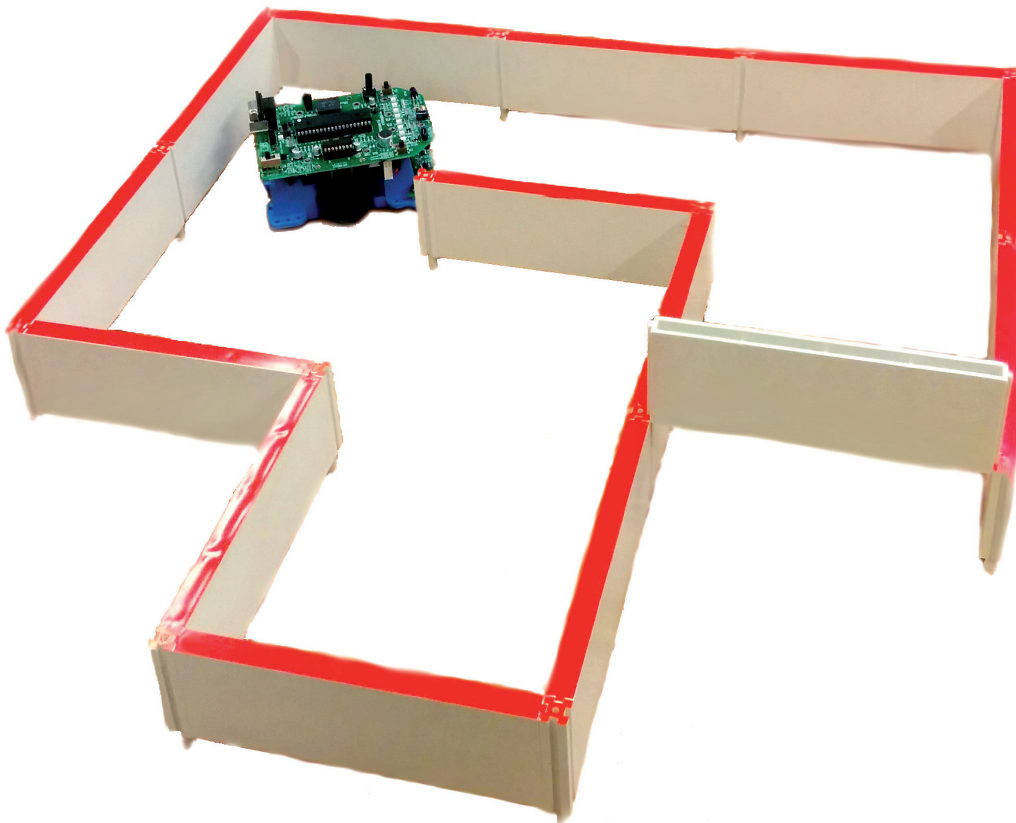


Robot Formula Flowcode

Faire évoluer le robot Formula
dans un labyrinthe simple



Faire évoluer le robot Formula dans un labyrinthe simple

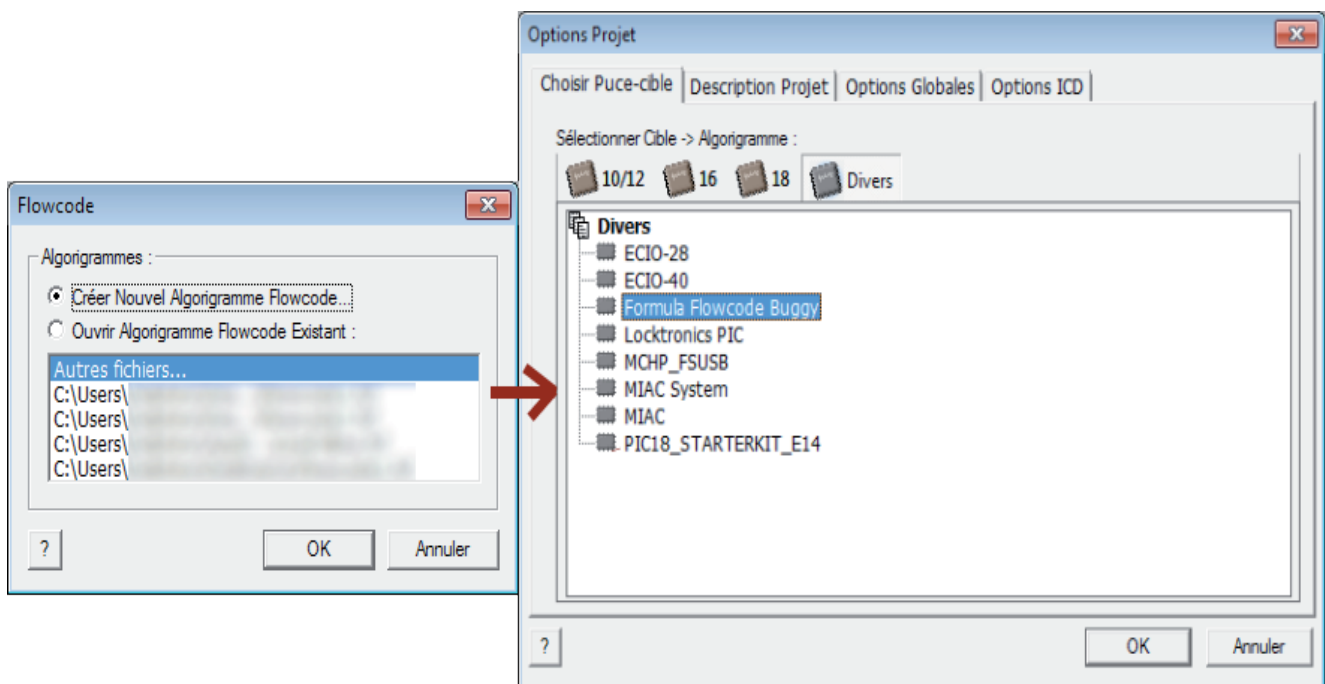
L'objectif de ce TP est de créer un programme qui permettra au robot Formula Flowcode de sortir d'un labyrinthe sans intersection ni impasse.


