais également les flux de matière, énergie, information qui sont modifiés par le système. Il représente ainsi une spécification plus précise de ce que l'on décrit par des diagrammes IBD.

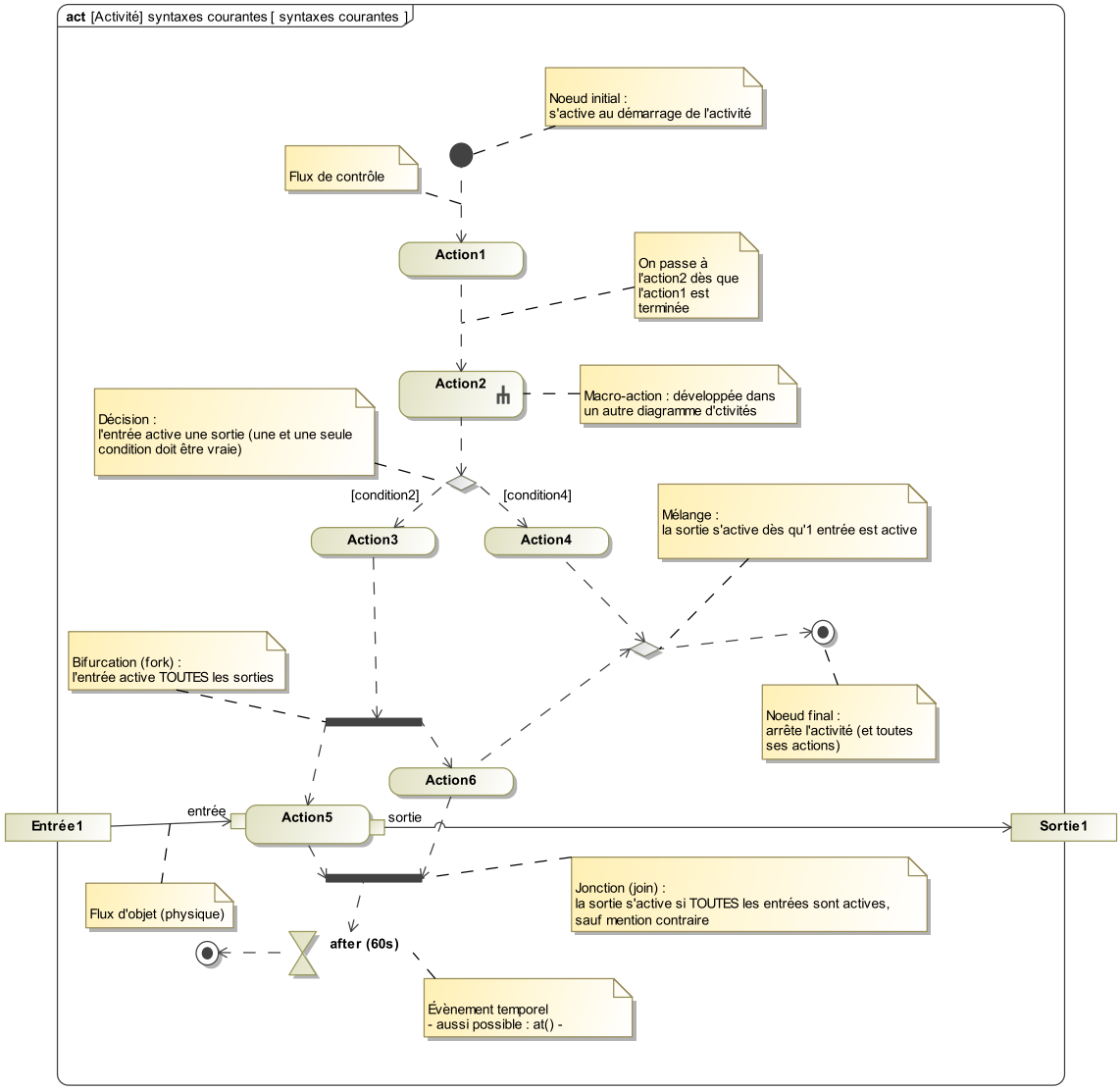


FIGURE 9 : Structures usuelles d'un diagramme d'activités.

TD 1 – Exercices de transfert

Additionneur logique

Présentation

On s'intéresse à un additionneur logique qui permet de réaliser l'addition de 3 bits a, b et c.

Cet élément de base permet de réaliser une addition entre deux mots binaires composés de plusieurs bits.

Le résultat de l'opération élémentaire est composé de la somme S et d'une retenue R.

Objectif

L'objectif est de déterminer les équations logiques de R et S en fonction des entrées a, b et c, puis de réaliser le logigramme de ces équations.

Q1. Quels sont les variables logiques présentent dans ce composant ?

Table de vérité

La table de vérité régissant l'évolution des sorties en fonction des entrées est :

a	b	c	R	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Q2. Déterminer les équations logiques de R et S en fonction de a, b et c.

Q3. En utilisant les règles de manipulations algébriques, montrer que : $R = ab + bc + ac$.

Logigramme

Q4. Réaliser le logigramme des fonctions R et S.

a b c

Drone de cinéma

Introduction

Afin de réaliser des prises de vue aériennes en haute définition lors de la réalisation de reportages ou de films, des drones sont fréquemment utilisés. Ces appareils sont loués par la production à des entreprises spécialisées qui fournissent, outre le matériel, un pilote maîtrisant parfaitement le vol de son engin et pouvant donc répondre à toutes les demandes du réalisateur et de son chef opérateur [1], le tout pour un prix très raisonnable par rapport aux solutions classiques utilisant des avions ou des hélicoptères : on passe en effet d'un prix horaire d'au moins 2000 £ HT pour un avion ou un hélicoptère (donc environ 12 000 £ HT la journée de location) à un prix journalier de 1000 à 3000 £ HT selon les prestations attendues pour le drone.

L'entreprise Ciné-Drone est un acteur important au niveau mondial pour la réalisation de prises de vue par drone. Elle est implantée en France, au Costa-Rica, aux États-Unis et en Chine et propose ses services dans le monde entier. Afin de répondre à toutes les demandes des réalisateurs, cette société propose trois types de drones à rotors équilibrés dynamiquement à structure carbone et nacelle stabilisée par gyromètre (voir figure 1).

FIGURE 1 : Drone Ciné-Drone, modèles S et L.

Le tableau de la figure 2 montre les principales différences des trois modèles S (small), L (large) et XL (extra large).
Pour tous les modèles :

- Les pieds sont repliables en vol, ce qui permet à la caméra de viser toutes les directions.
- Le drone peut s'orienter sur ± 180 ° en lacet (rotation autour de l'axe vertical) et ± 45 ° en roulis et tangage (rotations autour de l'axe longitudinal et de l'axe transversal).
- La nacelle supportant la caméra peut s'orienter ± 180°en lacet (rotation autour de l'axe vertical), ± 90 ° en roulis (rotation autour de l'axe longitudinal) et ± 135 ° en tangage (rotation autour de l'axe orthogonal).
- Transmission vidéo au niveau normalisé de 5,8 GHz - 10 mW.

