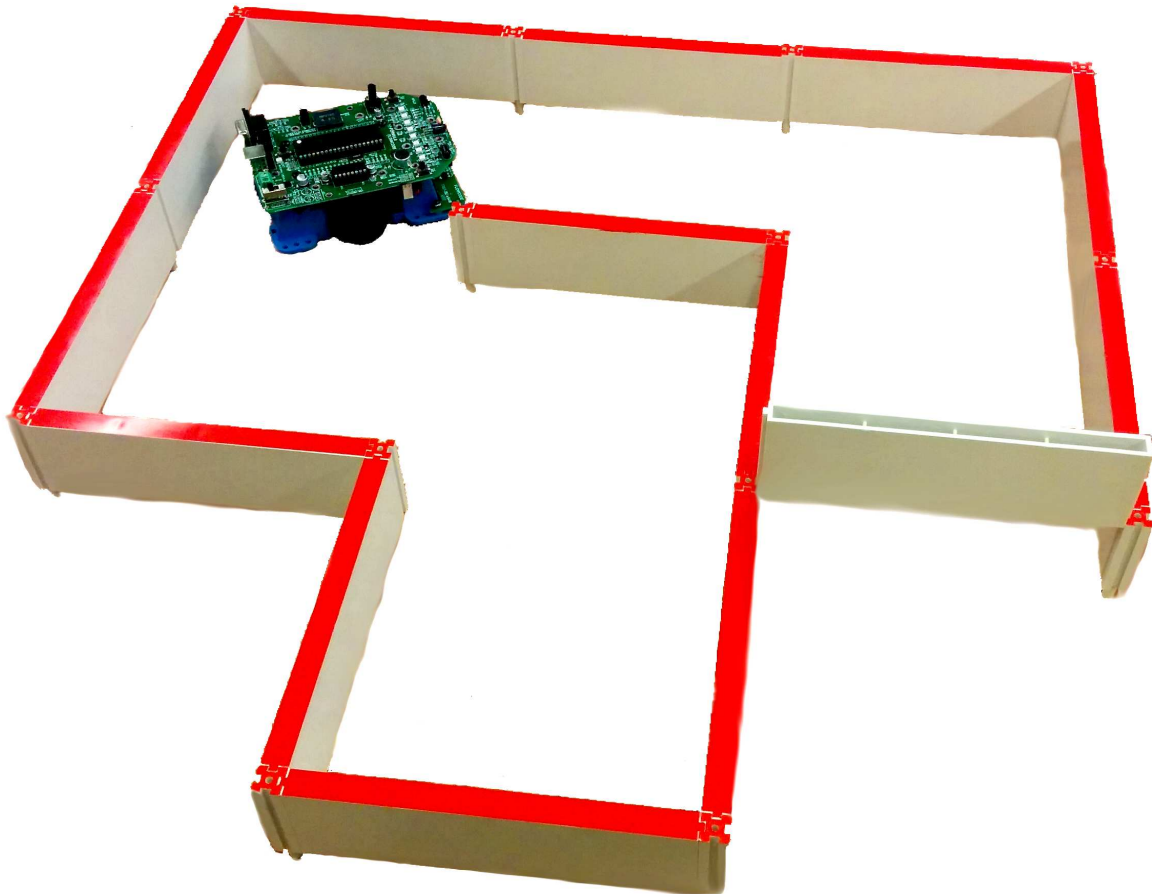


## TP

### **Buggy FORMULA FLOWCODE** **Simulation dans FLOWCODE V6** Sortir d'un labyrinthe simple



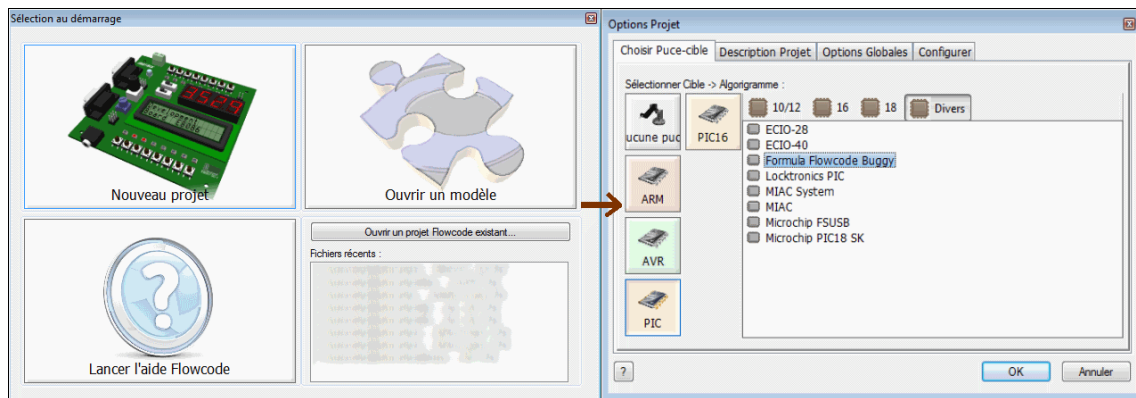
L'objectif de ce TP est de créer un programme qui permettra au robot Formula Flowcode de sortir d'un labyrinthe sans intersection ni impasse dans l'interface de simulation.