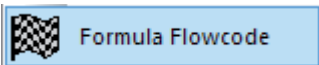
Matériel demandé :

- **1 Robot Formula Flowcode Buggy Driver (Réf : HP794)**
- **Flowcode V5 pour PIC**
- **Un jeu de Murs Labyrinthe pour Formula (Réf : HP458)**

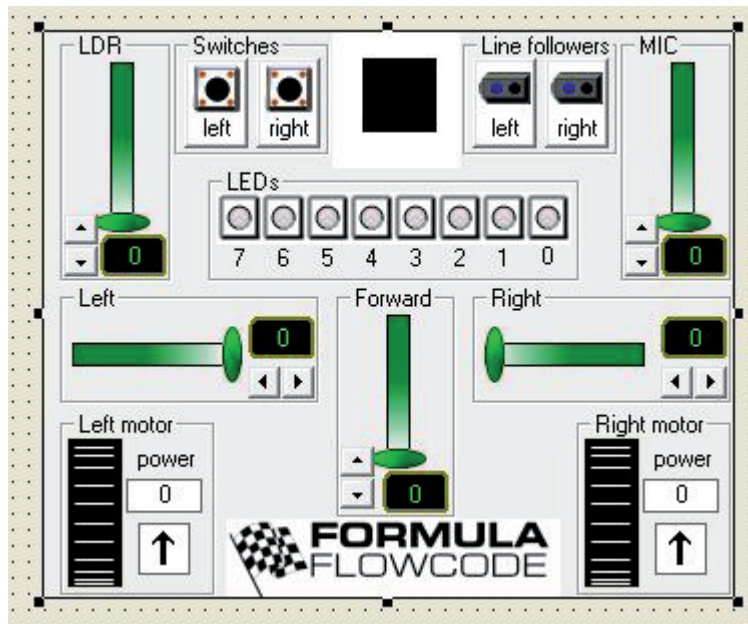
I. Création du projet

1) Lancez Flowcode V5 PIC et créez un nouveau projet pour le Formula Flowcode.



2) Une fois le panneau affiché à l'écran, cliquez sur  puis, dans le menu déroulant, sélectionnez  pour ajouter le robot au projet.

3) L'élément suivant apparaît sur le panneau :



Explication de cette interface :

Power : se trouve à coté des 2 moteurs « right » et « left » et montre la puissance développée par le moteur en question.

Les 2 roues noires : montre le sens de rotation.

La flèche noire en haut : indique la direction que suit le robot (carré noir si arrêté)

Left, Forward, Right: permet de simuler le changement d'état des capteurs.

Switches : Permet de simuler la pression des boutons poussoirs.

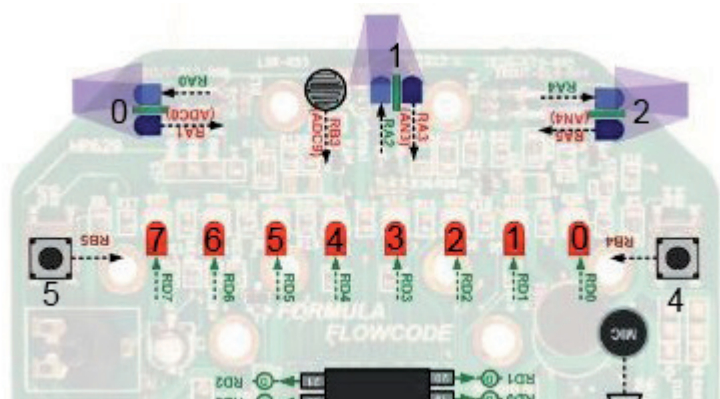


Schéma du robot Formula Flowcode


1) Le programme :



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

Dans un premier temps, 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 ; c'est le cas de notre robot.

