STI2D Option SIN Terminale	AP2.1 : programmation sous forme d'algorithmes	

Programmation informatique

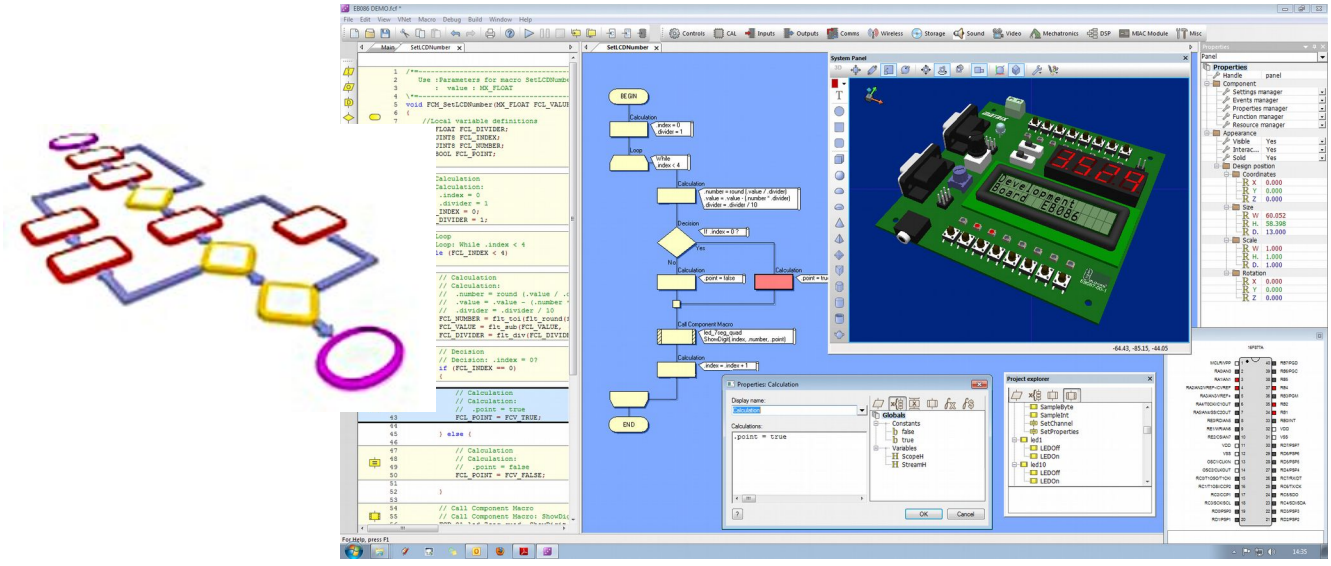
Durée prévue : 4 séances de 3h (12h)
Problématique : révision des structures des langages informatiques

Compétences visées	Savoirs associés
Être capable de mettre en œuvre et simuler un programme informatique sous forme d'algorithmes.	

Plan de l'étude	Remarque
I. Introduction, rappels et découverte du matériel	
II. Découverte du logiciel Flowcode	
III. Clignotement d'une Led	
IV. Chenillard	
V. Afficheurs 7 segments et chaînes de caractères	
VI. Gestion d'un parking	
VII. Feux de carrefour	

Logiciels	Matériels
Flowcode	Ordinateur

Mode de distribution	Format numérique
Dossier technique associé	PIC16f88.pdf
Dossier ressource associé	notice_flowcode_v4 (ferry-Versailles).pdf

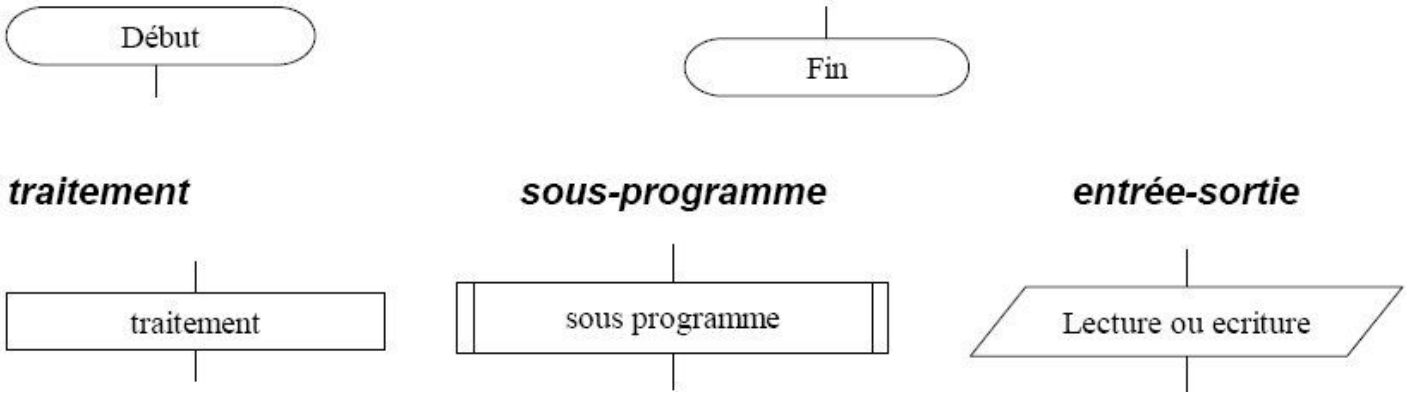


I. Introduction, rappels et découverte du matériel

1. Algorithmes

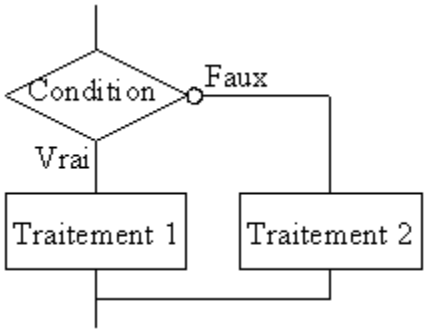
Nous allons créer et simuler des programmes informatiques simples sous forme d'algorithmes ('flowchart' en anglais). Cette représentation graphique de la description du fonctionnement d'un programme va nous permettre de nous familiariser avec les structures utilisées en informatique. La forme utilisée ici est un algorigramme (ou organigramme de programmation).