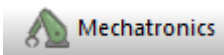
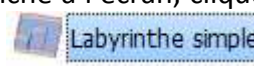
## **Matériel demandé :**

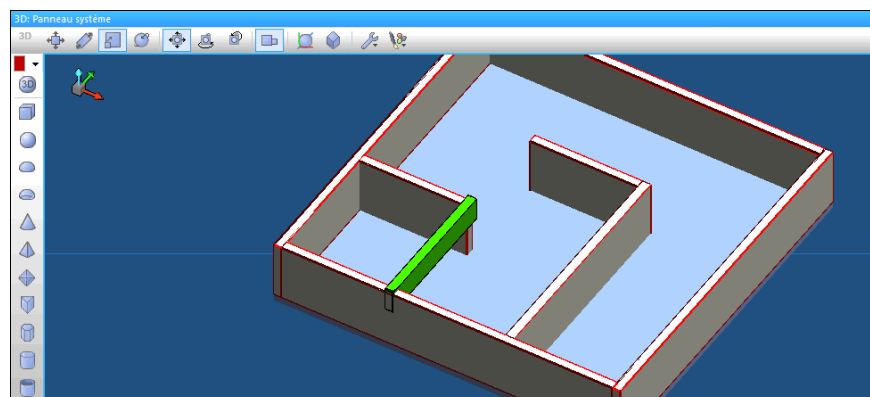
1 PC avec Flowcode V6 pour PIC


## **I. Création du projet**

- 1) Rendez vous dans le dossier <C:\Program Files\Flowcode 6\components> pour y déposer le fichier « laby\_simple.fcp » qui correspond à notre labyrinthe.
- 2) Lancez Flowcode V6 et créez un nouveau projet pour le Formula Flowcode.

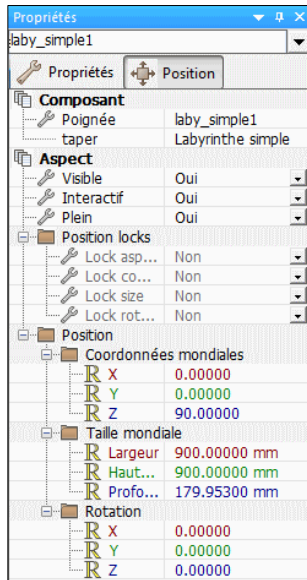


- 3) Une fois le panneau affiché à l'écran, cliquez sur  puis, dans le menu déroulant, sélectionnez  pour ajouter le labyrinthe au projet.
- 4) L'élément suivant apparaît donc sur le panneau (Si vous ne voyez rien, essayez de dé-zoomer en appuyant sur la touche « Ctrl » et en ajustant le niveau de zoom avec la molette souris) :



- 5) Cliquez sur le labyrinthe. Dans la partie de gauche de l'écran, vous retrouvez les propriétés du composant. Cliquez sur  **Position**.



- 6) Réglez les paramètres de position comme suit :



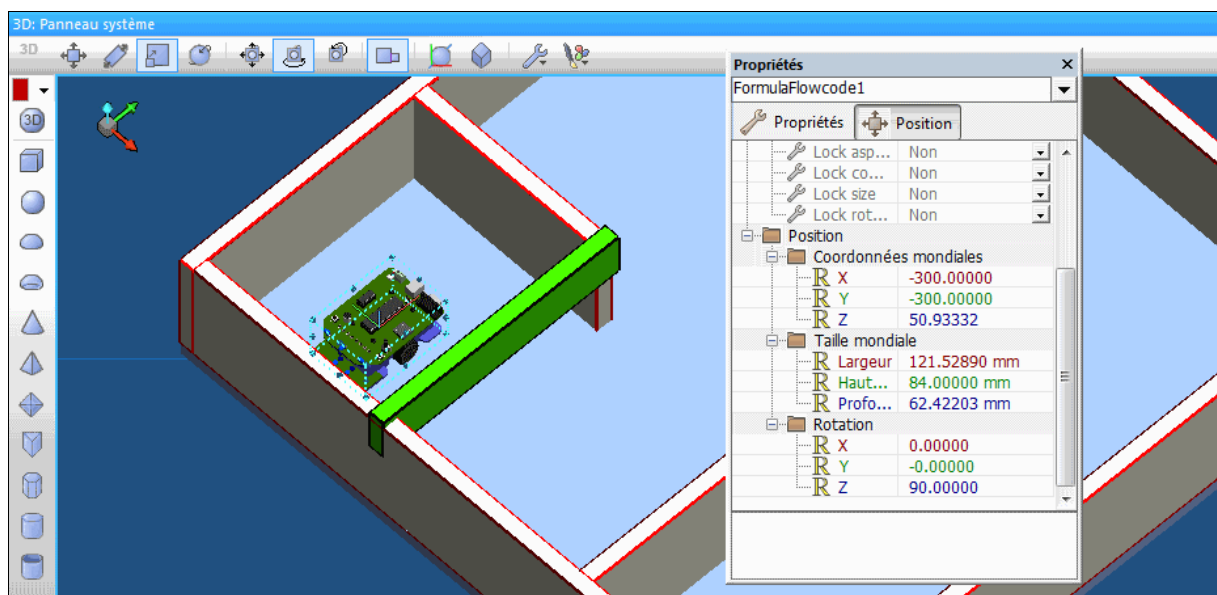
*Remarque : Ici nous positionnons le labyrinthe au centre du repère et à la surface du plan (x,y). D'où le « Z=90 ».*