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

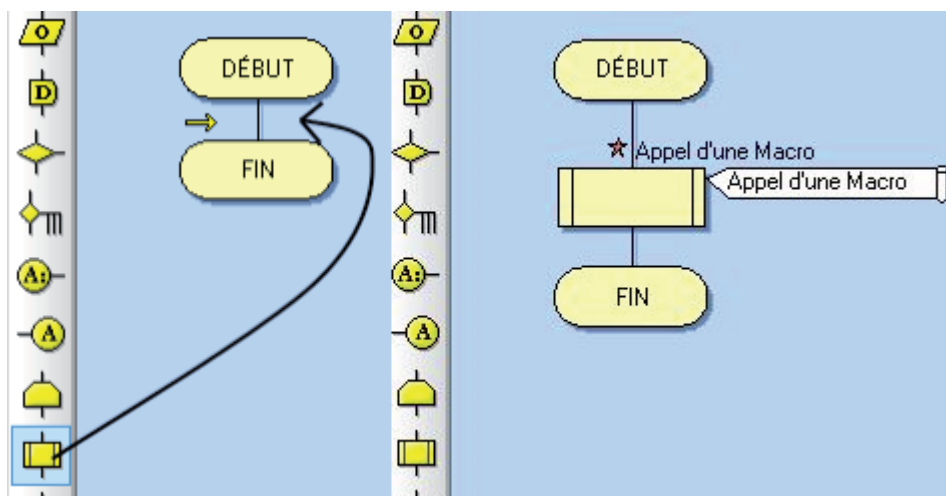
Note :

- 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.

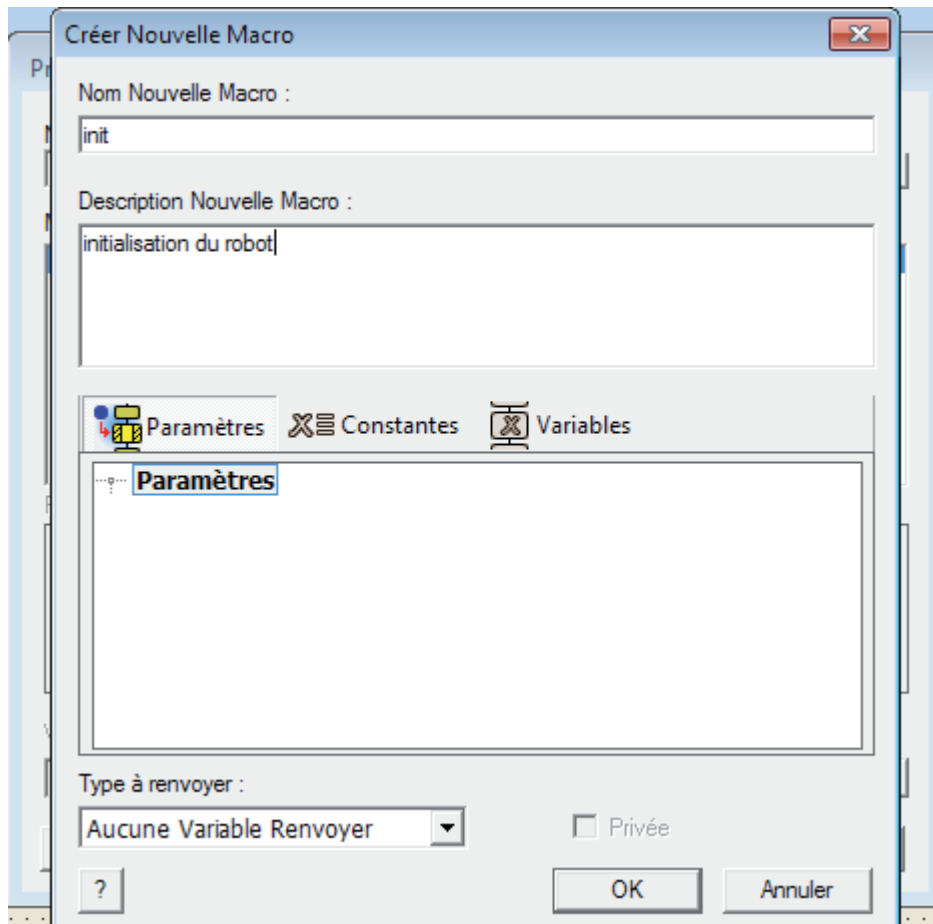
2) Création d'une macro

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

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



b. Double cliquez dessus et double cliquez sur le texte « ...Créer Nouvelle 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 ».



3) 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.

