Benoît Girard
26 février 2019

Initiation à la Neuro-Robotique

Trucs à savoir sur blockly :

- Les tableaux : dans blockly, les variables sont par défaut stockées dans des tableaux. Lorsqu'on a besoin d'une seule variable, il faut déclarer un tableau de taille 1 :

  La numérotation des variables dans un tableau commence à 0, donc pour atteindre la première valeur, il faut écrire :

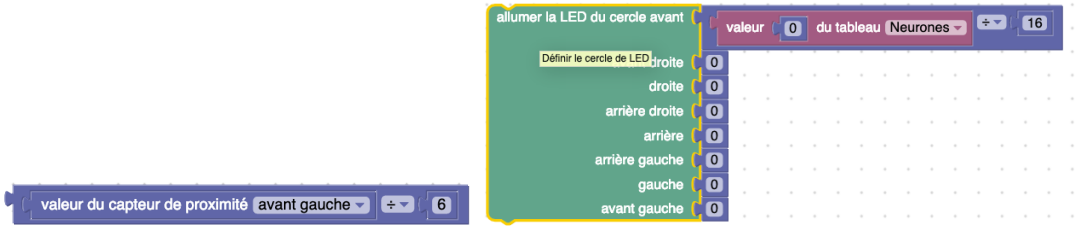
  Et pour un tableau de taille 4 :

Tableau Blockly de taille 4	
0	1ere valeur
1	2e valeur
2	3e valeur
3	4e valeur

Trucs à savoir sur le Thymio 2 :

- Les capteurs de distance prennent des valeurs de 0 (aucun obstacle détecté) jusqu'à plus de 3000 (si un obstacle est très près).
- Les moteurs prennent des valeurs de -500 (en marche arrière à toute vitesse) jusqu'à 500 (en marche avant à toute vitesse).
- Les LEDs orange prennent des valeurs entre 0 (éteintes) et 32 (allumées au maximum).

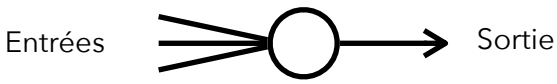
Pour avoir des valeurs cohérentes dans nos réseaux de neurones, on va donc toujours diviser les entrées des capteurs par 6 (pour qu'elles aillent, en gros, de 0 à 500), et quand on voudra afficher une activité avec une LED, on la divisera par 16 (pour qu'elles aillent de 0 à 31).