*Dans Flowcode V6, le positionnement se fait par rapport au centre de l'objet. Si nous voulons positionner le dessous du labyrinthe sur le plan (x,y), il faut positionner le centre à  $Z = P/2$  mm. Avec P la profondeur soit  $180 / 2 = 90$  mm.*

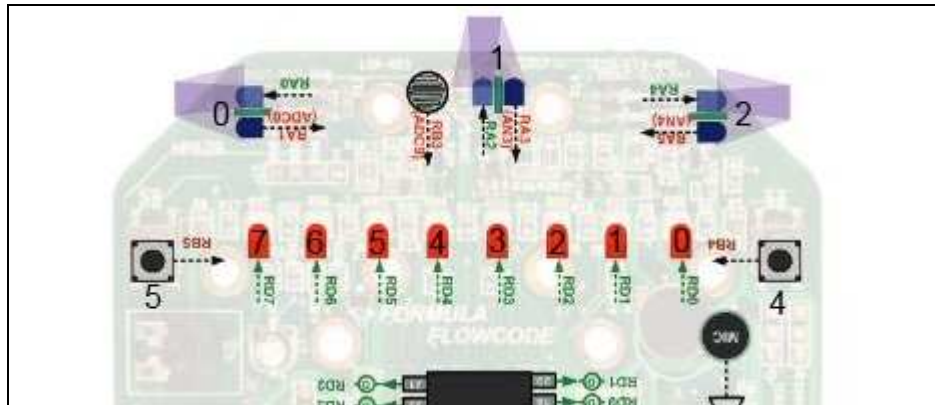
*La compréhension de cette notion n'est pas indispensable. Elle nous sert juste à positionner avec précision les éléments dans la visualisation 3D et à bien faire interagir le robot avec les murs du labyrinthe.*

- 7) Cliquez de nouveau sur  **Mechatronics** puis sélectionnez  **Formula Flowcode**.

- 8) Le Formula devrait apparaître en plein centre du labyrinthe. Sélectionnez le et configurez ses paramètres de position comme ci-dessous. Le robot doit se trouver dans la « case » de départ et face à un mur :

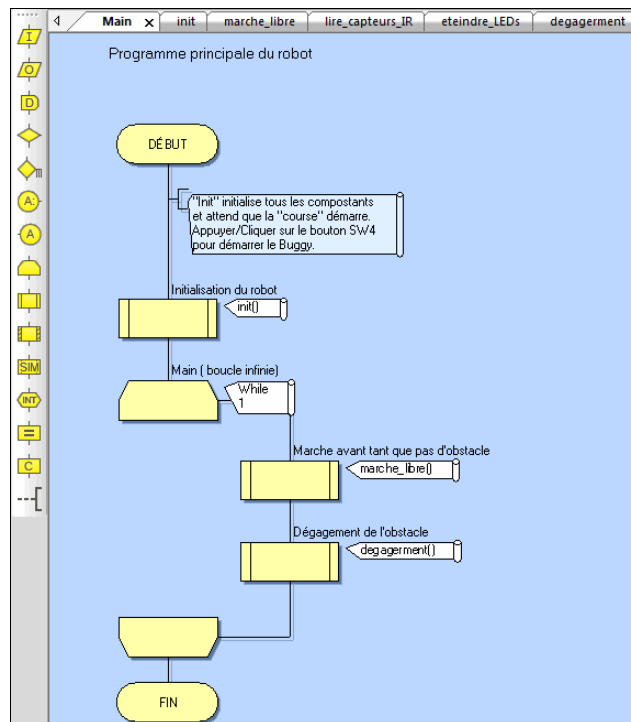


9) A savoir : Pour la suite du TP, il est important de garder en tête ce schéma donnant les identifiants des composants pour Flowcode:



## II. Structure du projet et création de « macros »

### 1) Le programme final :



Vous remarquez que le programme principal est très court. C'est normal ! Nous avons fractionné le code (algorithme) pour faciliter la lecture.





Ici, nous initialisons le robot puis nous exécutons en boucle les macros `marche_libre()` et `degagement()`. Ces macros (aussi appelées fonctions) sont des fractions de code que l'on peut « appeler » à tout moment. Dans Flowcode, il y a deux types de macros :



**Les macros standard**, celles intégrées dans le logiciel. On les utilise pour commander des composants. Ici notre robot.



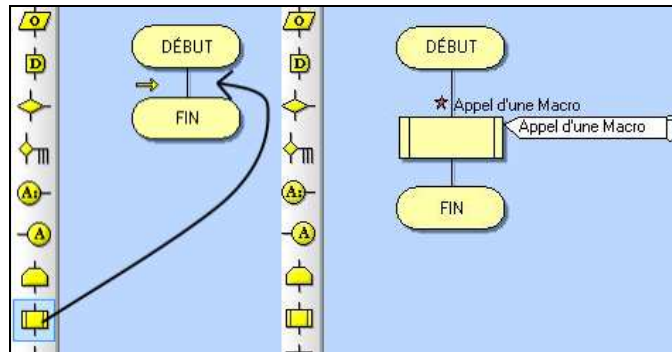
**Les macros personnelles**. Elles sont créées manuellement par l'utilisateur.