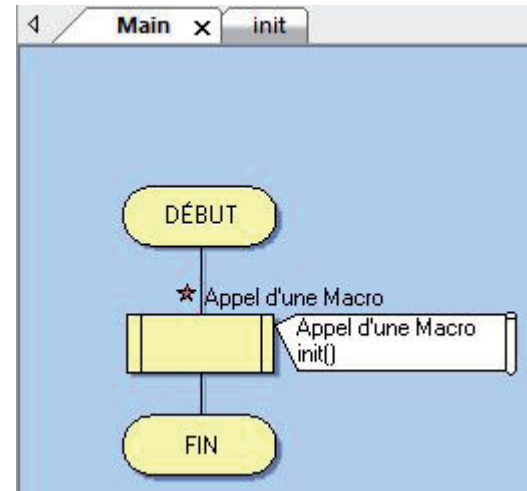
III. Ecriture du code

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

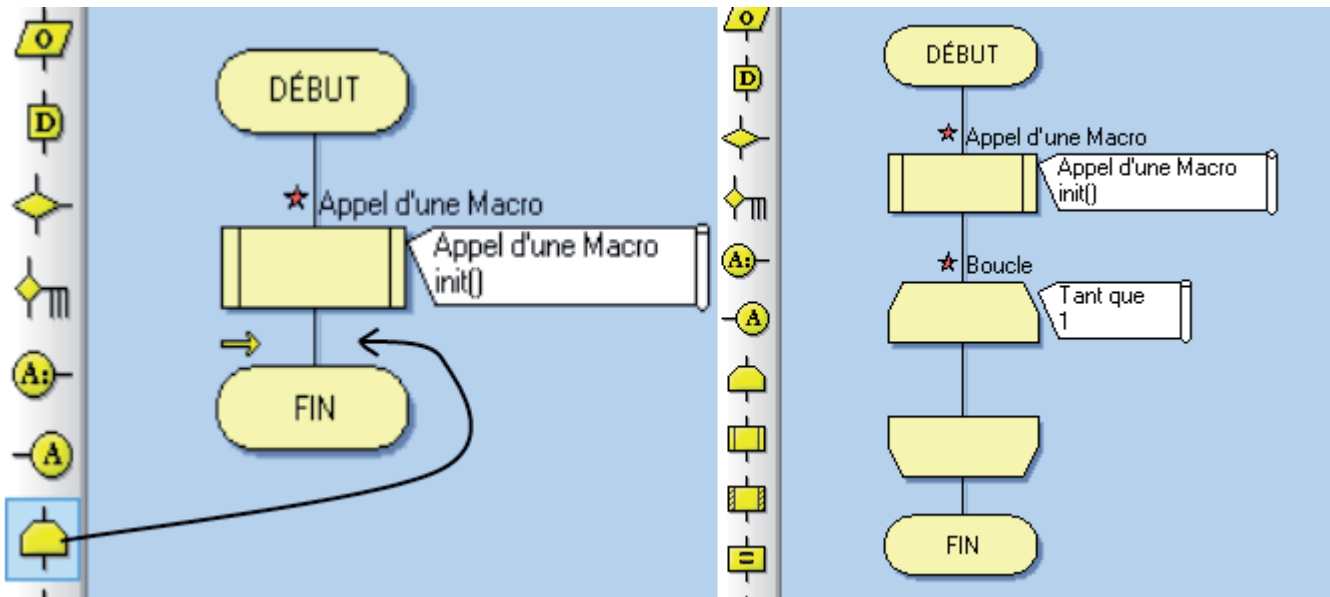
1) Remplissage du « main » (programme principal) :

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

Voici le résultat --->



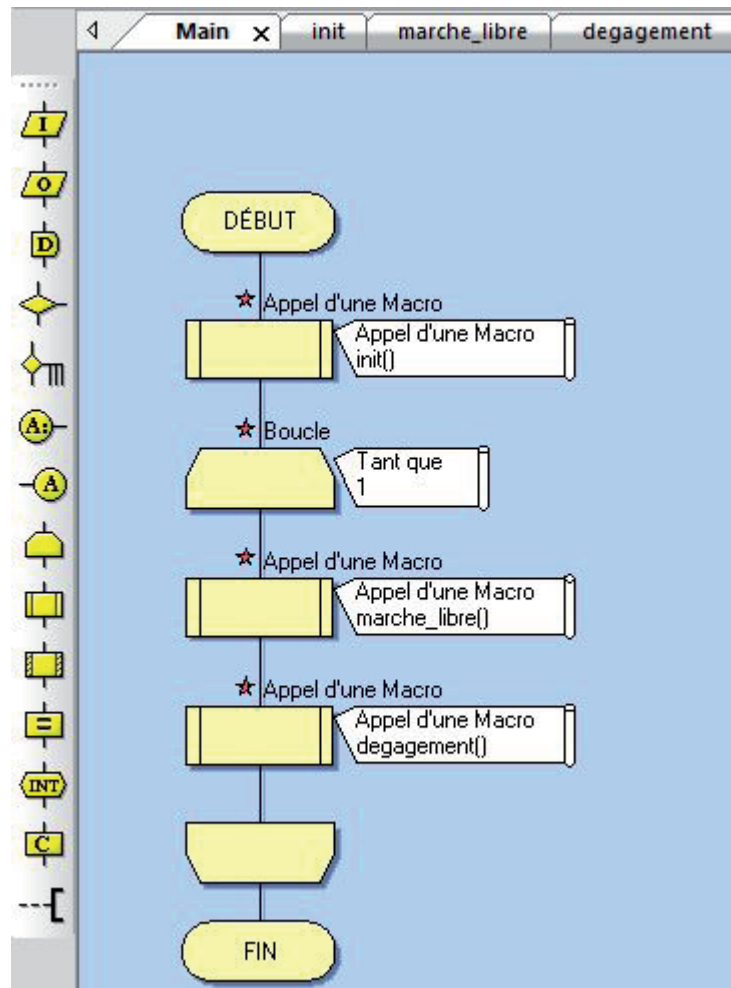
b. Glissez déposez une boucle sous la macro init().