1. Le chef opérateur, aussi appelé « directeur de la photo », est la personne responsable de la prise de vue sur un tournage.

Caractéristiques	**Modèle S**	**Modèle L**	**Modèle XL**
Moteurs	6 rotors 15 A - 12 V	8 rotors 30 A - 12 V	8 rotors 60 A - 12 V
Charge embarquée	jusqu'à 1,5 kg	jusqu'à 2,3 kg	jusqu'à 5 kg
Charge maximale	3 kg	5 kg	14 kg
Exemple de caméra	Go-Pro 2	Black Magic	Red Epic
Vent maximal	25 kmh^{-1}	35 kmh^{-1}	65 kmh^{-1}
Durée de vol maximale	12 min	8 min	5 min

FIGURE 2 : Caractéristiques principales des drones de l'entreprise Ciné-Drone.

— Conformément à la législation, deux « télépilotes » sont présents pour l'utilisation du drone [2] : le premier gère l'attitude [3] de la structure volante du drone par rapport au sol selon les attentes du réalisateur ; le second gère le positionnement angulaire de la nacelle par rapport au drone selon les attentes du chef opérateur.

La qualité de la prise de vue (profondeur de champ, zoom, etc.) est gérée par le chef opérateur grâce au retour vidéo en temps réel. La coordination avec l'équipe de télépilotes permet d'obtenir des images conformes à celles réalisées par hélicoptère.
Le drone est livré sans caméra mais intègre :

— une nacelle alvéolée qui permet de fixer tous les appareils de prise de vue (appareils photo numériques et caméras) existants ;
— un système de communication vidéo avec une prise de connexion universelle pour les caméras professionnelles fournies par l'équipe de tournage.

Les caméras utilisées dans les prises de vue haute définition disposent d'une mise au point autofocus et d'un système de stabilisation optique permettent d'obtenir une image nette malgré les vibrations de l'appareil dès lors que les mouvements globaux (translations et rotations selon les trois directions de l'espace) restent assez lents : le pilotage du drone par un professionnel permet donc d'obtenir une image nette dans la très grande majorité des configurations de vol ce qui réduit la durée, et donc le prix, de la phase de traitement informatique de l'image.

Les drones de cinéma de l'entreprise Ciné-Drone sont par ailleurs équipés d'une gestion spécifique « anti-crash » avec deux niveaux de sécurité :

— Dès que le niveau de batterie devient inférieur au niveau de sécurité (niveau 1), le drone coupe le transfert de flux vidéos et descend automatiquement au sol en un point défini par GPS : le pilote peut accompagner cet atterrissage pour changer de cible mais ne peut pas forcer le maintien en vol.
— Si le système n'a plus les capacités de redescendre au sol à cause d'un niveau insuffisant de batterie (niveau 2), le drone sort ses pieds, coupe les moteurs et déclenche son parachute, ce qui assure un retour au sol sans casse.

Cette double sécurité, définie, conçue et implantée par l'entreprise Ciné-Drone n'est absolument pas obligatoire mais elle permet de préserver l'intégrité des équipements de prise de vue en cas de panne de batterie, ce qui assure la grande renommée de cette entreprise auprès des

2. Dans le cas particulier du modèle XL, l'équipe est accompagnée d'un informaticien pour la gestion et le contrôle des flux de commande et de contrôle.

3. L'attitude correspond aux six paramètres permettant de définir la position et l'orientation du drone dans l'espace : déplacements en abscisse, ordonnée et altitude, angles de roulis, tangage et lacet.

réalisateurs.

Diagramme de séquence

Pour le système étudié, il est possible de mettre en place le diagramme présenté sur la figure 3 qui décrit l'évolution séquentielle des échanges entre les deux techniciens de vol (le pilote du drone et le pilote de la nacelle supportant la caméra) lors de la phase de préparation au vol : cette description répond à une exigence portant sur la norme et ne représente donc pas un scénario complet car seule la phase de test préalable à l'utilisation du système est représentée, ce qui correspond à une partie des cas d'utilisation « Piloter le drone » et « Orienter la nacelle ».

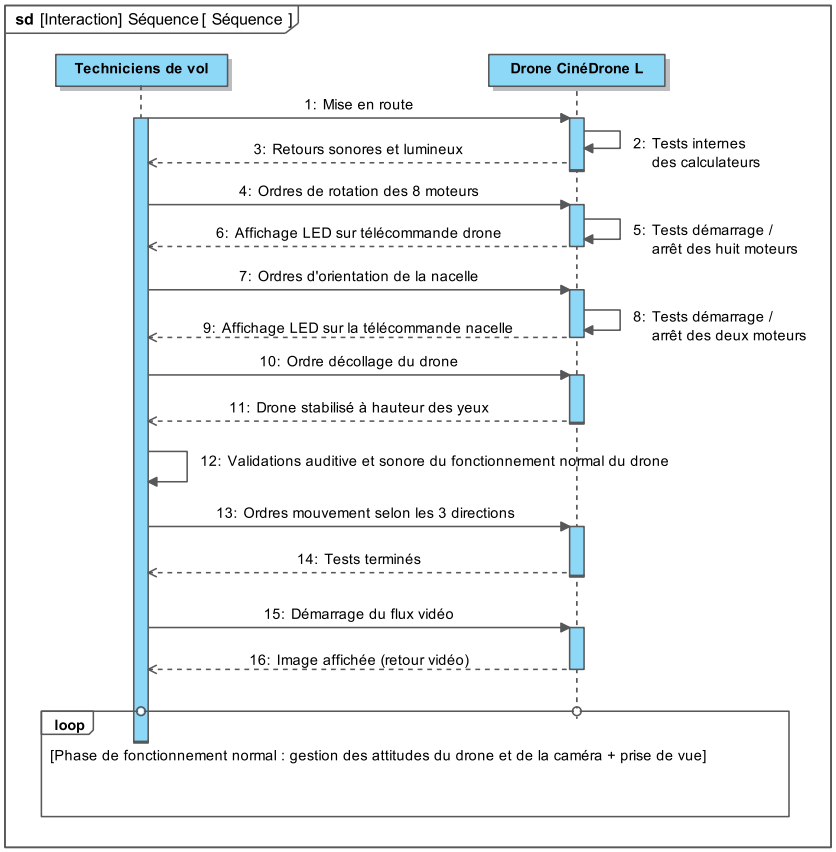


FIGURE 3 : Un exemple de diagramme de séquence pour le drone de cinéma.

Le diagramme d'états

Pour le système étudié, il est possible de mettre en place le diagramme d'états (indicateur **stm**) présenté sur la figure 4 qui permet de représenter les différents modes de fonctionnement du système, ainsi que les évènements qui permettent de passer de l'un à l'autre.