-  Une macro peut contenir d'autres macros sans limite.
-  Dans ce TP, les macros personnelles que nous utiliserons seront simples. Pas de valeur renvoyée ni de paramètre à fournir.
-  De plus, « main » est la macro principale ou « mère ».
-  Dans les explications de ce TP, les macros personnelles seront toujours suivies de parenthèses et écrites en bleu, les conditions seront entre accolades et les variables seront précédées de « \$ ». Ceci n'est pas à reproduire dans le programme.

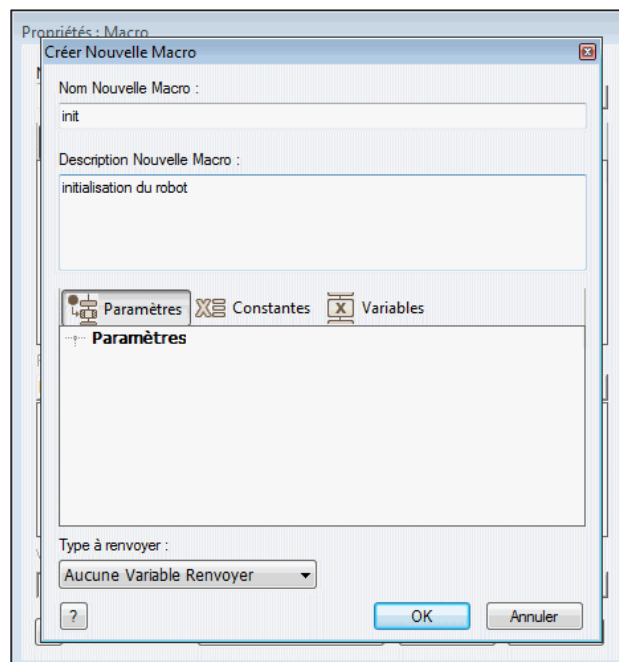
## 2) Création d'une macro

Nous allons dès à présent apprendre à déclarer une macro pour l'utiliser ensuite.

- Insérez une macro personnelle dans votre algorithme en la glissant déposant depuis la barre latérale gauche.



- Double cliquez dessus et double cliquez sur  **Macro**. Une fenêtre apparaît. Donnez un nom à votre macro, ici « init ». (la description est facultative mais recommandée pour la lisibilité du programme)  
Enfin, cliquez sur « OK ».



- Faites de même pour les macros `degagement()`, `marche_libre()`, `eteindre_LEDs()` et `lire_capteurs_IR()`. Les noms de macro sont choisis arbitrairement. Veillez à n'utiliser que les lettres A à Z, a à z. Pas d'espace ni d'accent autorisé. Retournez dans `main()`.

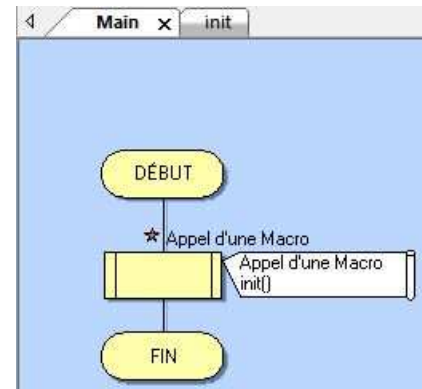
## III. Ecriture du code

 Le symbole  apparaît sur un élément quand une modification n'a pas été enregistrée. (CTRL + S pour enregistrer)

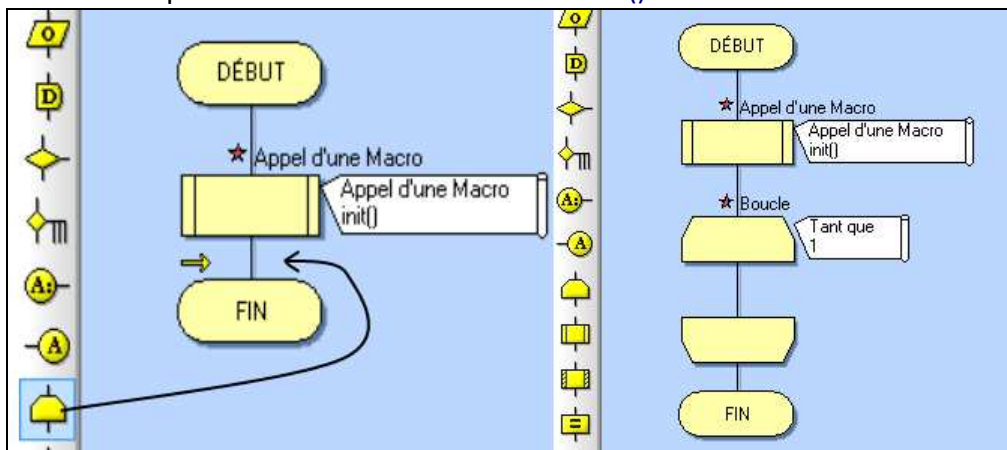
### 1) Remplissage du « main » (programme principal) :

a. Dans l'algorithme, double cliquez sur votre macro et sélectionnez « init » (surligné en bleu). Cliquez sur « OK & Editer Macro ». La macro `init()` s'ouvre dans un nouvel onglet, revenez dans `main()`.