Il comporte un début et une fin, des liaisons orientés, des blocs "symbole" : traitement (affectations de variables, calculs..), gestions des entrées et des sorties, sous-programmes (fonctions).



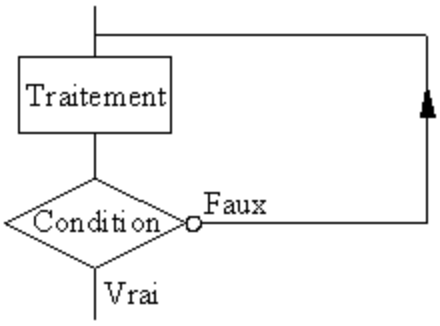
Quelques structures :

Le choix :

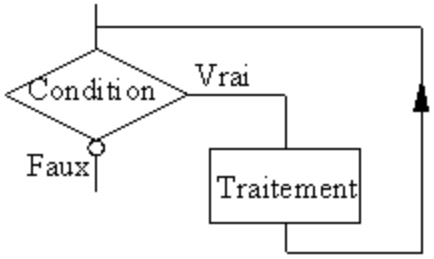


Si condition validées
Alors
traitement 1
Sinon
traitement 2
Fin_Si

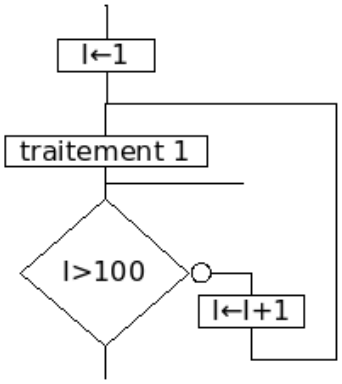
Les boucles (structures répétitives) :



Répéter
traitement 1
Jusqu'à condition
Fin_répéter



Tant que condition
traitement 1
Fin_tant



Depuis I=1 à 100
traitement 1
I←i+1
Fin_depuis

2. Qu'est qu'un PIC ?

Nous allons travailler 'virtuellement' sur le PIC 16F88 (18 pin).

Un PIC est un microcontrôleur c'est à dire est une unité de traitement et d'exécution de programmes informatiques (comme un microprocesseur) à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans nécessiter l'ajout de composants annexes. Un microcontrôleur PIC peut donc fonctionner de façon autonome après programmation (sur notre carte arduino il y a un aussi microcontrôleur du même genre).

Les PIC intègrent une mémoire programme non volatile (FLASH), une mémoire de données volatile (SRAM), une mémoire de données non volatile (E2PROM), des ports d'entrée-sortie (numériques, analogiques, MLI, UART, bus I2C, Timers, etc.), et même une horloge interne.

A partir de la documentation technique (PIC16f88.pdf) répondre aux questions suivantes :

a) Les entrées et sorties sont regroupées par 'port'. Combien y-a-t-il de ports sur notre PIC 16F88 ?
b) Combien y-a-t-il de bits par port ?
c) Donnez les broches utilisées par le port A
d) Pour la broche 17 (RA0/AN0), donnez ses caractéristiques et utilisations possibles (3 différentes)

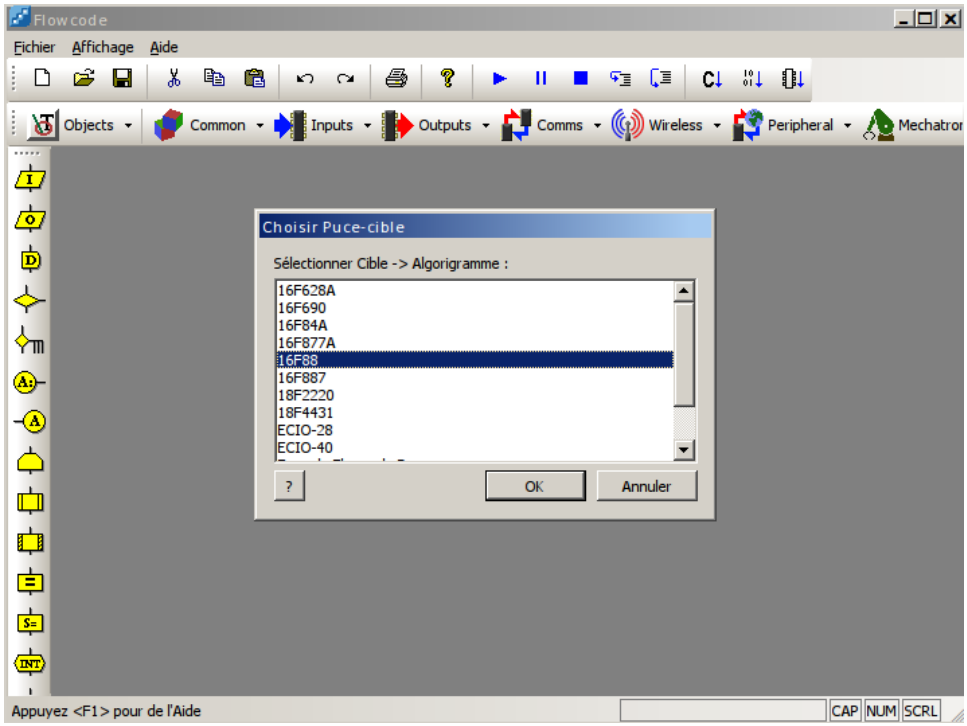
II. Découverte du logiciel Flowcode

Nous allons créer et simuler des programmes informatiques simples sous forme d'algorithmes à l'aide du logiciel Flowcode.

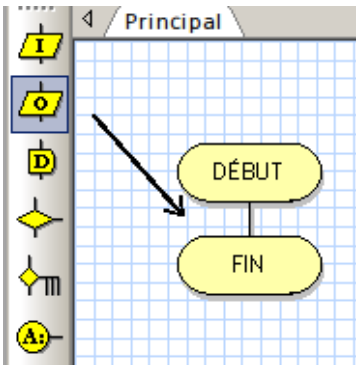
Remarque : le TP est conçu à partir de la version 4 de Flowcode (votre version peut donc être différente).