L'initialisation est un mode temporaire dans lequel le système effectue tous ses tests avant de donner la main aux pilotes. La transition est ensuite franchie automatiquement car il n'y a pas d'évènement particulier à marquer (c'est la fin de l'initialisation). Viennent ensuite le mode de fonctionnement normal puis les modes d'urgence en cas de niveau de charge insuffisant.

Le comportement lors de l'entrée et de la sortie de chaque état est décrit : par exemple, l'entrée dans le mode urgence 1 (évènement interne *entry*) provoque automatiquement la coupure du flux vidéo pour économiser l'énergie et un mode de pilotage dégradé (pilotage à vue) où le drone ne peut que redescendre.

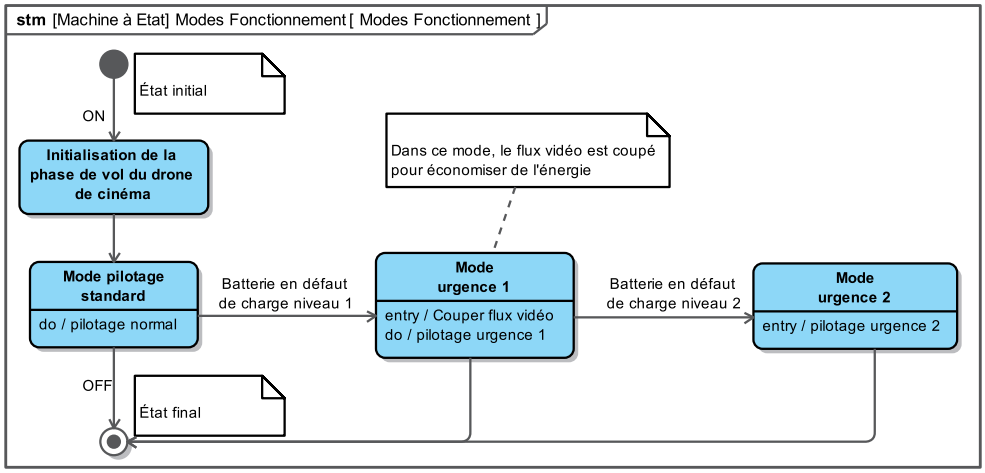


FIGURE 4 : Un exemple de diagramme d'états pour le drone de cinéma.

Le diagramme d'activités

Dans le diagramme d'états de la figure 4, ce qui se passe dans chaque mode n'est pas décrit : pour ce faire, il est possible de choisir des nouvelles machines d'états ou des activités.

Par choix, la description de chaque mode a été réalisée uniquement par des diagrammes d'activités (indicateur **act**), comme sur la figure 5 où il apparaît que :

— Le mode normal correspond à un pilotage complet du drone avec en parallèle l'envoi du flux vidéo (les deux barres noires marquent le début et la fin de l'envoi des deux flux).

— Dans le mode d'urgence 2, la procédure pour atterrir automatique est décrite, la note précisant les limitations de ce mode.

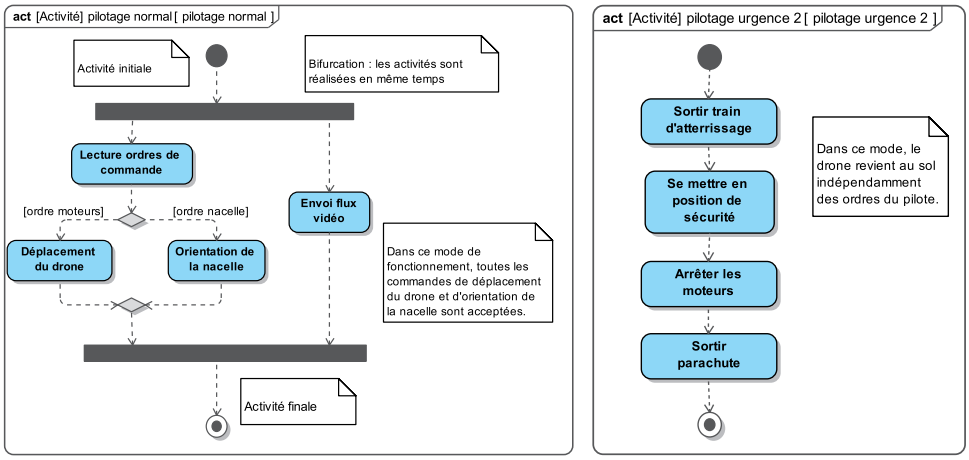


FIGURE 5 : Exemples de diagrammes d'activités du drone de cinéma.

Diagramme d'état d'une voiture

Le diagramme d'état de la voiture décrit la mise en marche et arrêt de celle-ci ainsi que le passage des vitesses.

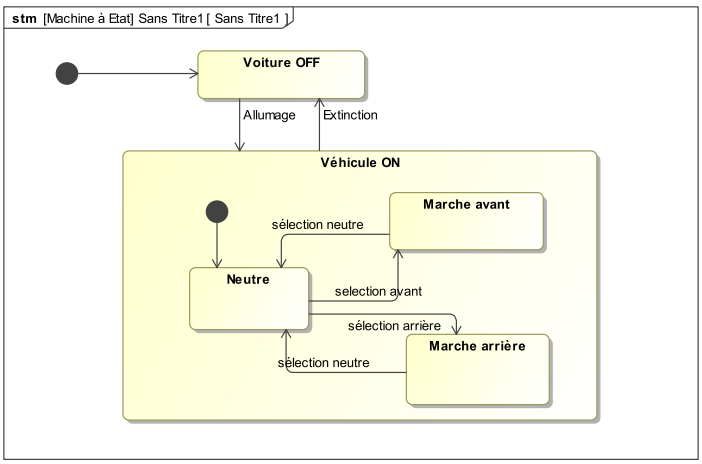


FIGURE 6 : Diagramme d'état d'une voiture.

Balance Halo

Présentation

La balance Halo de Teraillon est une balance de cuisine ayant une précision de 1 g. Elle permet de peser des aliments jusqu'à 3 kg. Elle se décline en 5 coloris et coûte 25 euros environ.

La balance possède un système de tarage du zéro et un bouton permettant de convertir les grammes en millilitres.

La notice d'utilisation du produit est proposée ci-dessous.

Diagramme de séquence

Le diagramme de cas d'utilisation le plus simple de la balance est donné ci-dessous.

Q5. En vous aidant de la notice d'utilisation, proposer un diagramme de séquence correspondant au cas d'utilisation où l'utilisateur allume l'appareil et tare la balance.

