1. Traitement de l’information

AGIR

*Chaîne d’information*

* 1. La fonction TRAITER :

Les informations arrivent en **ENTREES** (fonction **ACQUERIR** de la chaîne d’information) :

Enfin, en **SORTIES** (fonction **COMMUNIQUER** de la chaîne d’information) :

De nos jours, dans la plupart des cas, les différentes tâches de la fonction TRAITER sont réalisées par un constituant programmé.

* 1. Organisation fonctionnelle simplifiée d'un système micro programmé.



Mémoire de programme :

Mémoire de données :

Horloge :

Microprocesseur :

Interface d’entrée / sortie :

Pour organiser et permettre la communication entre tous ces éléments, on trouve des bus. Un bus est un ensemble de conducteurs qui transportent en même temps des informations binaires pour constituer un code numérique

Le bus de donnée :

Le bus d’adresse :

Le bus de contrôle :

Informations supplémentaires :

Un micro-processeur 64 bits signifie que le micro-processeur possède un bus de données de 64 bits (64 conducteurs). Il peut donc traiter des informations utilisant jusqu’à 64 bits. Nous n’utiliserons pendant l’année que des microprocesseurs ou microcontrôleurs de 8 bits.

* 1. Nature des informations traitées.

**1**

**t**

**0**

***Bouton A relâché***

***Bouton A appuyé***

***Info BP A***

***Information issue du BP A***

***BP A***

Il arrive que les informations varient de façon continue au cours du temps. Ces informations sont de nature analogique. Elles doivent être acquises par une interface d'entrée particulière permettant de les convertir (transformer) en des informations de nature numérique.

C'est le cas d'un son (vibration de nature analogique) enregistré dans un smartphone (l'information stockée est de nature numérique).

1. La programmation
	1. Présentation :

Pour obtenir le fonctionnement souhaité, il faut exécuter des opérations en fonction des informations acquises et envoyer des ordres. Ceci est décrit dans un programme préalablement placé dans le système micro programmé

* 1. L’algorigramme :

Principaux symboles :

* 1. Le langage Arduino :

La carte Arduino repose sur un circuit intégré (un mini-ordinateur appelé également microcontrôleur) associée à des entrées et sorties qui permettent à l'utilisateur de brancher différents types d'éléments externes.

Le programme est lu par le micro-contrôleur de haut vers le bas. Une variable doit être déclarée avant d'être utilisée par une fonction. La structure minimale est constituée :

Exemples de structure algorithmique :

|  |  |  |
| --- | --- | --- |
| **Algorithmique** | **Algorigramme** | **Langage Arduino** |
| - **Si** condition vraie  - Faire opération 1  **Sinon**  - Faire opération 2 - **Fin si** | **C:\Users\Laokali\Desktop\Lycée Gustave Eiffel\2015-2016\Cours\S_SI_Traitement_et_programmation\Algo_if.PNG** |  |
| - **Tant que** condition vraie  - Faire opération 1 - **Fin Tant que** | **C:\Users\Laokali\Desktop\Lycée Gustave Eiffel\2015-2016\Cours\S_SI_Traitement_et_programmation\Algo_tantque.PNG** |  |
| - Faire opération 1- **Tant que** condition vraie  - Faire opération 1 - **Fin Tant que** |  |  |
| - **Pour** i de I1 à I2  - Faire opération 1  - **Fin pour** | C:\Users\Laokali\Desktop\Lycée Gustave Eiffel\2015-2016\Cours\S_SI_Traitement_et_programmation\Algo_for.PNG |  |

Exemple de commandes en langage Arduino

Structure générale :

* void setup() (configuration-préparation)
* void loop() (exécution)

Opération de comparaison (à utiliser dans les conditions) :

* == (équivalent à)
* != (différent de)
* < (inférieur à)
* > (supérieur à)
* <= (inférieur ou égal à)
* >= (supérieur ou égal à)

Contrôle et condition :

* if (si...)
* if...else (si...alors...)
* for (pour...)
* switch case (dans le cas où...)
* while (pendant que ...)

Niveaux logiques des connecteurs numériques :

* HIGH (état 1)
* LOW (état 0)
* INPUT (configuré en entrée)
* OUTPUT (configuré en sortie)

Types de variables :

* char (variable ‘caractère’)
* int (variable ‘nombre entier’)
* long (variable ‘nombre entier de tès grande taille’)
* string (variable ‘chaine de caractères’)
* array (tableau de variables)

Les fonctions usuelles :

* pinMode(**broche**, **mode**) - Configure la **broche** spécifiée pour qu'elle se comporte soit en **entrée (INPUT)**, soit en **sortie (OUTPUT)**.
* digitalRead(**broche**) - Lit l’état (= le niveau logique) d'une **broche** précisée en entrée numérique, et renvoie la valeur **HIGH** (HAUT en anglais) ou **LOW** (BAS en anglais).
* digitalWrite(**broche**, **valeur**) - Met un niveau logique **HIGH** ou **LOW** sur une **broche** numérique.
* delay(**ms**) - Réalise une pause de **ms** millisecondes dans l'exécution du programme.
1. Exercices

Exercice 1 :

**Comprendre l’algorithme suivant et écrire son algorigramme :**

|  |  |
| --- | --- |
|  Sortie moteur à 1Et Lire Entrée capteurTant que Entrée capteur est à 0 Sortie moteur à 1 Et Lire Entrée capteurFin tant queFaire Sortie moteur à 0 | Sortie moteur (S\_mot) : pilote un moteur d’un robot roulant. Lorsqu’il est à 1 se met en marche.Lorsqu’il est à 0 se met en arrêt.Entrée capteur (E\_cap) : Détecte la présence d’un obstacle devant le robot.Lorsqu’il est à 0, aucun obstacle n’est présent.Lorsqu’il est à 1, un obstacle est détecté. |

Exercice 2 :

Le robot a cette fois-ci une sortie moteur pour le piloté en avançant (S\_mot\_AV), une sortie moteur pour le piloté en reculant (S\_mot\_AR), un capteur de détection d’obstacle à l’avant (E\_cap\_AV) et un capteur de détection d’obstacle à l’arrière (E\_cap\_AR).

Scénario :

Au début le robot avance. (S\_mot\_AV = 1)

S’il détecte un obstacle devant lui (E\_cap\_AV = 1), il arrête d’avancer (S\_mot\_AV = 0) et se met à reculer (S\_mot\_AR = 1)

S’il détecte un obstacle derrière lui (E\_cap\_AR = 1), il arrête de reculer (S\_mot\_AR = 0) et recommence à avancer (on revient au début du programme).

**Ecrire le programme en algorigramme.**