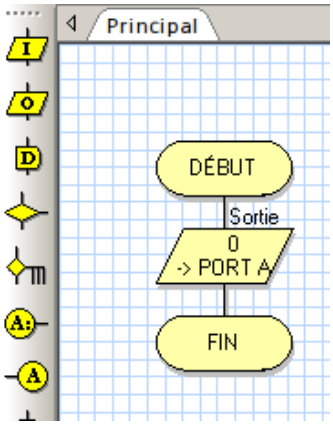
Objectif : écrire sur une sortie du PIC

- Lancer le logiciel Flowcode et créez un nouvel algorithme Flowcode
- Choisir le PIC : 16F88

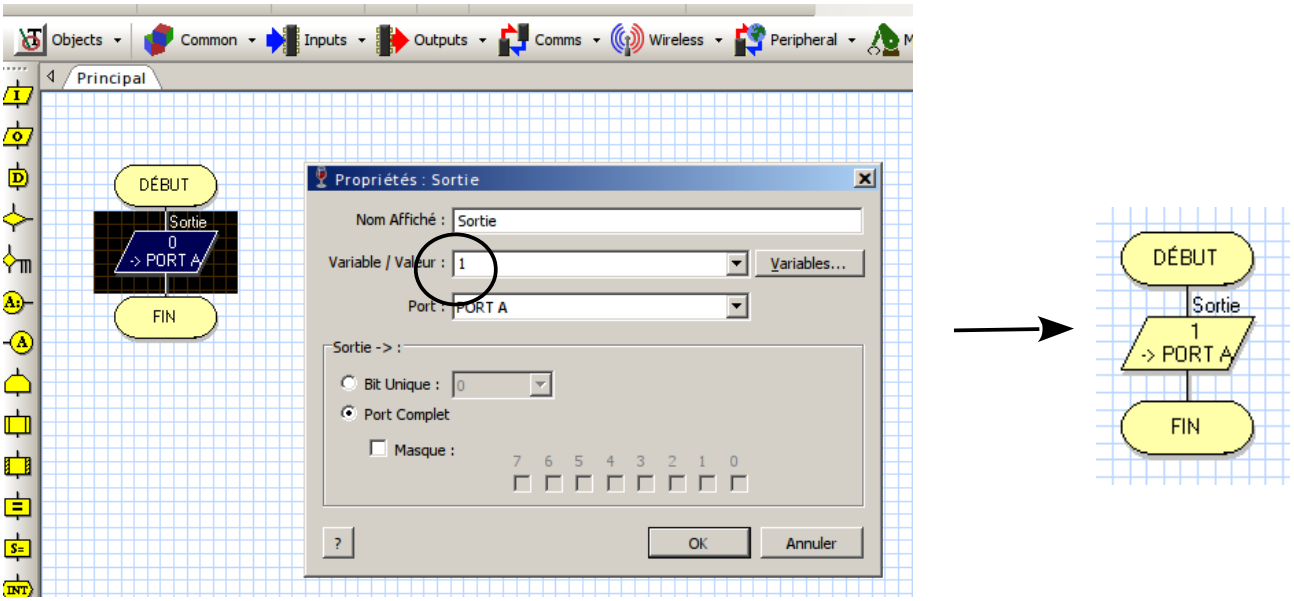


• Glissez le bloc 'Output' au milieu de la structure début/Fin afin d'obtenir :

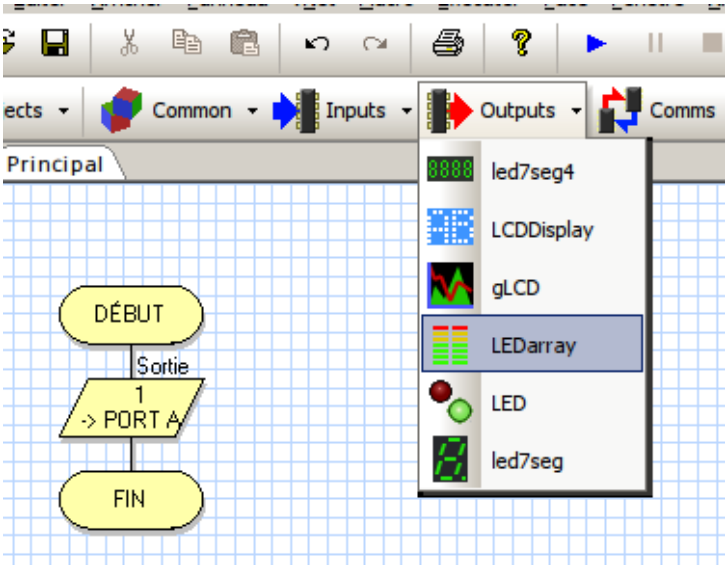




• Double cliquez sur le bloc de sortie afin de paramétrer la valeur '1' à affecter au port A

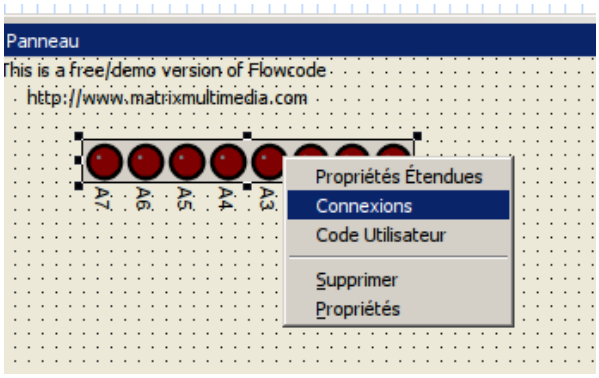


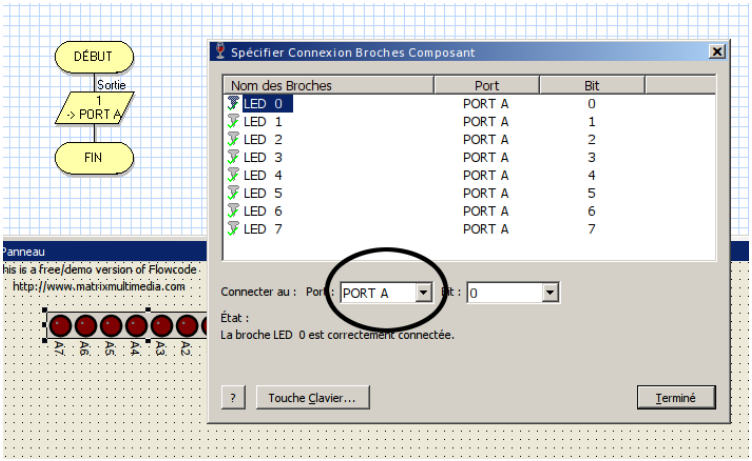
Pour pouvoir tester (simuler) le programme nous allons associer à notre microcontrôleur des leds