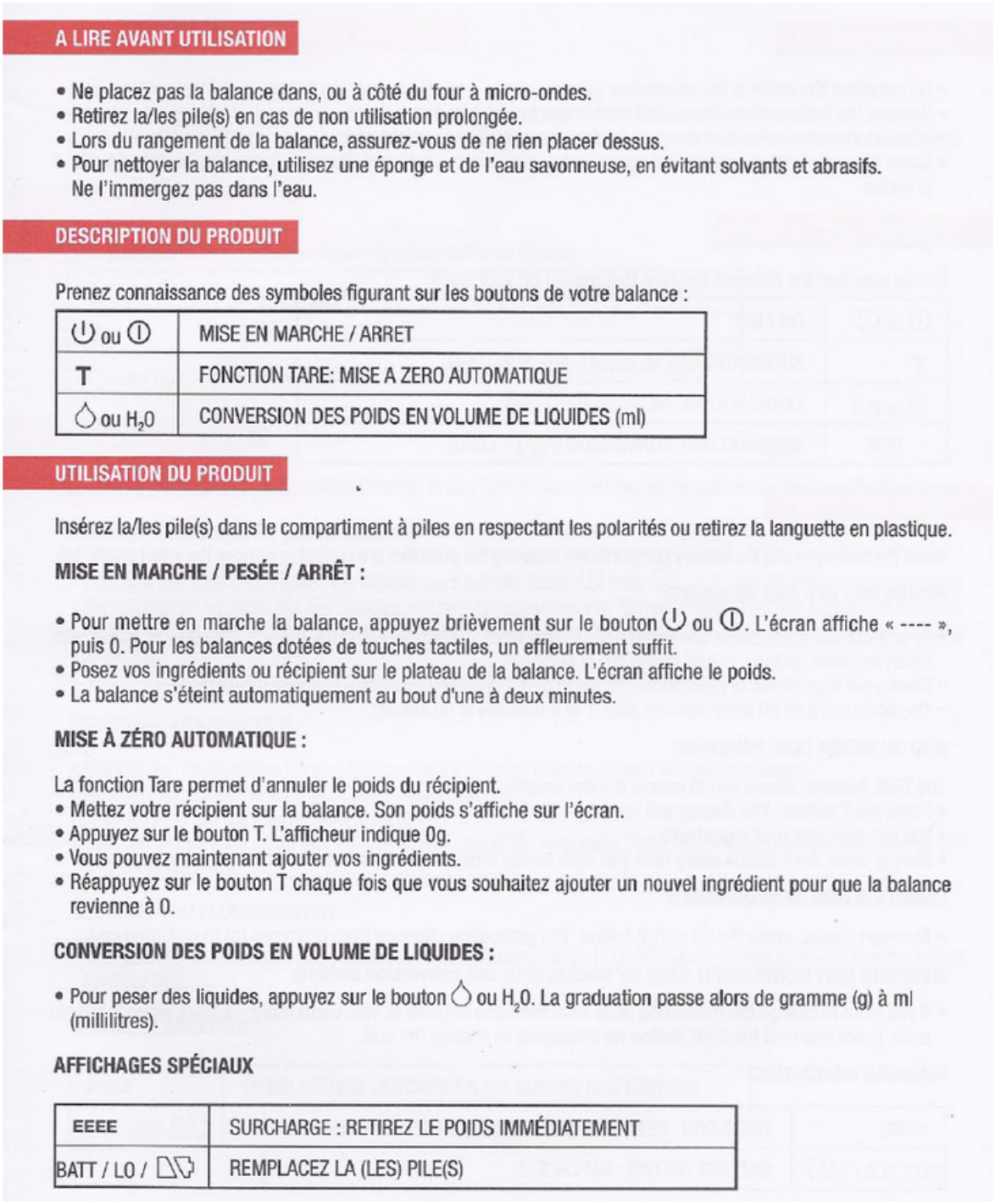
Q6. Compléter ce diagramme de séquence en ajoutant l'erreur qui apparaît si la masse du produit est supérieure à 3 kg ou si elle est inférieure.

Diagramme d'états

On considère les 4 états suivants : tarage, pesage, conversion, maintien bouton. La notice d'utilisation décrit le fonctionnement complet de la balance. On peut également éteindre la balance en restant appuyé au moins 2 secondes sur le bouton On/Off (état maintien bouton).

Q7. Compléter les événements permettant de passer d'un état à un autre.

Q8. Ajouter des actions associées aux états pour préciser le comportement de la balance.



A LIRE AVANT UTILISATION

- Ne placez pas la balance dans, ou à côté du four à micro-ondes.
- Retirez la/les pile(s) en cas de non utilisation prolongée.
- Lors du rangement de la balance, assurez-vous de ne rien placer dessus.
- Pour nettoyer la balance, utilisez une éponge et de l'eau savonneuse, en évitant solvants et abrasifs. Ne l'immergez pas dans l'eau.

DESCRIPTION DU PRODUIT

Prenez connaissance des symboles figurant sur les boutons de votre balance :

⏻ ou ⓘ	MISE EN MARCHE / ARRET
T	FONCTION TARE: MISE A ZERO AUTOMATIQUE
◇ ou H_2O	CONVERSION DES POIDS EN VOLUME DE LIQUIDES (ml)

UTILISATION DU PRODUIT

Insérez la/les pile(s) dans le compartiment à piles en respectant les polarités ou retirez la languette en plastique.

MISE EN MARCHE / PESÉE / ARRÊT :

- Pour mettre en marche la balance, appuyez brièvement sur le bouton ⏻ ou ⓘ. L'écran affiche « ---- », puis 0. Pour les balances dotées de touches tactiles, un effleurement suffit.
- Posez vos ingrédients ou récipient sur le plateau de la balance. L'écran affiche le poids.
- La balance s'éteint automatiquement au bout d'une à deux minutes.

MISE À ZÉRO AUTOMATIQUE :

La fonction Tare permet d'annuler le poids du récipient.
- Mettez votre récipient sur la balance. Son poids s'affiche sur l'écran.
- Appuyez sur le bouton T. L'afficheur indique 0g.
- Vous pouvez maintenant ajouter vos ingrédients.
- Réappuyez sur le bouton T chaque fois que vous souhaitez ajouter un nouvel ingrédient pour que la balance revienne à 0.

CONVERSION DES POIDS EN VOLUME DE LIQUIDES :

- Pour peser des liquides, appuyez sur le bouton ◇ ou H_2O. La graduation passe alors de gramme (g) à ml (millilitres).

AFFICHAGES SPÉCIAUX

EEEE	SURCHARGE : RETIREZ LE POIDS IMMÉDIATEMENT
BATT / LO / ⊟	REMPLACEZ LA (LES) PILE(S)

Figure 7 : Notice d'utilisation de la balance Halo.

