Illustration de l'algorithmique avec le robot Thymio

Intérêt pédagogique du robot Thymio.

- L'interface graphique VPL permet une programmation intuitive sans l'apprentissage préalable d'un langage. Dès la première séance les élèves peuvent programmer.
- L'apprentissage du code se fait en comparant ce dernier au comportement du robot et à l'idée qu'on en avait.
- L'intérêt du robot est de permettre aux élèves d'observer directement les effets de leur code ou d'une modification de ce dernier. Il peuvent également observer en temps réel les valeurs des variables.

L'idée est d'utiliser le robot pour illustrer des bases de la programmation :

Variables
Entrée/sortie
Types de variables
Choix
Boucles
Procédures.

Pris en main avec « VPL »

Le langage de programmation est basé sur des couples « événement-action ». L'interface « VPL » permet de créer une série de couples de façon intuitive en faisant glisser des icônes. Toute la documentation est disponible sur le site :

https://aseba.wikidot.com/fr:start

Faire avancer le robot 

A ce couple correspond le code :

```
onevent buttons                 //      Dès qu'un bouton renvoie une valeur
    motor.left.target = 200     //      vitesse du moteur gauche = 200
    motor.right.target = 200    //      vitesse du moteur droit = 200
```

Le code ne précise pas de valeur attendue pour les variables « button ». On constate en revanche qu'il renvoit une valeur puisque le robot avance.

En cochant la case « auto » de la colonne de gauche on observe les valeurs des variables en temps réel. Ce sont des valeurs entières signées ou non. On observe que les variables « button » ne

peuvent prendre que deux valeurs (0 ou 1). Ceci nous permet d'introduire la notion de booléen.

Notion de choix

On veut que le départ du robot soit commandé une pression sur le bouton avant.

```
onevent buttons                     //    Dès qu'un bouton renvoie une valeur
    if button.forward  == 1 then    //    Si la valeur du bouton avant est 1 alors
        motor.left.target = 200     //    vitesse du moteur gauche = 200
        motor.right.target = 200    //    vitesse du moteur droit = 200
    end                             //    Fin de la condition
```

Différence entre affectation « = » et comparaison « == »

« button.forward » variable d'entrée, « motor.right.target » et « motor.left.target » variables de sortie .

L'avancée du robot se poursuit bien que la variable « button.forward » soit revenue à 0. Le code a juste un effet déclencheur.

Applications

Diriger le robot avec les boutons
Utiliser « else » et « elseif »
Introduire une variable vitesse

Notion d'itération

On utilise l'événement « timer » qui génère un signal périodique.

```
var iteration = 0                   //    Déclaration et initialisation de la variable « iteration »
timer.period[0] = 1000              //    Définition d'une période de 1000 ms
onevent timer0                      //    A chaque période
    iteration = iteration + 1       //    La valeur de la variable « iteration » augmente de 1
```

On observe dans la fenêtre « Aseba Studio » l'incrémentation chaque seconde de la variable « itération ».

Applications

Faire clignoter le robot (chaque seconde).
Faire avancer le robot durant un temps déterminé.

Notion de procédure (sous routine)

Ouvrir et exécuter le fichier « NotionProcedure1 » *(On peut visualiser le tracé en plaçant un crayon dans l'orifice prévu à cet effet).*

Créer et utiliser les sous routines :
Avance
Tourne_A_ Gauche
Tourne_A_ droite

Puis créer une sous routine « Motif » qui reprend le tracé.

Notion de boucle

Ouvrir et exécuter le fichier « NotionProcedure2 »
Placer un crayon sur le robot.

Exécuter le programme et observer le tracé.
Décommenter la ligne « itera = 0 ».
Exécuter à nouveau le programme et observer le tracé.
On constate que le motif se répète indéfiniment. Le programme crée une boucle *(morceau de code qui se répète)*.

Boucle for
Le nombre d'itérations *(exécution de la boucle)* est déterminé préalablement. On incrémente un compteur jusqu'à ce qu'il atteigne le nombre d'itérations souhaitées.

Boucle while
Le nombre d'itérations est indéterminé au départ. Il dépend d'une condition annexe par exemple la détection d'un objet.

Le langage Aseba intègre des fonctions boucle « while » et boucle « for ». Cependant, il n'est pas possible d'intégrer une temporisation à l'intérieur d'une de ces boucles. Pour illustrer ces concepts à l'aide du robot il faut passer par des procédures.

Application

Créer une procédure boucle « while » et une procédure boucle « for ».