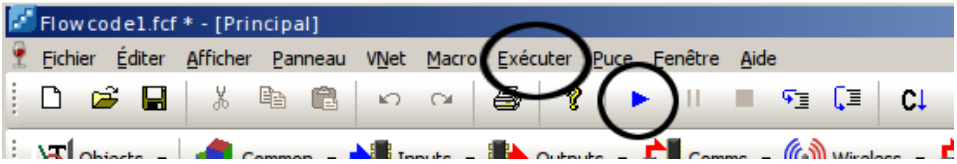
Choisir 'LEDarray'

- Avec le clic droit de la souris choisir 'Connexions' et associez le Port A à nos Leds.





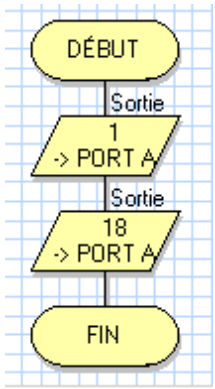
• Simuler le programme (menu 'Exécuter' ou ►)



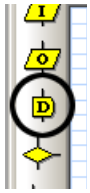
a) Expliquez ce qui se passe

b) Modifiez le programme comme suit :

Expliquez ce qui se passe



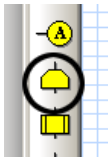
c) Rajouter une pause (delay) de 2 secondes entre l'écriture des valeurs. Expliquez maintenant ce qui se passe



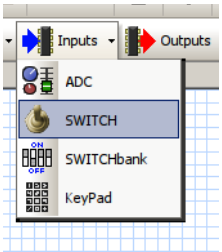
III. Clignotement d'une Led

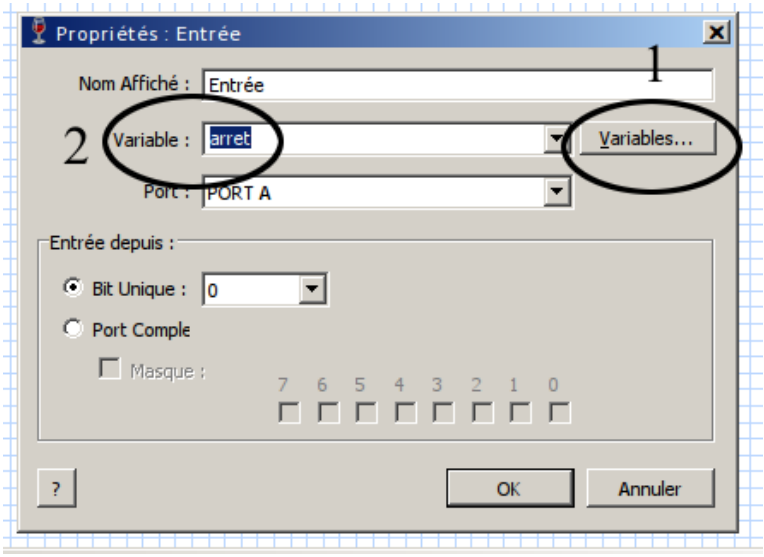
3.1. Version 1

Cahier des charges :



- objectif du programme : on veut faire clignoter la led 0 du port B (B0) à la fréquence de 1 Hz
- pour éteindre la led on affectera 0 à la sortie B0, pour l'allumer on affectera 1 à cette même sortie B0.
- le programme contiendra une boucle 'tant que' dont l'arrêt dépendra d'une entrée (lorsque l'entrée sera à 1, la boucle s'arrêtera donc le programme s'arrêtera sinon le programme tournera sans s'arrêter).
- Cette entrée sera A0 du port A et simulée par un interrupteur (switch).
- Cette entrée A0 sera associée à la variable 'arret' que vous allez créer et pouvoir ensuite réutiliser dans le test de votre boucle.





IMPORTANT

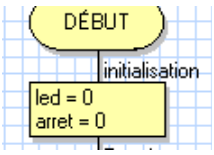
Quand on veut débugger un programme il est parfois utile d'avoir des outils appropriés. C'est le cas de Flowcode qui propose notamment d'exécuter le programme pas à pas (menu 'Exécuter' ou touche F8). On peut aussi y associer un tableau dynamique des variables du programme (on visualise ainsi l'état des variables en temps réel). Quand un programme ne fonctionne pas correctement c'est souvent le meilleurs moyen de comprendre pourquoi!

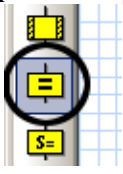
Essayez votre programme et quand tout fonctionne faites valider par le professeur

3.2. Version 2

On se propose d'améliorer un peu ce programme :