TD 2 – Systèmes à logique combinatoire

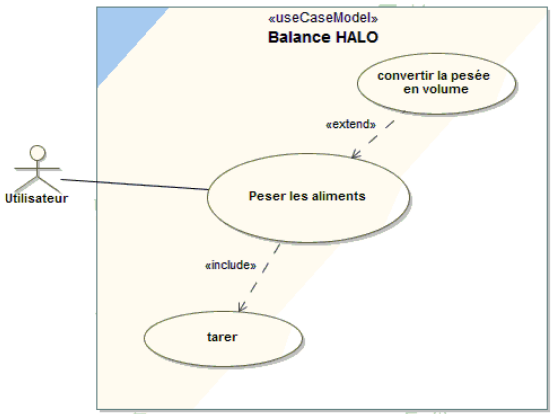


FIGURE 8 : Diagramme de cas d'utilisation de la balance Halo.

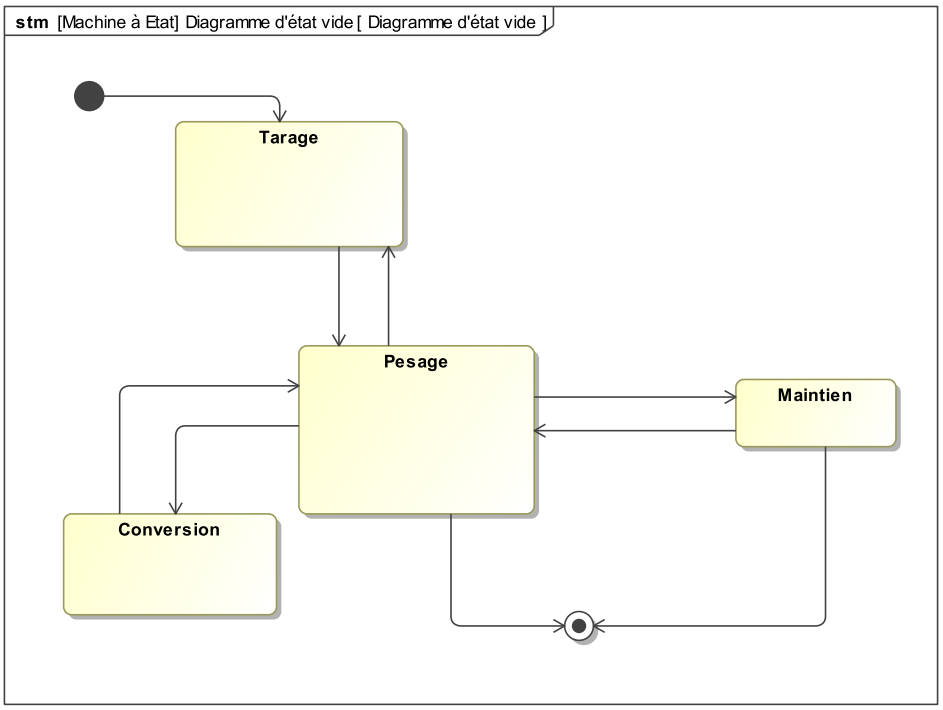
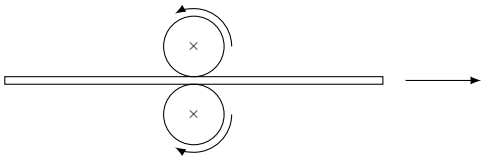


FIGURE 9 : Diagramme d'état à compléter de la balance Halo.

Exercice 1 : imprimante

On considère les rouleaux permettant de faire avancer le papier dans une imprimante. Ces rouleaux sont commandés par la variable logique P. Au repos $P = 0$. Dans la phase de défilement de papier $P = 1$.



On dispose de 3 entrées :
— touche avance papier accessible à l'utilisateur : a
— signal d'impression envoyé par l'ordinateur : b
— signal de présence de feuille donné par un capteur : c
Cahier des charges :

— le papier doit défiler si l'utilisateur appuie sur la touche avance papier.
— le papier doit avancer si le signal d'impression est envoyé, et si le capteur donne un signal de présence de feuille.

Q1. Déterminer l'équation logique définissant P.

Q2. Établir le logigramme de P.

Exercice 2 : Code barres

On considère un self service dans lequel les personnes disposent de cartes d'accès pour se faire repérer. Ces cartes d'accès comportent à la fois une piste magnétique et un code à barres. Lorsque l'on passe le code à barres devant un lecteur optique capable de lire le code « 2 / 5 – INTERLEAVED » (« 2 parmi 5 entrelacé »), le calculateur, relié au lecteur, interprète le code et le titulaire d'une carte d'accès au self-service est alors reconnu automatiquement.

Chaque titulaire est identifiable par un nombre à six chiffres décimaux (entre 0 et 9) notés C_5, C_4, C_3, C_2, C_1, C_0. Le code « 2 parmi 5 entrelacé » utilise 5 bits pour coder un chiffre décimal en vérifiant toujours que, parmi les 5 bits, il y en a 2 au niveau haut (valant 1) tandis que les 3 autres sont au niveau bas (valant 0). Par exemple : 10100 est un codage possible, alors que 10101 ne l'est pas.

Chaque chiffre de 0 à 9 est ainsi codé principalement sur les 4 premiers bits (a, b, c, d) de poids respectifs 1, 2, 4, 7 (les poids permettent de déterminer la valeur décimale directement). Ce codage est ensuite complété par un bit de contrôle e permettant de vérifier la condition soulignée ci-dessus. Par exemple, pour définir 2, on place un 1 directement au niveau du bit de poids 2 (2 en décimal) puis pour respecter la condition on ajoute un 1 au niveau du bit de contrôle.

Q3. Déterminer les codes des chiffres de 4 à 9 en complétant la partie grisée du tableau ci-dessous. En déduire le code du chiffre 0 pour assurer l'unicité du code.

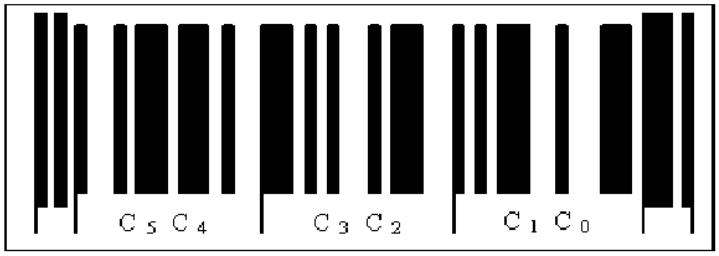
Chiffre	poids	1	2	4	7		$2^3 = 8$	$2^2 = 2$	$2^1 = 2$	$2^0 = 1$
décimal	bit	a	b	c	d	e	S_3	S_2	S_1	S_0
0										
1		1	0	0	0	1				
2		0	1	0	0	1				
3		1	1	0	0	0				
4										
5							0	1	0	1
6										
7										
8										
9										

Les chiffres de rang impair (c'est-à-dire C_5, C_3 et C_1) sont codés sur les barres noires et les chiffres de rang pair (c'est-à-dire C_4, C_2 et C_0) sont codés sur les espaces entre les barres noires :

— les 1 sont codés par les barres ou les espaces larges en utilisant deux largeurs de base,

— les 0 sont codés par les barres ou les espaces étroits en utilisant seulement une largeur de base.

