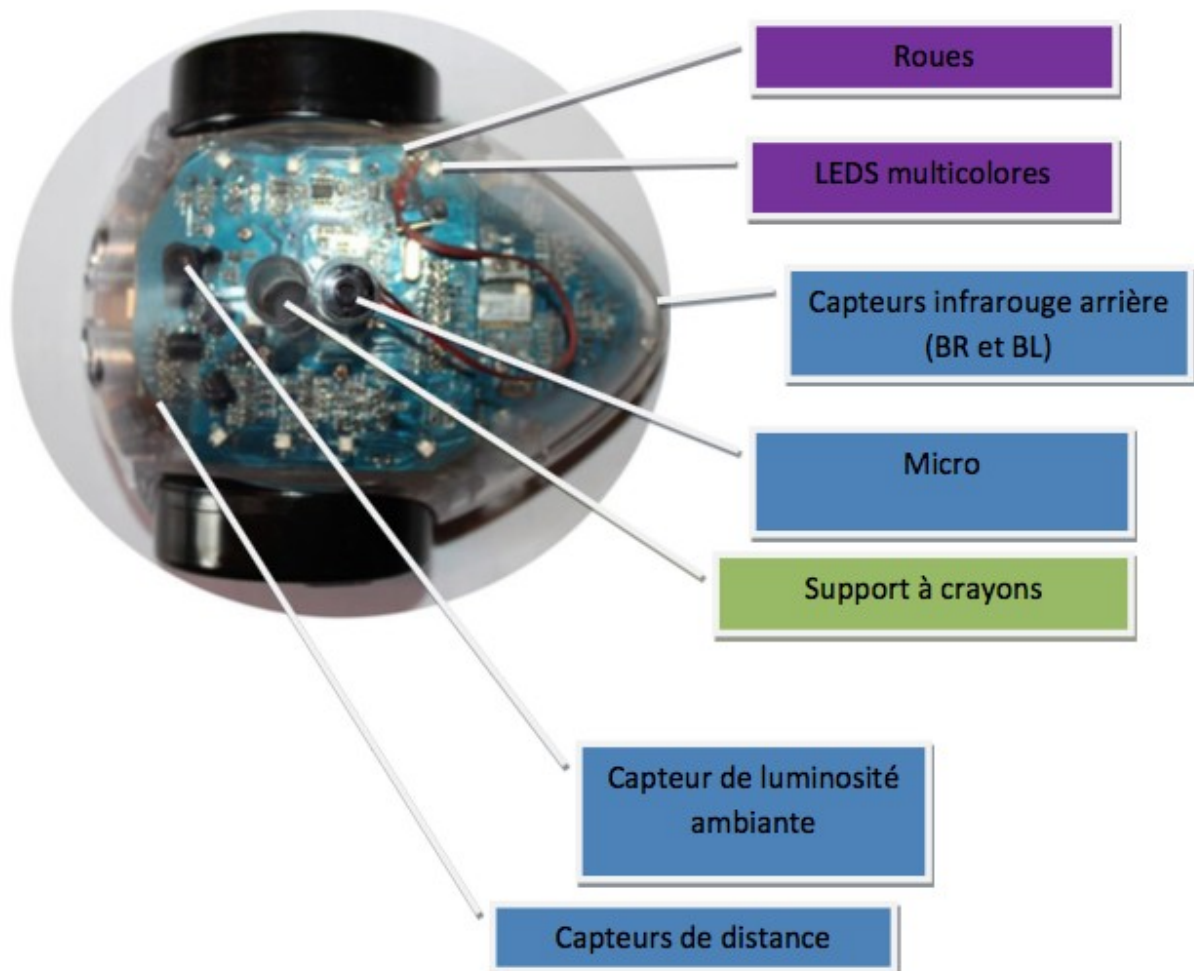
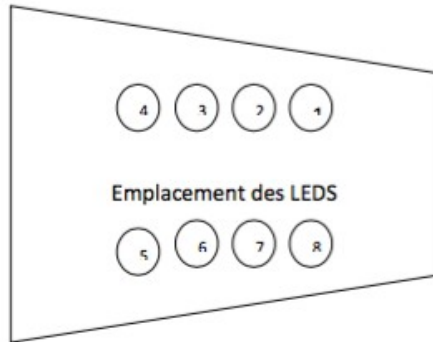