Voici le résultat --->



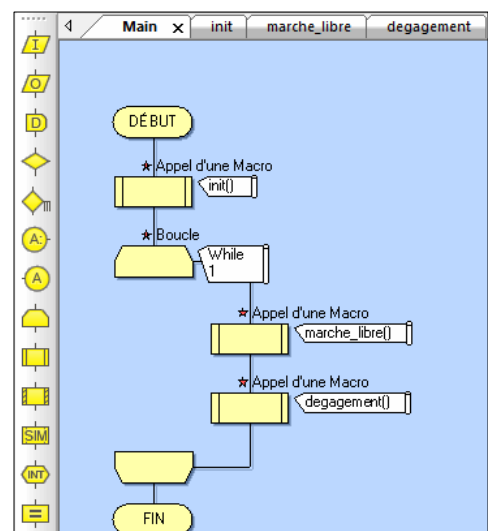
b. Glissez déposez une boucle sous la macro `init()`.



La boucle indique « Tant que 1 », ceci signifie que la boucle s'effectuera tant que  $1=1$ . Or ceci est toujours vrai donc la boucle est infinie.

c. Glissez déposez deux macros dans la boucle. La première sera `marche_libre()` et la seconde sera `degagement()`. Les deux devraient s'ouvrir dans un onglet.

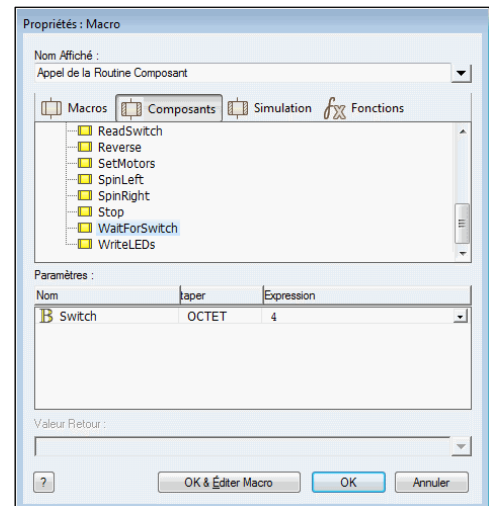
d. Notre `main()` est maintenant terminé --->



## 2) Remplissage de init() :

- Cliquez sur l'onglet « init ». La macro est vide.
- Glissez déposez trois macros standard.
- Double cliquez sur la première. Une fenêtre apparaît. Déroulez la liste « FormulaFlowcode1 » puis sélectionnez « Initialise ». Cliquez sur Ok.
- Pour la macro suivante, sélectionnez « WaitForSwitch ». Dans le cadre « Expression », inscrivez « 4 ».

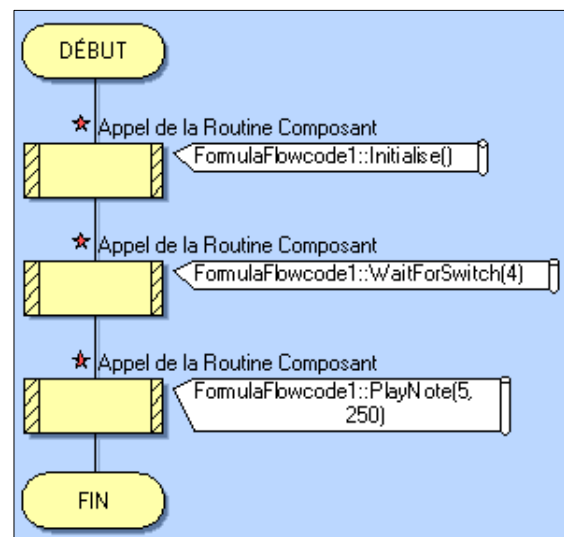
*Explications :* Nous venons de demander à Flowcode d'utiliser la macro « WaitForSwitch ». Celle-ci est différente des précédentes car elle demande à l'utilisateur de lui fournir un paramètre. (Ici un chiffre octet cad entre 0 et 255). Nous lui indiquons donc le bouton ("switch") n°4 qui correspond au bouton poussoir avant droit.



Cette macro bloque le programme tant que le bouton spécifié n'est pas activé. Ceci n'est pas indispensable pour le bon fonctionnement de notre algorithme, c'est juste une question de confort à l'utilisation. Nous vous conseillons fortement de le faire.

- Dans la macro standard suivante, sélectionnez la macro « PlayNote » avec comme paramètre « 5 » et « 250 ». Le robot jouera la note n°5 pendant 250ms pour nous indiquer que le robot est prêt à fonctionner.

- init() est terminé.

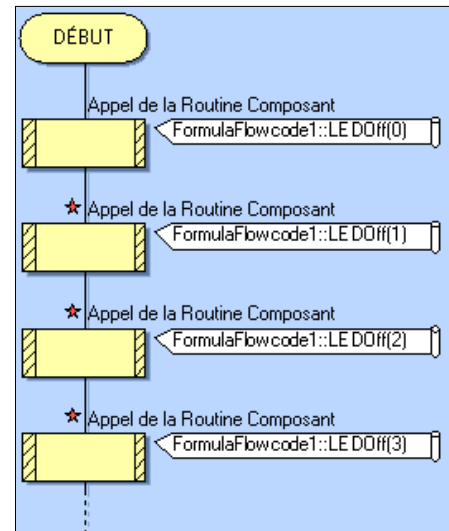




### 3) Remplissage de `eteindre_LEDs()` :

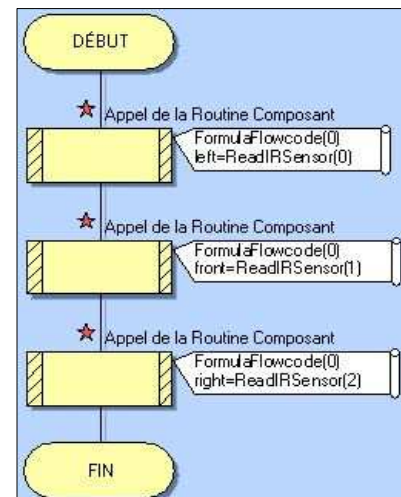
*Cette fonction n'apparaît pas encore dans le `main()` mais nous la remplissons avant les autres car elle est très utilisée par ces dernières. Idem pour `lire_capteurs_IR()`.*