Le code à barres débute par une entête de 2 barres et de 2 espaces étroits et se clôture par une barre large, un espace puis une barre étroite. Ce dispositif rend ainsi la lecture de ce code possible dans les deux sens. Le premier bit lu de chaque chiffre est le bit a.



Q4. Déterminer alors le nombre à 6 chiffres correspondant à l'identité du titulaire de la carte de la figure ci-dessus.

Le calculateur traduit chacun des 6 chiffres de ce code à barres en un nombre binaire naturel codé sur 4 bits (S_3, S_2, S_1, S_0), le poids du bit S_i valant naturellement 2^i : ainsi $S_3S_2S_1S_0 = 1101$ code le nombre 13.

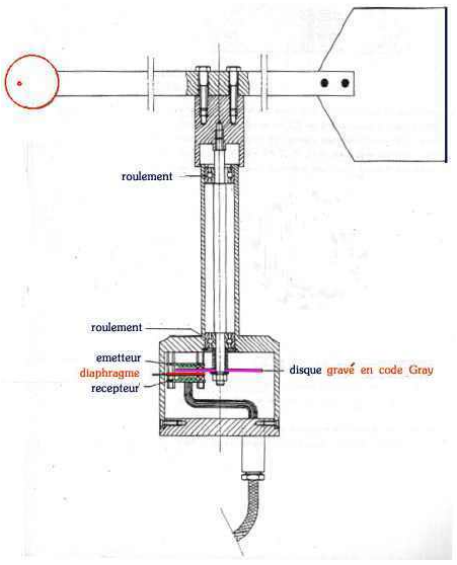
Q5. Établir la table de vérité des sorties S_i en fonction des entrées a, b, c, et d en complétant le tableau précédent.

Q6. Donner l'expression de S_1 en fonction de a, b, c et d.

Q7. Tracer le logigramme de S_1 en utilisant uniquement des opérateurs Non, Ou et Et.

Exercice 3 : girouette éléctronique

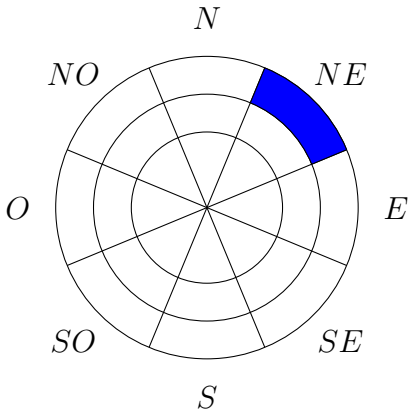
La mesure de la direction du vent constitue une donnée très importante en microclimatologie. La grande majorité des dispositifs sont basés sur le principe de la girouette, principe auquel on a ajouté un dispositif électronique de mesure de la position angulaire. Ce dernier peut-être analogique, il s'agit alors d'un simple potentiomètre dont le curseur est aligné avec la direction de la girouette.



Une autre solution consiste en la réalisation d'un codeur optique associé à l'axe pivotant de la girouette. Dans ce cas la résolution dépendra de celle du codeur. Avec un codeur à N pistes on pourra différencier 2^N directions de vent.

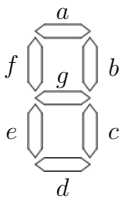
Le principe est le suivant sur une plaque émettrice on dispose un alignement de N diodes infrarouges et, en face d'elles, un jeu de N récepteurs qui reçoivent le faisceau IR s'il n'est pas arrêté. On dispose entre émetteurs et récepteurs un disque solidaire de l'axe de la girouette et comportant des zones opaques gravées en **code gray**, ce qui permet d'identifier numériquement la position angulaire de la girouette. Une zone opaque correspond à un 1 tandis qu'une zone transparente correspond à 0.

On s'intéresse à une girouette permettant de repérer 8 directions indiquées N, NE, SE, S, SO, O, NO et représentées ci-dessous.



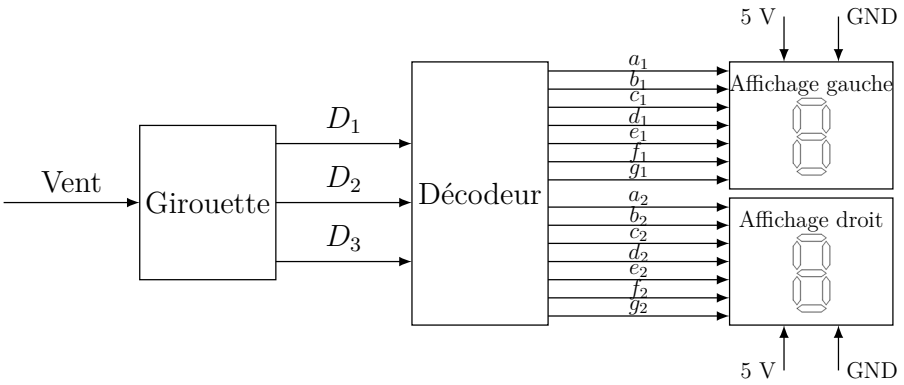
Q8. Justifier pourquoi 3 diodes photoélectriques sont utilisées.

Q9. Compléter le disque en noircissant les différentes cases selon le principe du code Gray sachant que les positions N et NE sont déjà complétées. Quel est l'intérêt d'utiliser un code Gray plutôt qu'un code binaire naturel ?

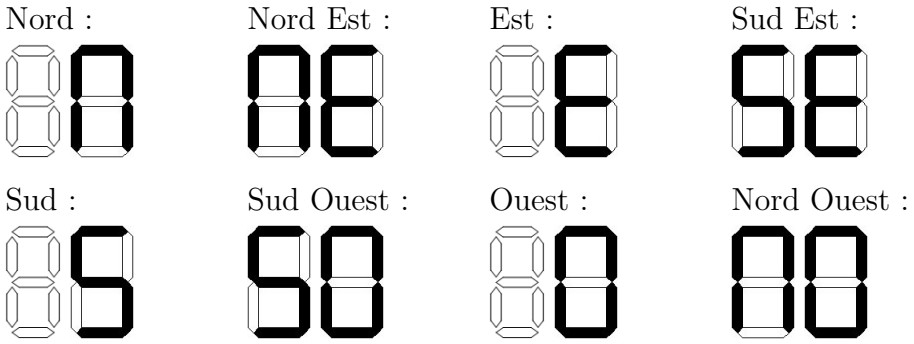


On souhaite maintenant afficher la position de la girouette sur deux afficheurs en fonction de la direction du vent mesurée. On utilise pour cela un décodeur qui convertit les informations transmises par les 3 diodes en signaux de commande des segments de chaque afficheur.

L'objectif est donc de déterminer le décodeur qui sert d'interface entre la mesure de la direction et l'affichage.