La boucle indique « Tant que 1 », ce qui 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()`.

d. Notre programme principale 'main()' est maintenant terminé.



2) Remplissage de init() :

a. Cliquez sur l'onglet « init ». La macro est vide.

b. Glissez déposez deux macros standard.

c. Double cliquez sur la première macro standard. Une fenêtre apparaît. Sélectionnez « FormulaFlowcode(0) », puis « 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.

Propriétés : Routine Composant

Nom Affiché : Appel de la Routine Composant

Composant : FormulaFlowcode(0)

Macro : LEDOn
LEDOff
ReadSwitch
WaitForSwitch
PlayNote

Paramètres :

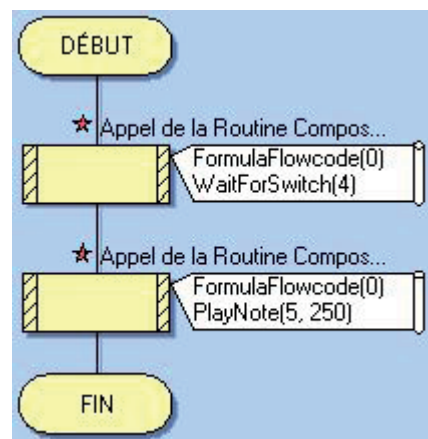
Nom	Type	Expression
sensor	OCTET	4

Valeur Retour :

? OK Annuler

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.

d. Dans la 2ème macro standard, 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.



e. `Init()` est terminé.

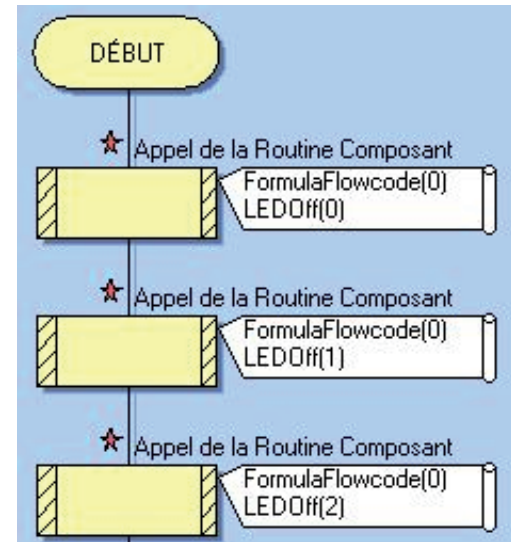
3) Remplissage de `eteindre_LEDs()` :

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

a. Dans le haut de la fenêtre, ouvrez le menu « macro », puis « Ouvrir comme algorithme ». Cliquez sur `eteindre_LEDs()`. La macro s'ouvre dans un onglet.

b. Insérez 8 macros standard et pour chacune d'elles, sélectionnez « LEDOff ». Attribuez pour chaque macro une valeur croissante de 0 à 7 en paramètre.

c. Dorénavant, quand nous appellerons `eteindre_LEDs()`, toutes les LEDs du robot s'éteindront.