- Dans le haut de la fenêtre, ouvrez le menu « macro », puis « Ouvrir comme algorithme » puis cliquez sur `eteindre_LEDs()`. La macro s'ouvre dans un onglet.
- Insérez 8 macros standard et pour chacune d'elles, sélectionnez « LEDOff ». Attribuez pour chaque macro une valeur croissante de 0 à 7 en paramètre.
- Dorénavant, quand nous appellerons `eteindre_LEDs()`, toutes les LEDs du robot s'éteindront.

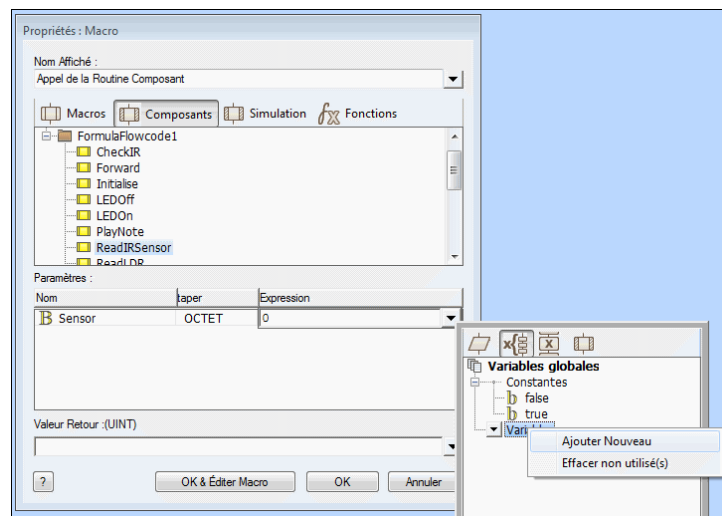


### 4) Remplissage de `lire_capteurs_IR()` :

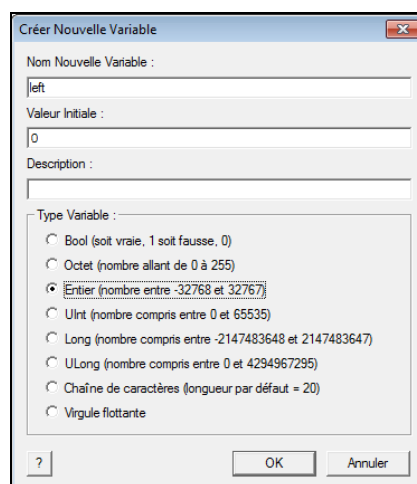
- Ouvrez `lire_capteurs_IR()` dans un onglet et remplissez la de trois macros standard.
- Ouvrez la première et sélectionnez « ReadIRSensor » avec 0 en paramètre. Cette fois ci nous abordons une fonction qui attend un paramètre mais qui renvoie aussi une valeur. Nous la stockerons dans une variable.



- Pour créer une variable cliquez sur la flèche du champ « Valeur Retour » de droite comme ci-dessous puis clique droit sur « Variables ». Enfin cliquez sur « Ajouter nouveau ».



d. Une fenêtre s'ouvre. Donnez un nom à votre variable, initialisez-la et donnez lui le type ENTIER car la macro « ReadIRSensor » renvoie des entiers (voir champ « Valeur Retour »).

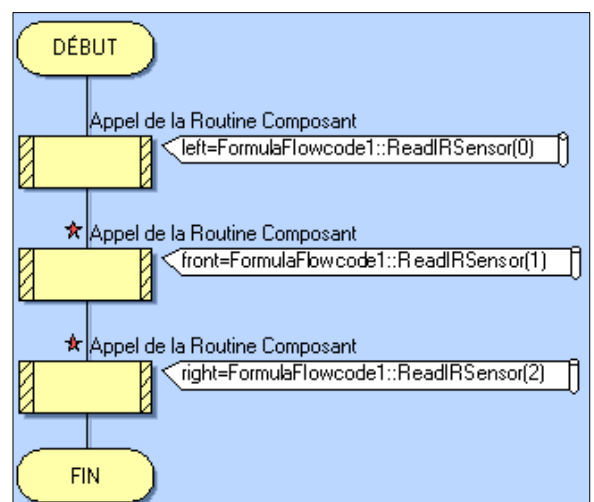


e. Cliquez sur OK et double cliquez sur votre variable. Elle doit apparaître dans le champ « Valeur Retour ».

f. Faites de même pour les deux autres capteurs 1 et 2 avec respectivement les variables \$front et \$right.

Vous obtenez ceci --->

g. Dorénavant, à chaque appel de cette fonction, les valeurs de distance seront mises à jour. Plus l'objet est loin, plus la valeur est grande.