On désire afficher les informations suivantes :



Q10. Compléter le tableau en associant les états des photo-diodes D_1, D_2, D_3 et la direction sachant que la diode D_1 est utilisée sur la piste en périphérie, la diode D_2 pour la piste centrale et la diode D_3 pour la piste intérieure

D_3	D_2	D_1		a_1	b_1	c_1	d_1	e_1	f_1	g_1	a_2	b_2	c_2	d_2	e_2	f_2	g_2
			N														
			NE														
			E														
			SE														
			S														
			SO														
			O														
			NO														

Q11. Compléter le tableau reliant les segments d'un afficheur aux lettres à afficher.

Q12. Donner les expressions de a_1, b_1, d_1 en fonction de D_3, D_2, D_1 et les simplifier en utilisant les symboles de OU-exclusif et ET-exclusif.

Q13. Donner les expressions de b_2, d_2, e_2, g_2 et les simplifier en utilisant les lois de De Morgan.

TD 3 – Systèmes à logique séquentielle

Panneau solaire asservi

Le rendement d'un panneau solaire peut être sensiblement amélioré en le plaçant sur un support orientable et motorisé, de façon à rester face au soleil. Un capteur fixé sur le panneau, constitué d'un cache et de deux photo-résistances montées en pont diviseur de tension, fournit une tension variable en fonction de l'orientation du panneau face au soleil (voir figure 1). Lorsque les deux résistances sont également éclairées, le capteur renvoie 2,5 V, et lorsqu'une des résistances est à l'ombre du cache, le capteur renvoie une valeur de 1 V ou 4 V selon la résistance à l'ombre.

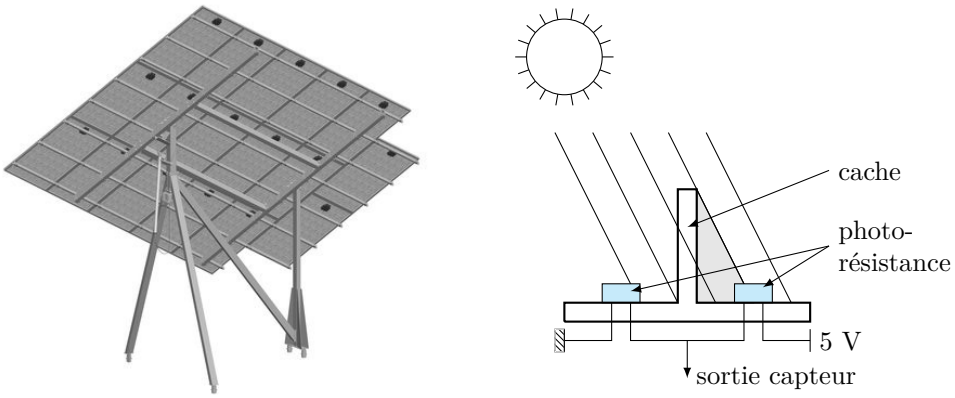


FIGURE 1 : Panneau solaire orientable et représentation du capteur d'orientation.

Le système de commande doit positionner le panneau solaire de façon à maintenir le panneau face au soleil. Un microcontrôleur permet de piloter le panneau solaire. La commande de mouvement se fait par deux consignes logiques $M+$ et $M-$ envoyées en entrée de l'interface de puissance du moteur.
Deux capteurs de fin de course $C+$ et $C-$ détectent les positions extrêmes permises pour le mouvement du panneau solaire.
On note $seuil-$ la tension minimale mesurée sur le capteur lumineux à partir de laquelle le moteur doit tourner ($M+$). On note $seuil+$ la tension maximale à partir de laquelle le moteur doit tourner dans l'autre sens ($M-$).

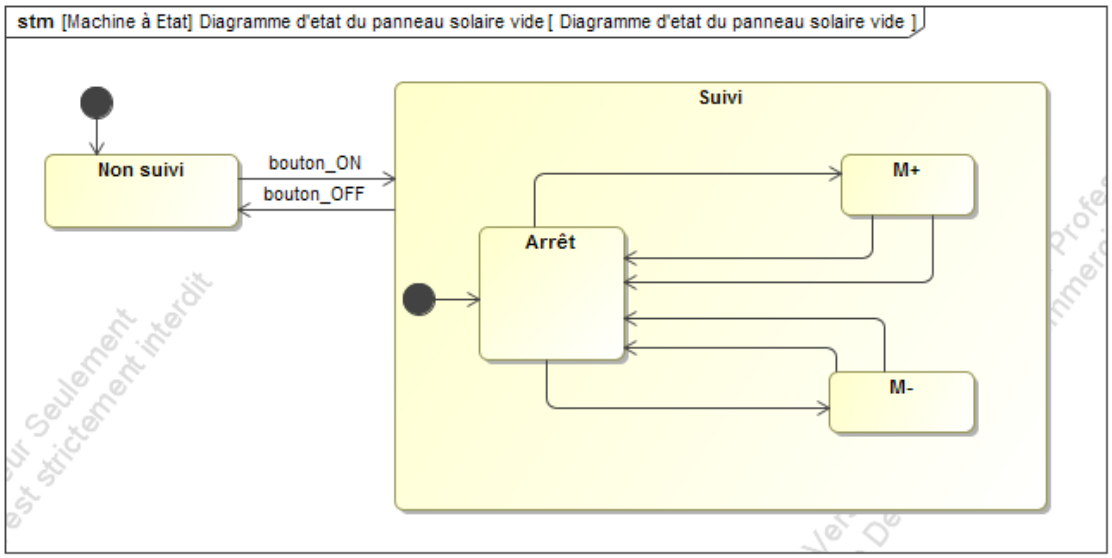
Le cahier des charges est le suivant :

- Un bouton poussoir mono-stable permet, sur le boîtier de commande, d'activer ou désactiver le suivi du soleil par une pression simple.
- En phase de suivi, une LED s'allume sur le boîtier de commande. Elle est éteinte en phase d'arrêt.
- Lorsque la tension capteur lux passe sous un seuil limite $seuil-$, le moteur tourne dans le sens $M+$ pendant au moins 2 secondes pour éviter les instabilités au voisinage du seuil.
- Lorsque la tension capteur lux passe au dessus d'un seuil limite $seuil+$, le moteur tourne dans le sens $M-$ pendant au moins 2 secondes.

— Le moteur doit s'arrêter si une fin de course est rencontrée. Il peut néanmoins revenir en arrière.

Q1. Déterminer la liste des entrées et sorties du micro-contrôleur ainsi que leur nature (logique ou analogique).

Q2. Compléter le diagramme d'états spécifiant le comportement du panneau solaire en renseignant les événements de l'état composite Suivi.