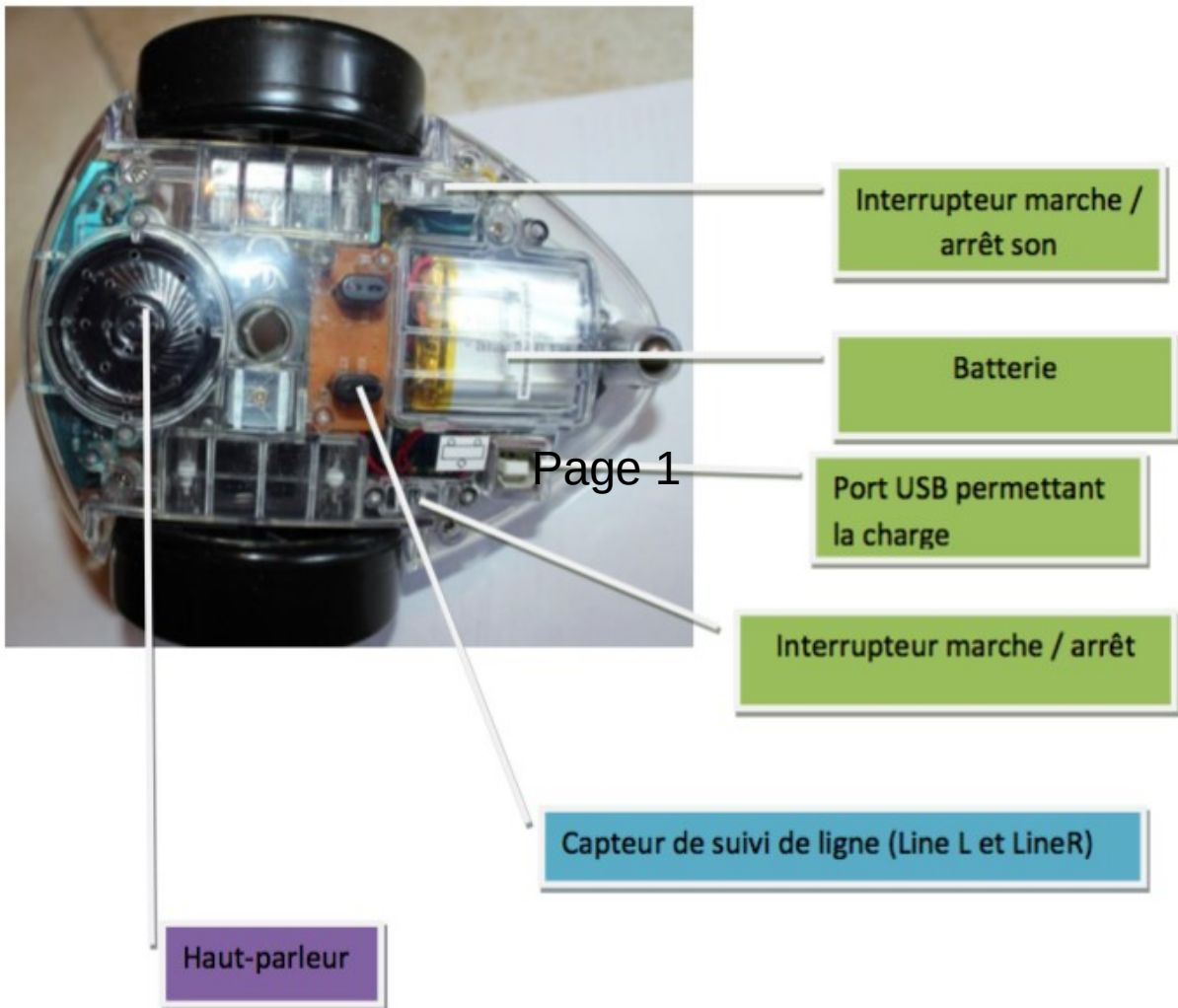
# Programmation CM avec Ino-Bot

D'abord voici une présentation de la machine.





Actuateurs Capteurs Support



Source Baptiste Girard Despraulex

## Ensuite on étudie les différents menus

Indique le robot connecté

Démarre et arrête le programme

Permet de sauvegarder

Ouvre un programme sauvegardé

Permet de sauvegarder ailleurs

Nom du programme

Permet de renommer le robot

Cliquer sur sauvegarder puis donner un titre

The screenshot shows a mobile application interface for Ino-Bot. At the top, a blue menu bar contains buttons: DÉCONNECTÉ, GO, ANNULER, RETABLIR, SAUVEGARDER, OUVRIR, EFFACER, CODE, PARTAGER, ?, and AUTOMATIC HEADLIGHTS. Below the menu, a 'Logic' block is visible in a workspace. A 'ROBOTS DISPONIBLES' window shows 'InO-Bot' with a 'CONNECT' button. Another 'ROBOTS DISPONIBLES' window shows 'InO-Bot' with 'RENAME' and 'DISCONNECT' buttons. An 'AIDE' window displays technical specifications for Ino-Bot blocks. A 'SAUVEGARDER LE PROJET' dialog box is open, showing 'SAUVEGARDER UN NOUVEAU PROJET' with the example name 'musique' and a timestamp 'Saved 01.06.2018 15:38'.