Benoît Girard
26 février 2019

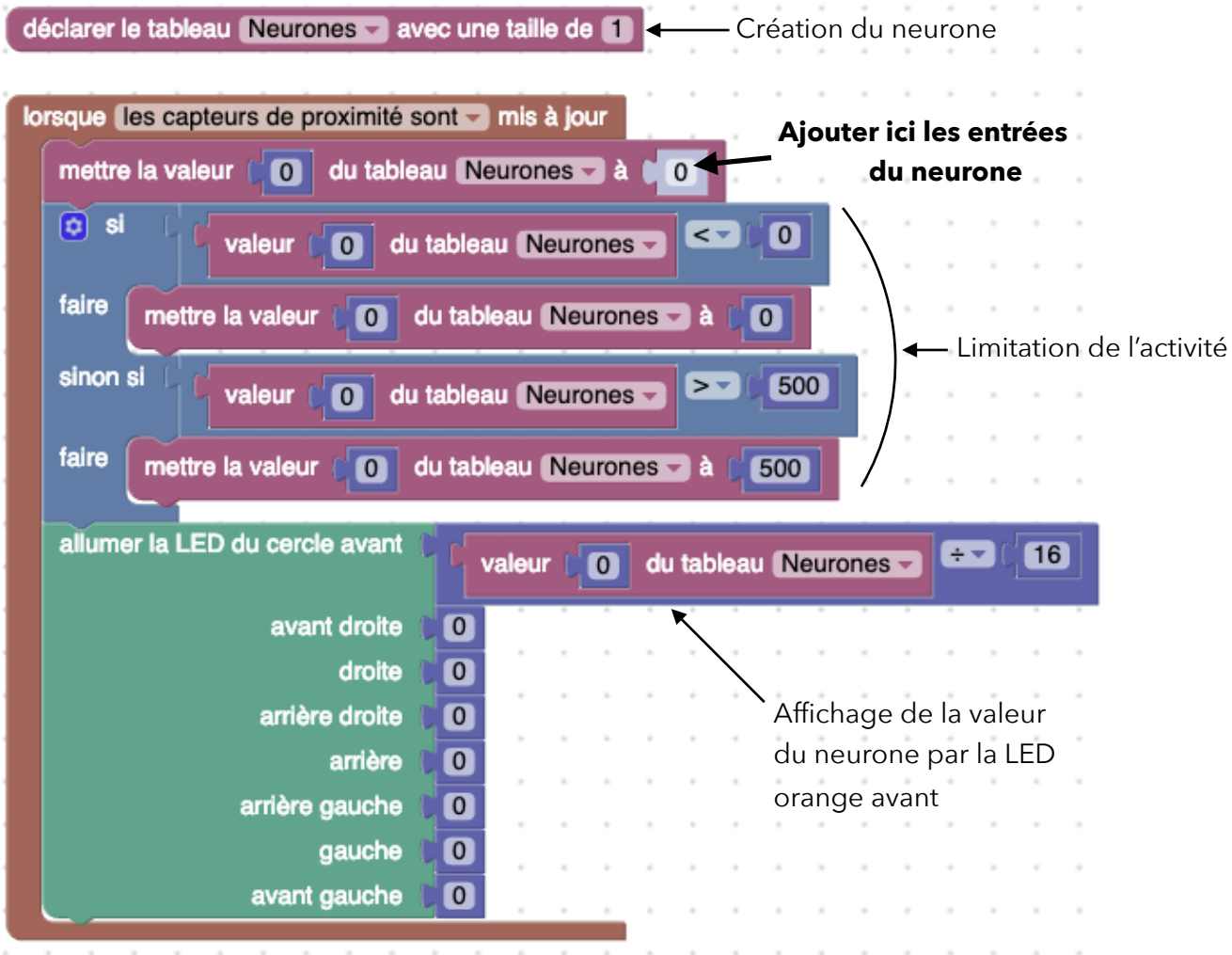
Programmer un premier neurone

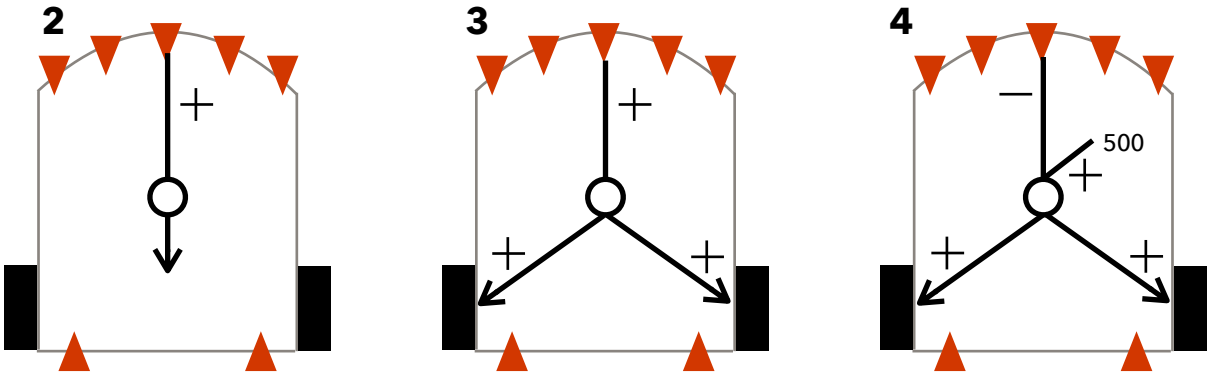
Le neurone artificiel simple que nous allons utiliser fait l'addition de valeurs d'entrée, les limite entre une valeur minimale (0) et une valeur maximale (j'ai choisi 500, comme la valeur maximale des moteurs) et les envoie ensuite au neurone suivant.



Les valeurs envoyées peuvent être positives (+) ou négatives (-).