Le programme implanté dans le micro-contrôleur boucle en permanence pour savoir dans quel mode il se trouve et à l'intérieur du mode Suivi ce que le panneau doit faire. En Python le programme principal serait le suivant :

```
while true :
    if bouton==1 :
        led=1
        suivi()
    else :
        led=0
        non_suivi()
```

On s'intéresse uniquement au mode Suivi. La lecture du niveau lumineux se fait par la commande `lux.read()` qui renvoie la tension mesurée sur le capteur.

Q3. Proposer un algorithme en Python correspondant au mode Suivi. On utilisera les noms des variables en les mettant à 1 ou 0 (exemple $M+ = 1$ correspond au fonctionnement du moteur dans le sens +). La lecture du temps actuel est faite par la commande `time.time()`.

Gestion de l'avance papier d'une imprimante

On s'intéresse à la gestion de l'avance du papier dans le cas d'une imprimante jet d'encre. Celle-ci possède un moteur associé à un codeur incrémental précis qui met en rotation un rouleau dans un sens ou dans l'autre. Un ensemble d'engrenages débrayables permet de prendre une feuille et la déplacer jusqu'à la tête d'impression ou de la maintenir et de la faire avancer pendant l'impression.

Le fonctionnement normal de l'avance papier est le suivant.

Lorsque l'impression est lancée, le moteur tourne dans un sens ($M++$) pour faire avancer rapidement la feuille jusqu'à un capteur de présence papier (noté *capt*). La tête d'impression est alors mise en position initiale, le moteur est arrêté pendant ce temps. Puis le moteur tourne dans l'autre sens (la feuille avance toujours !) à vitesse lente ($M-$) par a-coups à chaque fois qu'une ligne d'impression est réalisée. Le déplacement fait par la feuille est spécifié par la résolution d'impression choisie au départ par l'utilisateur. On note da le déplacement à faire et d le déplacement actuel. Lorsque l'impression est terminée, la feuille sort à vitesse rapide ($M--$).

Si aucune feuille n'arrive au capteur de papier pendant 3 s, alors un message d'erreur est envoyé pour avertir qu'il n'y a pas de papier.

Si le moteur force trop (blocage de feuille ou bourrage), un message d'erreur est également envoyé. La détection du blocage est faite en mesurant l'intensité dans le moteur qui ne doit pas être supérieure à une valeur maximale donnée $Imax$.

Un diagramme d'état est adapté à la description du fonctionnement de l'imprimante compte-tenu de la spécification indiquée précédemment.

Q4. Indiquer les différents états dans lesquels se trouve l'avance papier.

Q5. Proposer un diagramme d'états permettant de spécifier le comportement de l'avance papier.

Digicode

Les digicodes sont très répandus pour contrôler l'accès aux immeubles . Ils permettent de filtrer les entrées sans nécessiter de clés. Il suffit de taper une suite de 4 chiffres ou lettres correspondant au code mémorisé pour que la porte soit déverrouillée.

L'objectif est de réaliser le programme de commande du digicode, dont le cahier des charges est le suivant :

- Le digicode envoie un signal logique à 1 en sortie dès qu'une suite de 4 caractères correspondant au code juste est tapé.
- La porte est déverrouillée pendant 10 s, le temps que la personne ouvre la porte, puis se verrouille à nouveau.
- Le code est inscrit dans le programme : la façon de le modifier n'entre pas dans le cadre de cet exercice.

À chaque lecture du digicode, la touche lue est stockée dans un tableau T de 4 caractères contenant les 4 dernières touches lues. Ce tableau est comparé au code C pour décider de l'ouverture ou non de la porte. On utilise un compteur N initialisé à 0 pour compter les 4 caractères tapés.

Un diagramme d'activité est utile pour spécifier l'ensemble des tâches effectuées par le digicode. **Q6.** Proposer un tel diagramme en utilisant les variables N, C, T ainsi que les actions lecture d'une touche, ouverture, verrouillage.

D'un point de vue fonctionnement, le digicode est constitué d'une matrice de 12 touches et d'un réseau de fils. Sous chacune des touches se situe un interrupteur normalement ouvert, permettant lors d'une pression de connecter deux fils du réseau.

Les fils sont reliés à 7 entrées logiques du micro-contrôleur. Trois d'entre eux présentent une résistance de tirage (dite pull-up resistance, en anglais) permettant de maintenir par défaut le niveau logique à 1, lorsqu'aucun autre élément n'impose le niveau à 0.

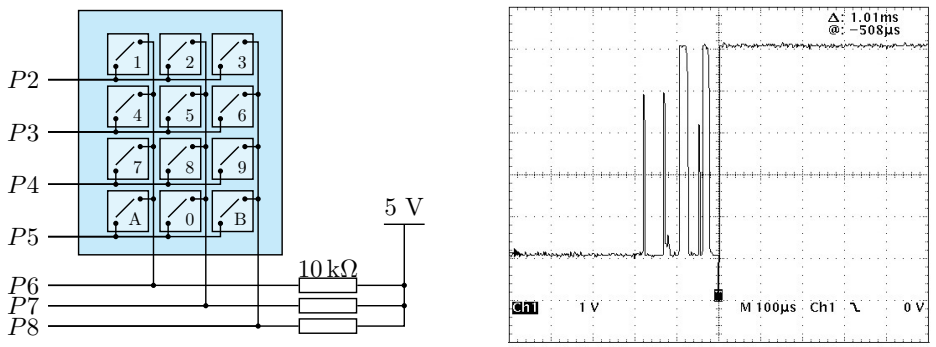


FIGURE 2 : Schéma de principe du digicode ; illustration du phénomène de rebonds.

La lecture du clavier se fait ligne par ligne, en analysant les états des colonnes : les lignes ($P2$ à $P5$) sont toutes mises à 1 sauf la première, mise à 0. Si, lors de la lecture d'une colonne ($P6$ à $P8$), un niveau logique 0 est détecté, cela signifie qu'une touche de la première ligne est appuyée dans la colonne correspondante. Les lignes suivantes sont lues de la même façon en quelques micro-secondes.

Les interrupteurs présentent un défaut bien connu en électronique : le signal présente durant quelques centaines de micro-secondes des rebonds de basculement de 0 à 1. Pour éviter de lire une information erronée à partir de la détection d'un changement d'état, il est nécessaire d'attendre environ 5 milli-secondes que le signal soit stabilisé avant de lire le clavier.

Q7. Répartir les variables $P2$ à $P8$ en entrées-sorties pour le micro-contrôleur.

On introduit un tableau représentant l'ensemble des touches réparties par lignes :

```
Liste_touches=[["1","2","3"],["4","5","6"],["7","8","9"],["A","0","B"]]
```

La touche lue est stockée dans une variables `Touche` initialisée à `N`.

Pour lire une variable, on utilise la commande `Pi==1` ou `Pi==0`. Pour imposer une variable à 1 ou 0, on utilise la commande `Pi=1` ou `Pi=0`.

Q8. Proposer un programme Python de la fonction lecture en utilisant les variables $P2$ à $P8$ et celles introduites ci-dessus.