Cette étape est surtout pour l'enseignant, les enfants vont découvrir ces fonctions à mesure de la progression.

Condition de l'action

comparateur

Et /ou

Vrai ou faux

contraire

The screenshot shows the 'Logic' menu on the left side of the software interface. The menu items are: Logic, Loops, Math, InO-Bot, and Variables. The 'Logic' menu is expanded, showing several blue block icons with their corresponding labels: 'si faire' (Condition de l'action), '=', 'et' (comparateur), 'vrai' (Vrai ou faux), and 'pas' (contraire).

16:40 Jeu. 6 déc.

DÉCONNECTÉ

Logic

Loops

Math

InO-Bot

Variables

## boucles

répéter 10 fois faire

répéter tant que faire

sortir de la boucle

2 types de boucles :  
 - un nombre de fois  
 - conditionnelle

Sortir ou continuer avec  
 le prochaine itération

16:40 Jeu. 6 déc.

DÉCONNECTÉ

Logic

Loops

Math

InO-Bot

Variables

0

90° °

1 + 1

racine carrée 9

sin 45

$\pi$

0 est pair

arrondir 3.1

reste de 64 ÷ 10

contraindre 50 entre 1 et 100

random integer from 1 to 100

fraction aléatoire

Fonctions  
 mathématiques  
 ( compliquées  
 pour les élèves  
 du primaire)

16:40 Jeu. 6 déc.

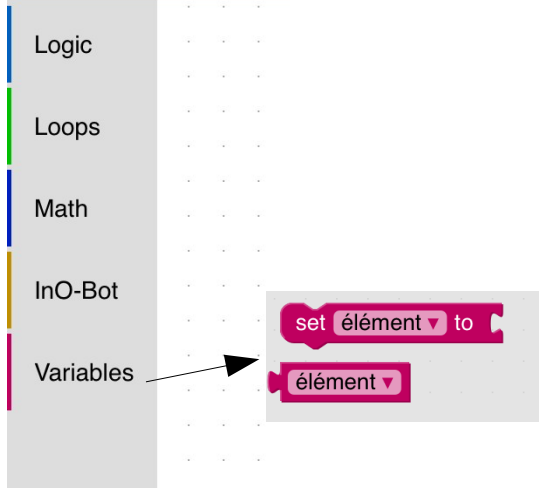
DÉCONNECTÉ

- Logic
- Loops
- Math
- InO-Bot
- Variables

The image shows a Scratch script for an InO-Bot robot. The script is composed of several blocks, each with a corresponding French annotation in a box:

- Light Level**: Niveau de lumière, distance...
- has obstacle**: Position obstacle
- robot is moving**: Le robot bouge
- left line sensor**: Le capteur droit ou gauche de ligne
- Motors forward slow 10 cm**: Avance vitesse et distance
- Turn 90° right slow**: Tourne : degré et sens ( droite ou gauche )
- Wait for 5 seconds**: Attendre en secondes
- Move forward slow**: Avance vitesse
- Stop Moving**: stop
- Front lights left: 0 right: 0**: Allumage des phares droit ou gauche
- RGB light # 1 red: 0 green: 0 blue: 0**: Allume les lampes de 1 à 8 avec des couleurs avec la possibilité de toutes les allumer
- RGB light (1-8) 1 red 0 green 0 blue 0**: Allume les lampes de 1 à 8 avec des couleurs
- External connector Forward**: Ne s'utilise que si le robot est connecté à un capteur externe
- IR Transmitter on**: Transmetteur infra rouge
- Move pen up**: Monte ou descend le crayon
- Play sound # 0**: Joue un son

DÉCONNECTÉ



The image shows the Scratch interface. On the left is a vertical menu with categories: Logic, Loops, Math, InO-Bot, and Variables. The 'Variables' category is highlighted in pink. An arrow points from the 'Variables' label to a 'set élément to' block in the workspace. Below it is another block labeled 'élément'.

Définir un élément et l'utiliser

## 1 ère séance

On commence par lire le programme sur les lumières de la machine.