Etape 1 : charge le fichier Etape1-Neurone.aesl, il contient déjà un neurone, et permet de visualiser son activité. Le programme devrait ressembler à ça :





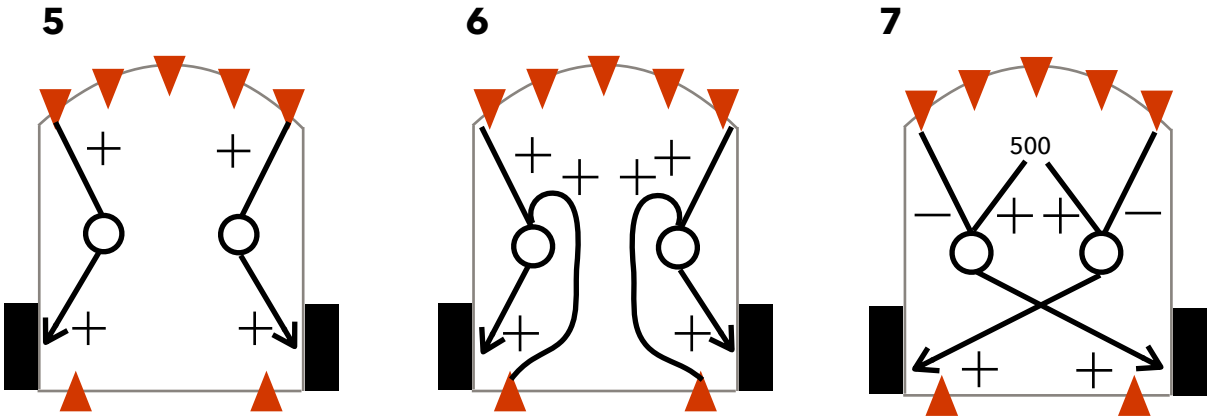
Etape 2 : Ajoute une entrée sensorielle à ce neurone, par exemple la valeur d'un capteur de proximité.

Etape 3 : Envoie sa sortie aux moteurs : selon la valeur du capteur, le robot avance plus ou moins vite. Notre robot a son premier (petit) cerveau ! Pour l'instant, il n'est pas très malin, il fonce sur les obstacles…

Etape 4 : On pourrait, au contraire vouloir qu'il avance toujours, sauf quand il y a un obstacle : on va ajouter à notre neurone une entrée positive à 500, lui soustraire la valeur du capteur, et envoyer tout ça dans les moteurs.

Deux neurones

Si on ajoute un deuxième neurone, on va pouvoir agir différemment sur chaque moteur, et donc permettre d'avancer, mais aussi de tourner.



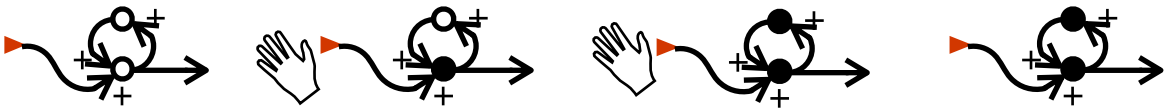
Etape 5 : Ajoute un deuxième neurone, connecte le neurone 0 au capteur avant-gauche en entrée et au moteur gauche en sortie, et le neurone 1 au capteur avant-droit en entrée et au moteur droite en sortie. Comment se comporte ce nouveau robot si tu approches ta main ? Et si tu le places dans un couloir ?

Etape 6 : Ajoute aux entrées de tes neurones la valeur des capteurs arrière : le robot avance lorsqu'on s'apprête à le pousser. Tu peux le faire poursuivre par un autre robot, de l'étape 3…

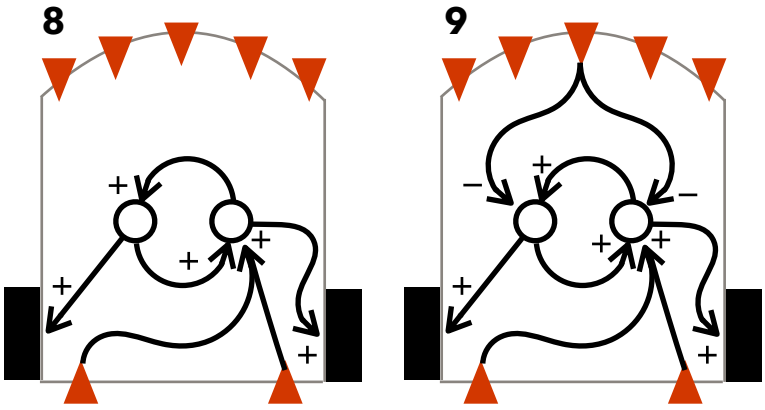
Etape 7 : Que se passe-t-il si, comme pour l'étape 4 du robot à un seul neurone, tu stimules chacun de tes deux neurones avec une entrée à 500, et que tu soustrais à chacun la valeur de ses capteurs d'entrée ? Et que se passe-t-il si tu croises les connexions vers le moteurs (le neurone qui reçoit les entrées de droite active le moteur gauche, et inversement) ?