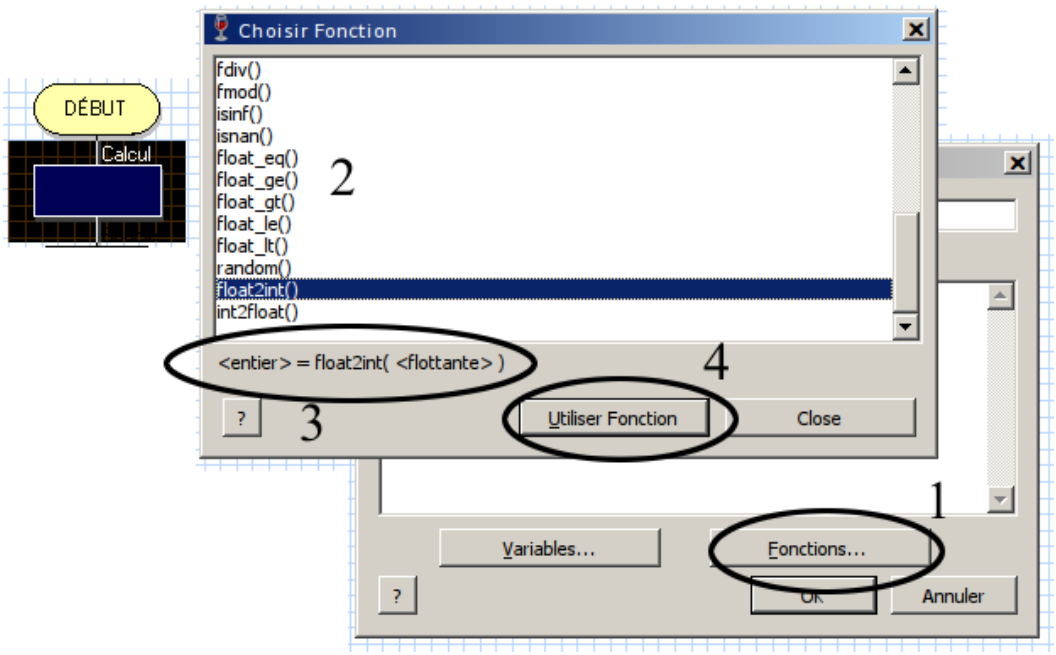
- ***Ajout d'une variable 'led' (type 'octet'). C'est cette variable que l'on va affecter à la sortie B0 (cette affectation ne doit avoir lieu qu'une seule fois).***
- ***Pour changer la valeur de cette variable 'led', on utilisera la fonction NOT dans un bloc de calcul. Les blocs de calculs permettent de manipuler les variables (affectation, calculs, …). Un exemple ci-contre de bloc de calcul utilisé en tête de programme pour l'initialisation des variables. Ces blocs de calcul peuvent contenir des fonctions mathématiques et logiques déjà programmées.***





Pour choisir une fonction déjà existante : exemple avec la fonction 'float2int():

1 : choisir 'fonction' en double-cliquant sur le bloc calcul de votre programme
2 : choisir le type de fonction (l'exemple donné ci-après concerne la transformation d'un float en integer)
3 : syntaxe d'utilisation de la fonction (on remplace les variables exemples <…..> par nos propres variables)
4 : On valide l'utilisation de la fonction



Syntaxe de la fonction NOT: ma_variable = NOT ma_variable1 (les variables sont de type octet donc sur 8 bits)

résultat: la variable 'ma_variable' est l'inverse de la variable 'ma_variable1'

exemples: si ma_variable1=0 (0000000) alors ma_variable sera égal à 255 (11111111)

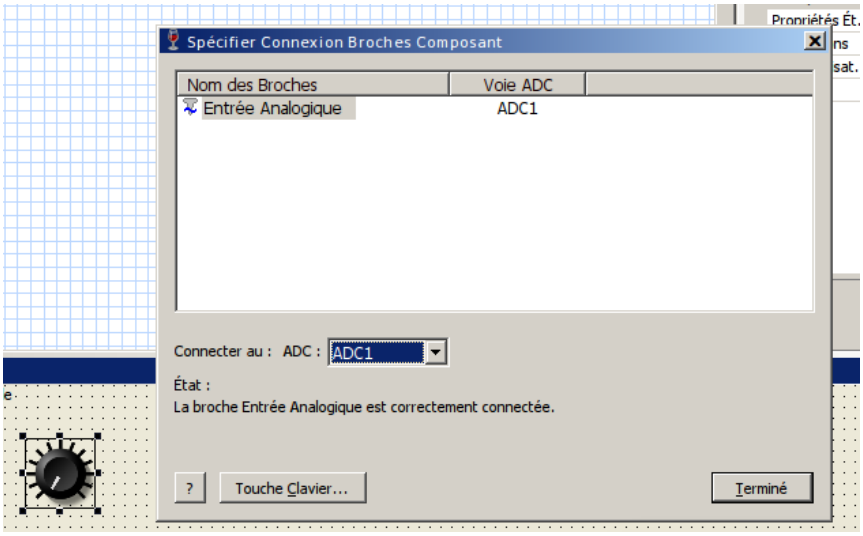
si ma_variable1=19 (00010011) alors ma_variable sera égale à 11101100 (c'est à dire 236)

Essayez votre programme (réalisé avec la fonction NOT) et, quand tout fonctionne, faites valider par le professeur

3.3. Version 3

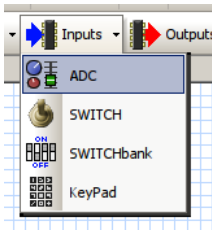
On se propose d'améliorer encore un peu ce programme :