## 5) Remplissage de marche\_libre() :

- Insérez deux macro : `eteindre_LEDs()` et `lire_capteurs_IR()`.
- Insérez une boucle avec pour condition « tant que » {front > 75} avec un test fixé au départ.
- Insérez une macro `lire_capteurs_IR()` puis deux éléments IF (losange) l'un après l'autre (non inversés) remplis comme tel :

IF n°1		IF n°2	
{left > 150}		{right > 150}	
Oui : left = 150	Non : ∅	Oui : right = 150	Non : ∅

Ici, si l'objet est trop loin, nous forçons la valeur de la distance à 150 pour éviter les accidents car les valeurs de \$left, \$right et \$front interviennent dans la vitesse du robot.

- Après ces « IF », placez une zone de calcul et créez deux variables OCTET nommées « compR » et « compL ». Puis écrivez :

- compR = 150 - right
- compL = 150 - left

Ici, les deux variables « comp » complètent \$left et \$right. Autrement dit, plus l'objet est loin, plus \$compR (ou \$compL) est petite. Ceci sera utile dans la définition de la vitesse de chaque roue.

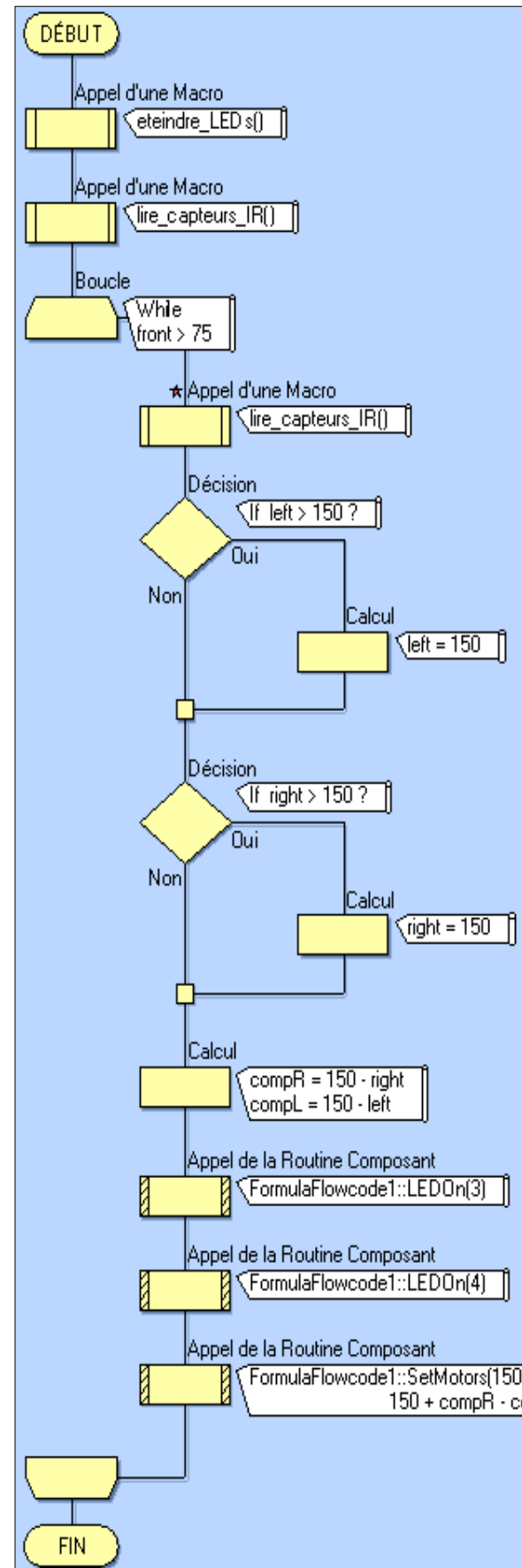
- A la suite toujours dans la boucle, ajoutez deux macros standard pour allumer les LEDs 3 et 4.

- Ajoutez une macro standard et sélectionnez « SetMotors » et remplissez la comme tel :

left_power	right_power
150 + compL - compR	150 + compR - compL

Ici, chaque roue se voit attribuer une vitesse de base de 150. Cette dernière est ajusté par les distances \$compR et \$compL. Cela corrige les défauts de trajectoire du Buggy et fluidifie les virages.