Activité interne avec des neurones : une mémoire

Le robot de l'étape 6 avance si on place sa main derrière pour le « pousser », mais il s'arrête lorsqu'on enlève la main. Si on lui donne une petite mémoire, il pourrait se souvenir qu'il a été poussé, et continuer à avancer jusqu'à ce qu'il rencontre un obstacle !



Pour faire une mémoire simple, on peut connecter deux neurones excitateurs en boucle : le premier neurone est excité par la détection de la main, il active alors le second neurone, qui active ensuite le premier, cette boucle d'excitation fait que si on enlève la main, les deux neurones continuent à s'exciter l'un-l'autre.



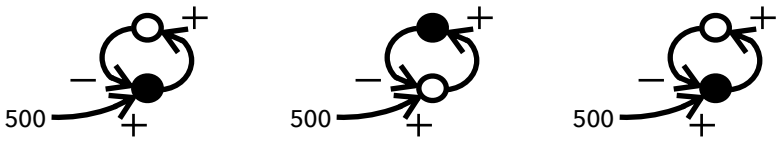
Etape 8 : Programme un robot qui avance quand on le pousse, et qui continue à avancer tout seul quand on enlève la main.

Etape 9 : Ajoute une inhibition issue du capteur de proximité avant, qui désactive la mémoire et arrête le robot lorsqu'un obstacle apparaît devant.

Alors que les robots de l'étape 1 à 7 étaient « réactifs », c'est-à-dire qu'ils ne faisaient que réagir instantanément aux changements de son environnement, notre robot a maintenant une activité interne qui lui est propre, sa petite boucle de mémoire.

Activité interne avec des neurones : des oscillations

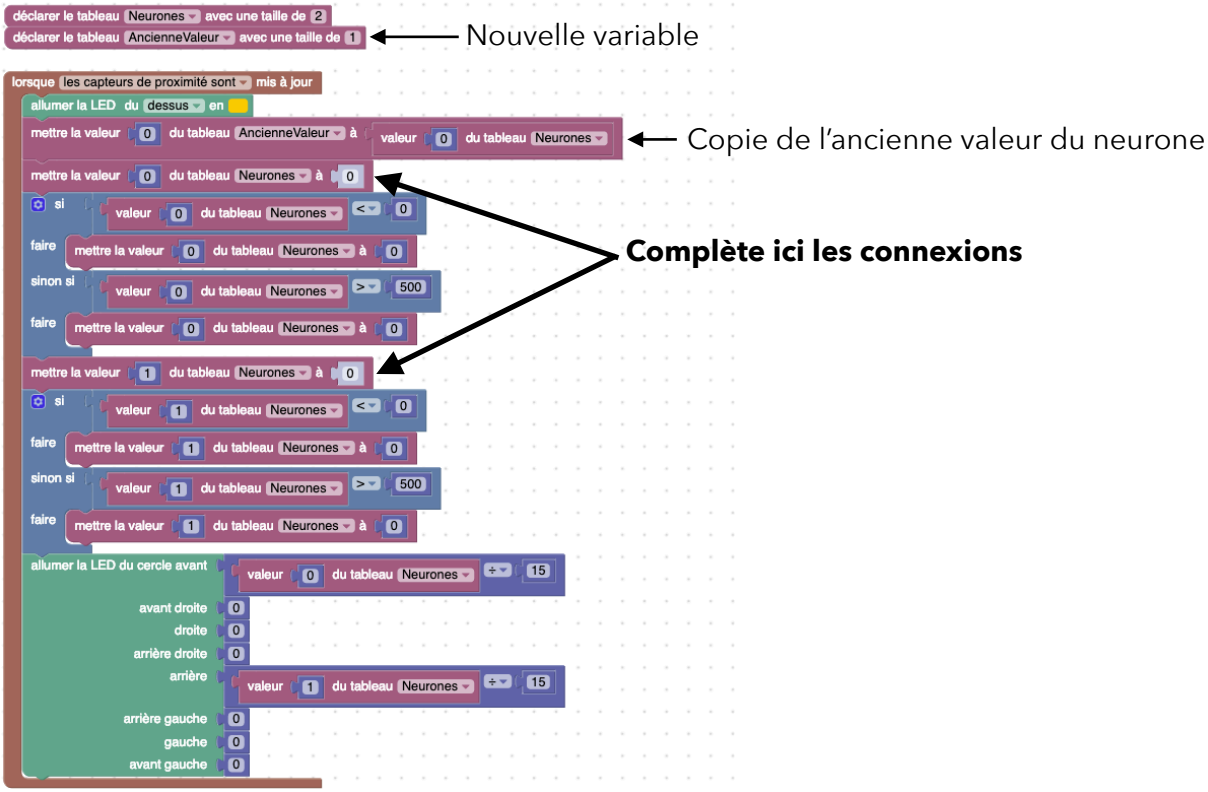
Notre mémoire, lorsqu'elle est activée, a une valeur qui ne change pas au cours du temps. Si on met une soustraction dans notre boucle, on peut aussi créer une activité durable, mais qui varie dans le temps.