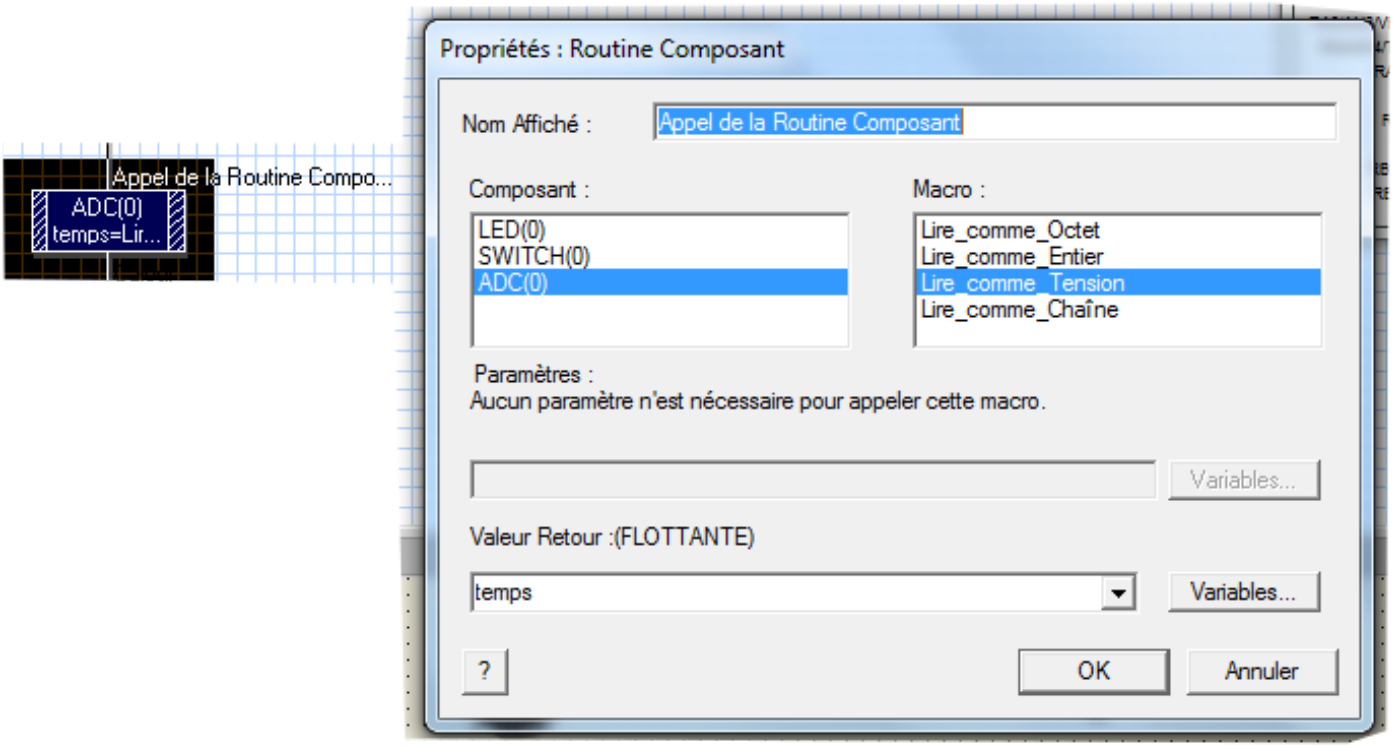
- ***Le temps d'attente va être réglable et dépendra de la variable 'temps'. Celle-ci sera lue sur l'entrée analogique ADC1***



Méthode :

- Installer d'abord le potentiomètre (ADC) dans la partie 'panneau' du logiciel.
- Il faut ajouter un sous programme spécial pour gérer l'entrée analogique associée au potentiomètre : ajouter le bloc 'routine composant' puis double-cliquez dessus :





Choisir 'Lire_comme_Tension' et choisir comme retour la variable 'temps' (qu'il faut créer en Float)

Essayez votre programme et quand tout fonctionne faites valider par le professeur

IV. Chenillard

Cahier des charges :

- objectif du programme : on veut allumer les leds du port B une par une (soit de droite à gauche, soit de gauche à droite, suivant la position d'un interrupteur 'Sens')
- vous utiliserez une variable 'led' qui sera du type octet (donc 8 bits) et qui sera affectée au port B (qui comporte 8 sorties)
- le temps de défilement des leds est réglable à l'aide d'un potentiomètre (1V = 1 seconde).
- Le programme boucle sur lui-même jusqu'à la demande d'arrêt (interrupteur 'arrêt').

Exemple de résultat que l'on veut obtenir (ici dans le sens droite vers gauche):

…etc .

Remarque : la fonction décalage peut être réalisée de 2 manières différentes, soit en utilisant les propriétés de l'algèbre du binaire, soit en utilisant la fonction 'décalage de bits' (>> ou <<) présente dans Flowcode.

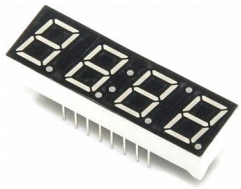
Petit indice sur le binaire (bien analyser le résultat en décimal) pour trouver comment réaliser un décalage à gauche:

Binaire sur 8 bits	Décimal
00000001	1
00000010	2
00000100	4
00001000	8
…….	

Essayez votre programme et quand tout fonctionne faites valider par le professeur

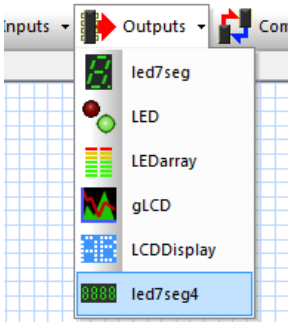
V. Afficheurs 7 segments et chaînes de caractères

Cahier des charges :



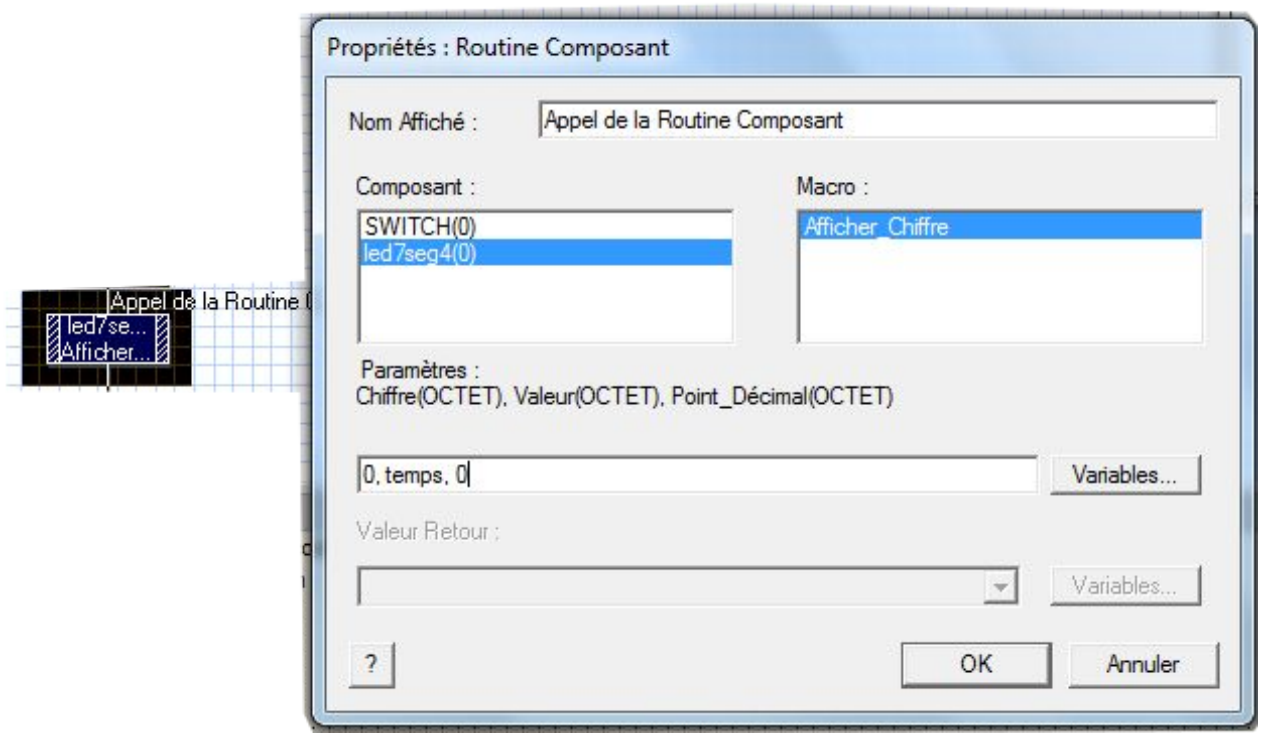
- objectif du programme : afficher les valeurs d'un potentiomètre (valeurs de 0 à 20) sur 2 afficheurs 7 segments (appartenant à un quadruple afficheur)
- objectif pédagogique: découverte et manipulation des chaînes de caractères