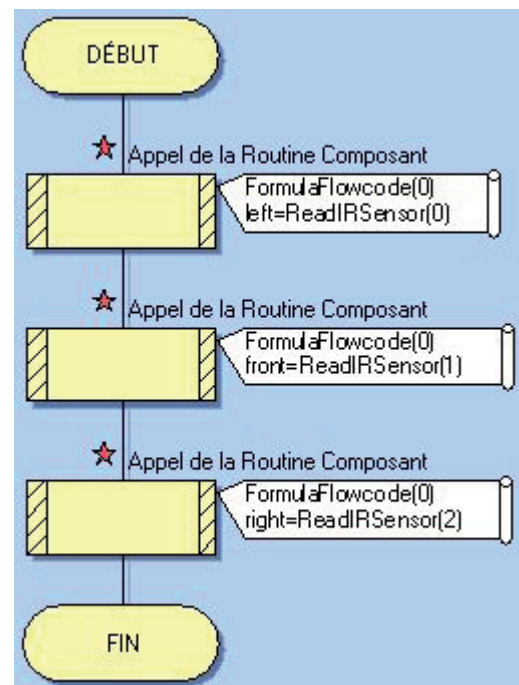


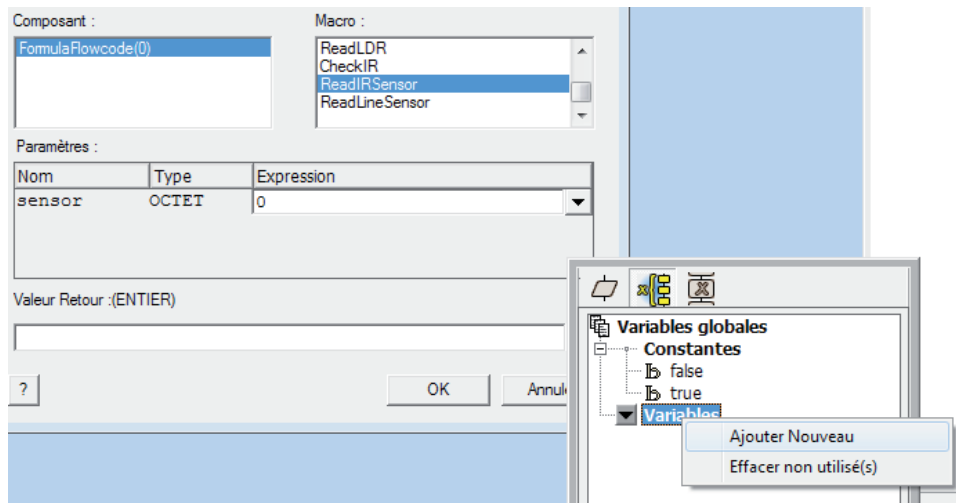
4) Remplissage de `lire_capteurs_IR()` :

a. De la même façon, ouvrez `lire_capteurs_IR()` dans un onglet et insérez trois macros standard.

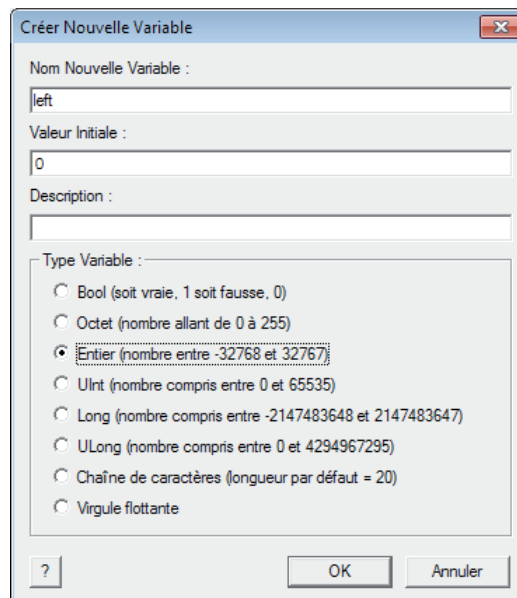
b. Ouvrez la première macro et sélectionnez « ReadIRSensor » avec 0 en paramètre. Cette fonction est une fonction qui attend un paramètre mais qui renvoie aussi une valeur. Nous la stockerons dans une variable.

c. Pour créer une variable, cliquez sur la flèche du champ « Valeur Retour » de droite comme ci-dessous puis cliquez 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.

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 macros : `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 : Ø	Oui : right = 150	Non : Ø

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.