Voici le résultat ---->

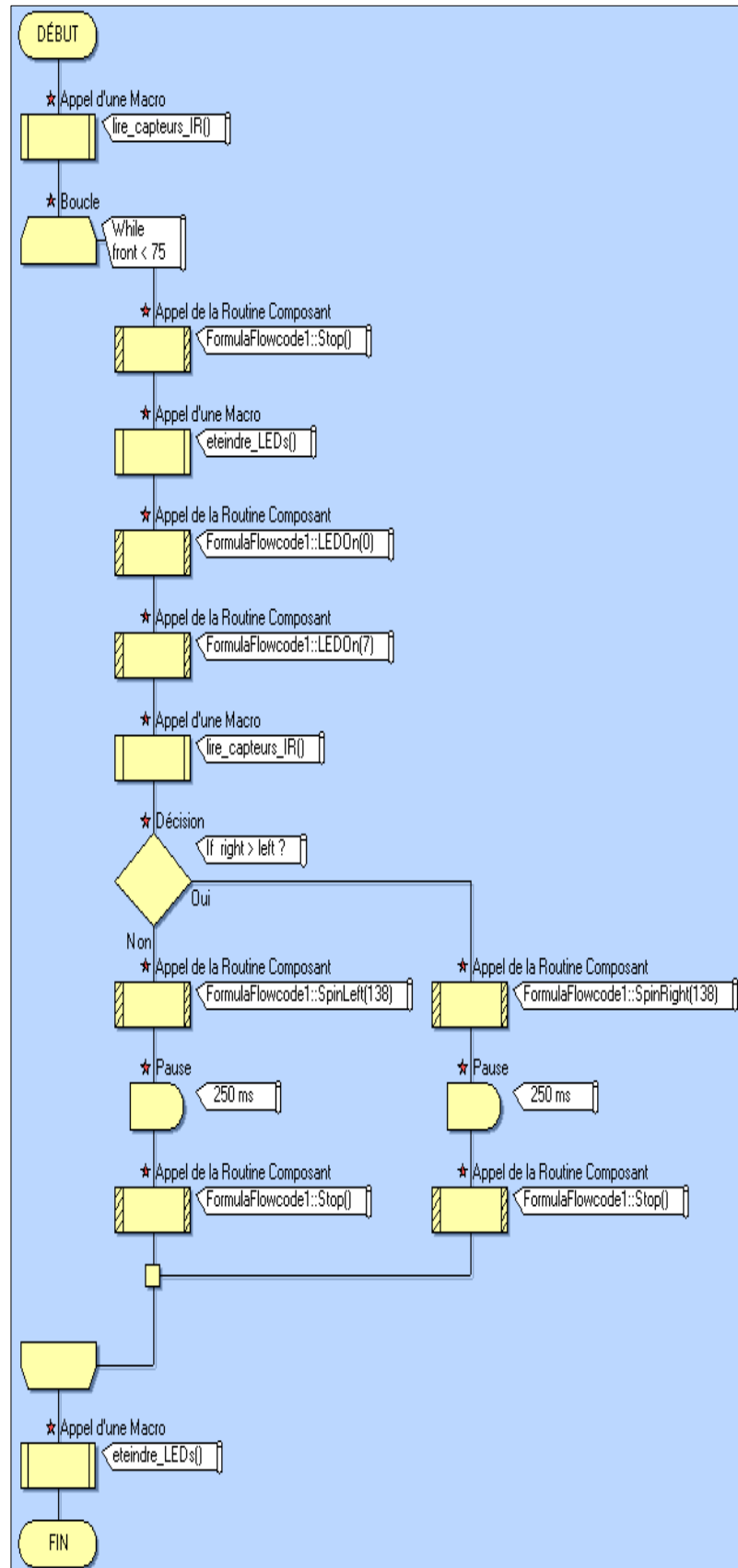


## 6) Remplissage de degagement() :

- Insérez une macro `lire_capteurs_IR()` suivi d'une boucle de condition « Tant que » { `front < 75` }.
- Dans la boucle, ajoutez une macro standard « Stop » suivie d'une macro `eteindre_LEDs()` puis de deux autres macros standard allumant les LEDs 0 et 7. Enfin, ajoutez une macro `lire_capteurs_IR()`.
- Toujours dans la boucle, ajoutez un IF de condition {`right > left`} (non inversée).
  - Si oui, ajoutez une macro standard « SpinRight » de paramètre 138 suivie d'une temporisation de 250 ms et d'une macro standard « Stop ».
  - Si non, ajoutez une macro standard « SpinLeft » de paramètre 138 suivie d'une temporisation de 250 ms et d'une macro standard « Stop ».

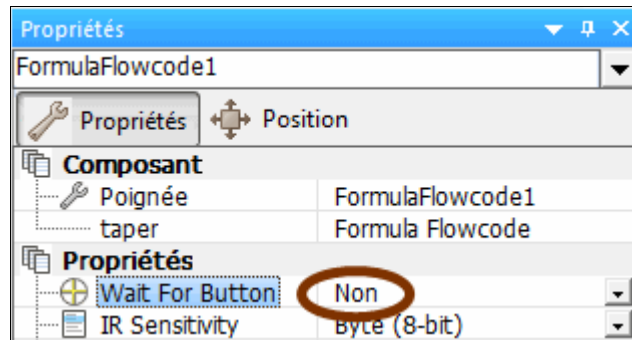
Ce IF permet de régler les conflits lorsque que le buggy est contre un mur et ne sait pas de quel côté tourner. Selon la position du mur latéral, le robot tournera du côté opposé avec une vitesse de 138 pendant 250ms. Ceci correspond à un angle de  $\sim 25^\circ$ . Ainsi, le formula pivote d'un pas de  $25^\circ$  jusqu'à ce qu'il n'ai plus d'obstacle devant lui.

- Après la boucle ajoutez une macro `eteindre_LEDs()`.




## IV. Configuration de Flowcode

Désactivons l'attente au démarrage via les propriétés du robot :



## V. Lancement du programme

- 1) Une fois le programme terminé, cliquez sur  pour lancer la simulation.
- 2) Appuyez/cliquez sur le bouton poussoir S4 du robot dans la simulation 3D.
- 3) Le Buggy devrait atteindre le bout du labyrinthe sans toucher aucun mur.
- 4) Une fois arrivé au bout du labyrinthe, le buggy devrait osciller entre droite et gauche.  
C'est normal, nous n'avons pas encore prévu la gestion des impasses.
- 5) Vous avez les outils pour le faire alors.... Amusez vous !

## VI. Informations

- Vous trouverez dans l'archive de ce TP, une version commentée du programme.