Etape 1 : compréhension de la gestion des afficheurs 7 segments



- vous allez d'abord choisir le composant contenant 4 afficheurs 7 segments et le déposer sur la fenêtre 'panneau'

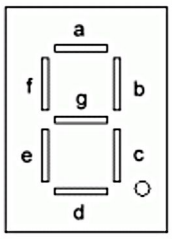
- vous allez ensuite choisir la 'routine composant' associée à l'afficheur choisi :



cette routine 'Afficher_Chiffre' nécessite 3 paramètres :

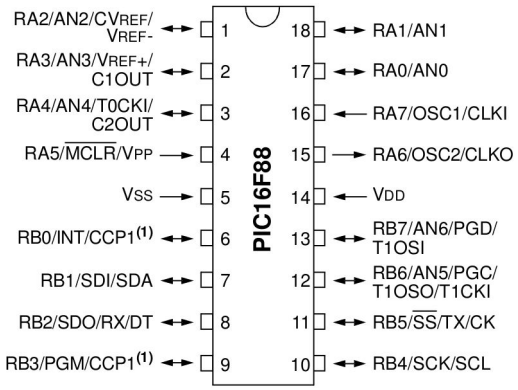
- chiffre : il vaut 0, 1, 2 ou 3 selon que l'on veut allumer le chiffre de poids faible, le chiffre de rang 2, de rang 3, ou le chiffre de poids fort.
- valeur : il s'agit de la valeur à afficher (de 0 à 9) ici on a mis par exemple la variable 'temps'
- point décimal : il vaut 0 ou 1 et permet d'afficher ou non la "virgule" à droite du chiffre sélectionné sur l'afficheur. (ici 0 donc pas de virgule)

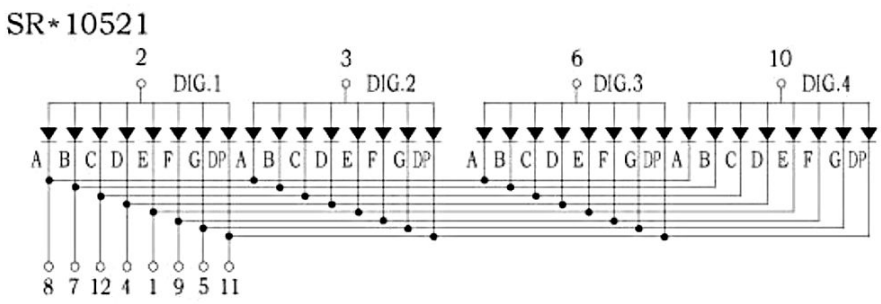
a) Faire le programme affichant 5 sur le digit des unités (variable 'unite' contenant 5 à afficher). Le programme contiendra une boucle infinie (fonctionnement comme une carte Arduino).



b) Faire le programme affichant 2 sur le digit des dizaines (variable 'dizaine' contenant 2 à afficher)

c) En allant sur l'afficheur proposé par Flowcode (clic droit) et en analysant les connexions proposées, faites le schéma de câblage 'physique' (électrique) de l'afficheur (référence SR 10521: anode commune) :





d) Expliquez électriquement comment votre programme allume les leds de l'afficheur (prendre l'exemple du chiffre 5 affichée sur l'unité).

Comme la routine ne peut afficher qu'un chiffre à la fois et que l'on veut afficher plusieurs chiffres, on va afficher "très vite" chacun des chiffres pour donner l'illusion que tous les chiffres sont affichés en même temps (une led ne s'éteignant pas instantanément cela sera invisible pour nous).
e) Faire le programme affichant 5 sur le digit des unités puis 2 sur le digit des dizaines. Cela fonctionne-t-il , sachant que la vitesse de simulation est bien plus lente que le fonctionnement réel du 16F88?

Etape 2 : affichons un chiffre sur 2 des afficheurs 7 segments

Maintenant notre but est d'afficher le chiffre 56 sur notre quadruple afficheur. Ce chiffre sera contenu dans la variable 'nombre1' qui sera du type 'float' (flottant ou virgule flottante). Comme il faut envoyer digit par digit, il va falloir trouver une astuce pour y arriver. La solution réside dans l'utilisation des chaînes de caractères (que l'on peut manipuler facilement).