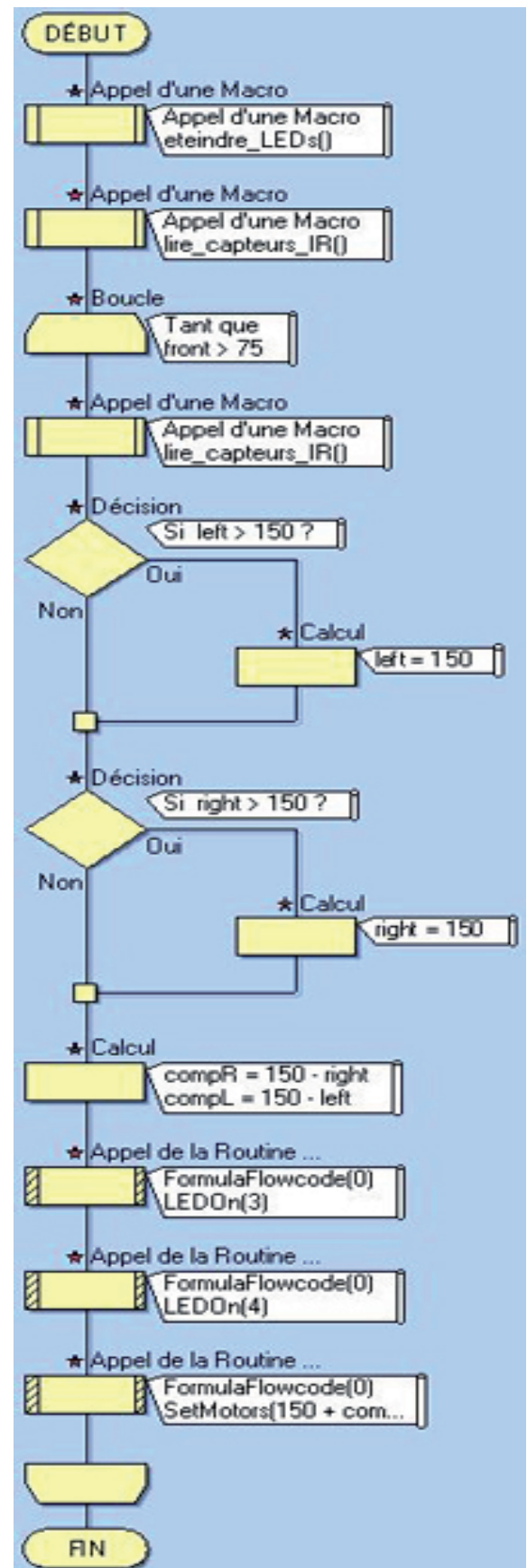
- A la suite toujours dans la boucle, ajoutez deux macros standard pour allumer les LEDs 3 et 4.

- 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 ajustée 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 () :

a. Insérez une macro `lire_capteurs_IR()` suivie d'une boucle de condition « Tant que » { front < 75 }.

b. Dans la boucle, ajoutez une macro standard « Stop » suivie d'une macro `eteindre_LEDs()` puis de deux autres macros standards allumant les LEDs 0 et 7. Enfin, ajoutez une macro `lire_capteurs_IR()`.

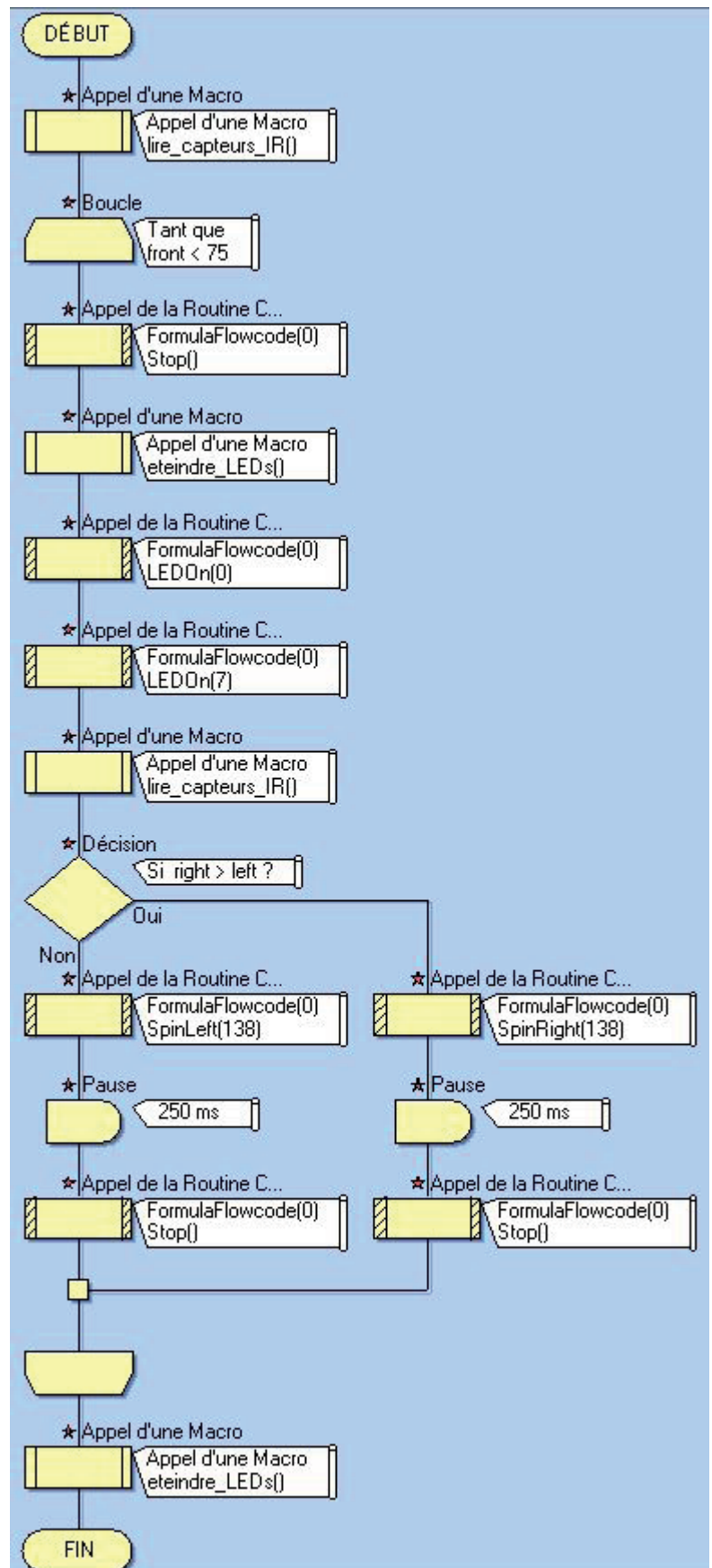
c. Toujours dans la boucle, ajoutez un IF de condition {right>left} (non inversée).