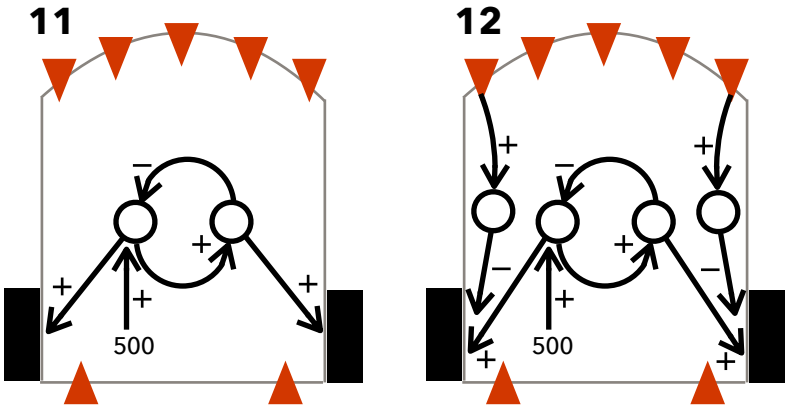


Pour faire marcher ce réseau, il faut en revanche surmonter une difficulté technique : dans un vrai réseau de neurones, toutes les activités sont calculées en même temps, à partir des activités du pas de temps précédent. Dans notre cas particulier on d'abord calculer la valeur du premier neurone, puis celle du deuxième, mais le deuxième neurone va alors utiliser la nouvelle valeur du premier neurone, celle qu'on vient tout juste de calculer, pas celle du pas de temps d'avant. Si on fait ça, notre oscillateur ne marchera pas.

La solution, c'est de créer une nouvelle variable, « AncienneValeur » pour stocker l'ancienne valeur du neurone 1, de calculer la nouvelle valeur du neurone 1, et enfin de calculer la nouvelle valeur du neurone 2, en utilisant « AncienneValeur ».

Etape 10 : Charge le fichier Etape10-Oscillateur.aesl, il contient déjà cette nouvelle variable et cette opération de copie de valeur. Tu n'as plus qu'à connecter les neurones et à voir l'oscillation sur les LEDs oranges.





Etape 11 : Connecte la sortie d'un neurone au moteur droit, l'autre au moteur gauche : le robot devrait avancer en faisant de petits pas alternativement à droite et à gauche.

Etape 12 : Comme à l'étape 7, ajoute des neurones inhibiteurs depuis les capteurs de proximité avant-gauche et avant-droit pour que le robot tourne lorsqu'il perçoit un obstacle d'un côté ou de l'autre.