Une « ***chaîne de caractères*** », c'est un nom pour désigner… du texte, tout simplement !
Une chaîne de caractères, c'est donc du texte que l'on peut mettre dans une variable en mémoire.
Exemple: ma_variable = 'je suis un élève de STI'
"ma_variable" est une variable du type 'chaîne de caractères' contenant le texte : je suis un élève de STI

1ère partie: découverte des fonctions de gestion et manipulation des 'chaînes de caractères':

A l'aide du document ressources 'notice_flowcode_v4.pdf', répondre aux questions suivantes :
a) Expliquez le fonctionnement ou le rôle, détaillez la syntaxe et donner un exemple d'utilisation de la fonction 'ToString$'
b) Expliquez le fonctionnement ou le rôle, détaillez la syntaxe et donner un exemple d'utilisation de la fonction 'Length$'
c) Expliquez le fonctionnement ou le rôle, détaillez la syntaxe et donner un exemple d'utilisation de la fonction 'Mid$'
d) Expliquez le fonctionnement ou le rôle, détaillez la syntaxe et donner un exemple d'utilisation de la fonction 'StringToInt$'

2ème partie: *utilisation des ces fonctions pour réaliser notre programme:*

Principe: on va convertir notre variable flottante en entier. Ensuite on la convertit en chaîne de caractère et on extrait l'unité et la dizaine. Cette unité et cette dizaine seront mis dans 2 variables du type 'chaîne de

caractères' puis convertis en entiers pour enfin être affichés sur les afficheurs 7 segments.

e) Faire le programme qui permet d'afficher 56 sur le quadruple afficheur 7 segments (vous aurez besoin d'utiliser les fonctions vues précédemment et de bien respecter le principe proposé). Essayez votre programme et quand tout fonctionne faites valider par le professeur

Etape 3 : faire le programme final répondant au cahier des charges : potentiomètre variant de 0 à 20 dont la valeur s'affiche sur le quadruple afficheur 7 segments. 0 correspond à la valeur minimum (0 Volts) et 20 à valeur max (5V).

Pour vous aidez voici le principe général à mettre en œuvre:

- *récupérer et traiter la valeur du potentiomètre*
- *mettre cette valeur sous forme de chaîne de caractères*
- *extraire l'unité et la dizaine*
- *convertir cette unité et cette dizaine en 'entier'*
- *affichez les valeurs*

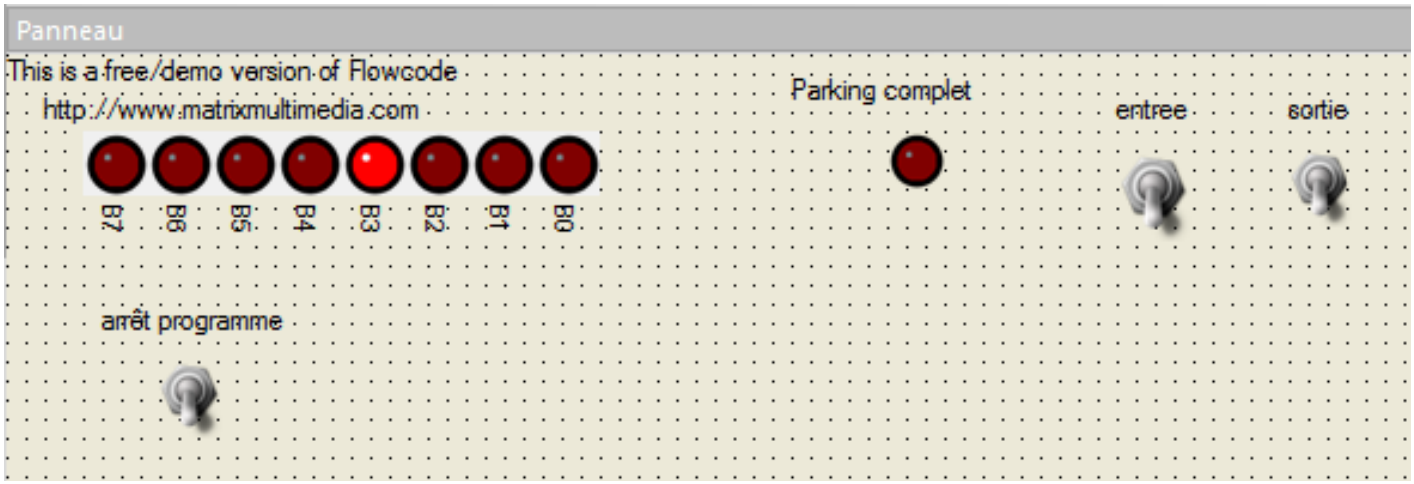
Essayez votre programme et quand tout fonctionne faites valider par le professeur

VI. Gestion d'un parking

Cahier des charges :

- objectif du programme : gestion d'un parking de 10 places
- le programme contiendra une boucle principale arrêtée par un interrupteur 'arrêt programme'
- lorsqu'une voiture entre (s'il y a de la place), elle active un capteur. Ce capteur sera simulé par un interrupteur 'entrée'.
- lorsqu'une voiture sort, elle active un autre capteur. Ce capteur sera simulé par un interrupteur 'sortie'.
- Chaque fois qu'un véhicule entre ou sort, il est pris en compte. Le nombre de véhicules présent dans le parking est affiché sur des leds (ou des afficheurs 7 segments)
- Lorsque que le parking est plein (10 véhicules), une led 'parking complet' s'allume (il ne peut plus alors rentrer de véhicules car les système de lecture de badge, non géré ici, sera bloqué).

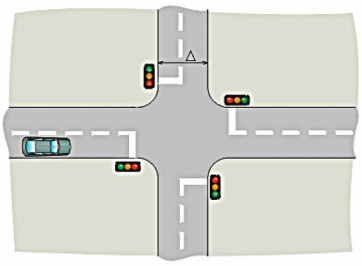
Voici une idée pour le panneau de simulation :



Essayez votre programme et quand tout fonctionne faites valider par le professeur

VII. Feux de carrefour

Cahier des charges :



- Vous allez concevoir la gestion des feux de carrefour d'une intersection de 2 routes.
- En temps normal, le temps de passage des voitures (au vert) dure 10 secondes sur la route prioritaire et 5 sur l'autre route
- Le feux orange dure 2 secondes
- Le feux vert piéton fonctionne pendant le feu rouge et le feux rouge piéton pendant les feux vert et orange
- La nuit (une horloge gère le passage jour/nuit. On simulera cette horloge par un interrupteur) les feux oranges clignotent (2 secondes)
- Une des voie est une voie plus importante et donc prioritaire. Deux capteurs 'présence voiture' signalent qu'une voiture attend au feux de la route non prioritaire. S'il n'y a pas de voiture, les feux de la route prioritaire restent au vert (sauf en cas d'appel piéton que l'on simulera avec un interrupteur).

Essayez votre programme et quand tout fonctionne faites valider par le professeur