- 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 de condition 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° jusqu'à ce qu'il n'ait plus d'obstacle devant lui.

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

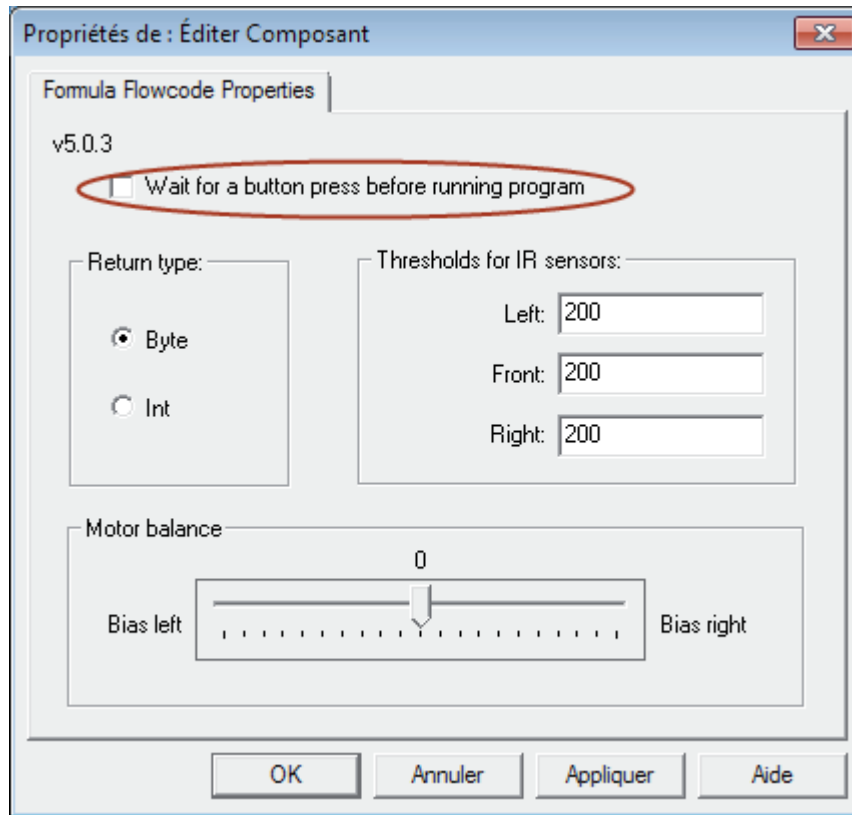
Voici le résultat --->



IV. Configuration de Flowcode

1) Démarrage du robot

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



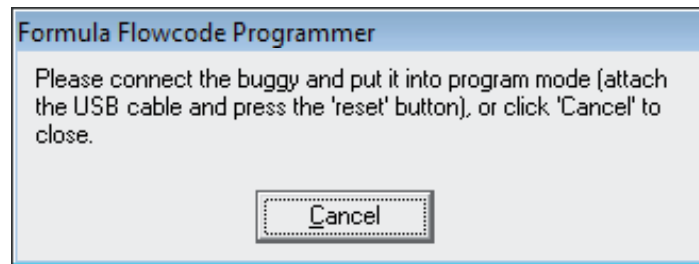
2) Calibrage des moteurs

Vous devez également régler la balance des moteurs dans les propriétés du moteur. Ce réglage est propre à chaque robot et doit permettre de conserver une trajectoire rectiligne avec une vitesse théorique égale sur les deux moteurs.

V. Téléversement

Une fois le programme terminé, cliquez sur  pour envoyer le programme au robot.

Pour cela, branchez le câble USB au buggy et alimentez-le avec l'interrupteur SW1 en bas à droite de la carte. Appuyez sur le bouton RESET (au dessus du port USB) quand l'ordinateur vous affiche le message ci-dessous :



La LED du port USB doit clignoter.

Débranchez le câble, appuyez sur le bouton n°4 et Amusez-vous !

VI. Annexes

- Vous trouverez dans l'archive de ce TP, une version commentée du programme.
- La forme et la taille du labyrinthe sont arbitraires.