N° de la  
LED

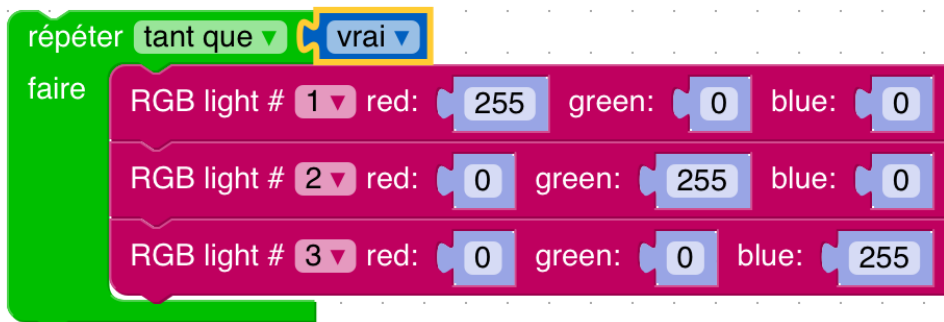
Valeur de la couleur  
Entre 0 et 255

Puis on valide ensemble ce que cela veut dire

Ensuite on teste sur le robot.

Problème cela s'allume et s'éteint tout de suite

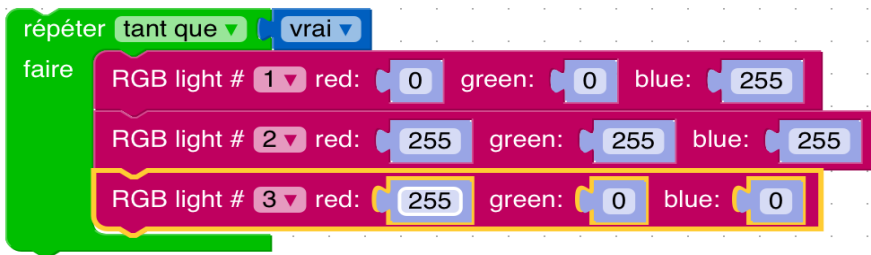
On va résoudre le problème grâce à une boucle perpétuelle.



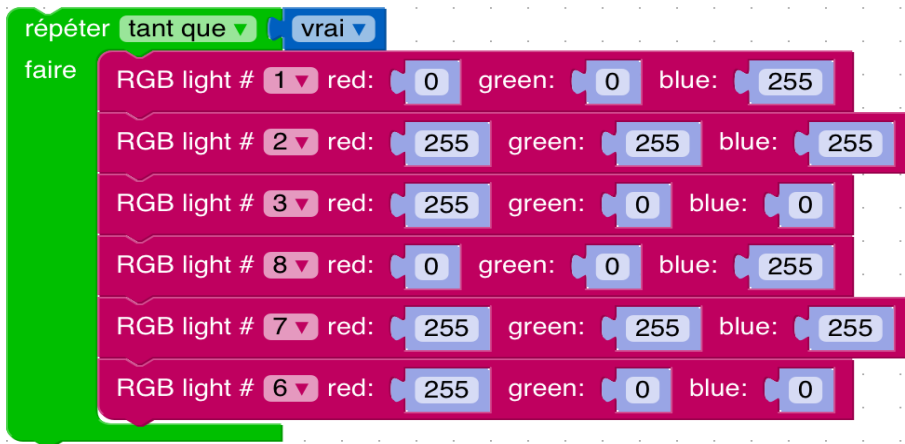
Ensuite on va leur demander de trouver :

Comment faire du violet, du jaune, du blanc...

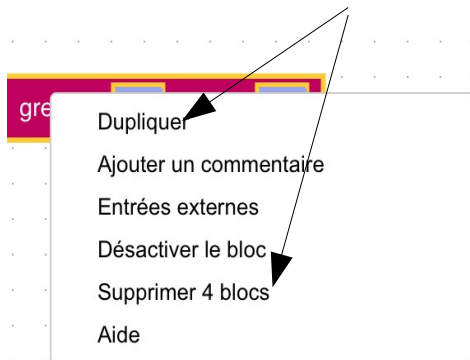
Puis on va demander d'allumer Les 3 premières LED en Bleu Blanc rouge et qu'elles restent allumer



Puis on va demander à ce que les LED bleu blanc rouge soient allumées des 2 cotés



On apprendra que l'on peut dupliquer ou supprimer une commande en maintenant le doigt sur celle-ci ( cela déclenche un menu contextuel)



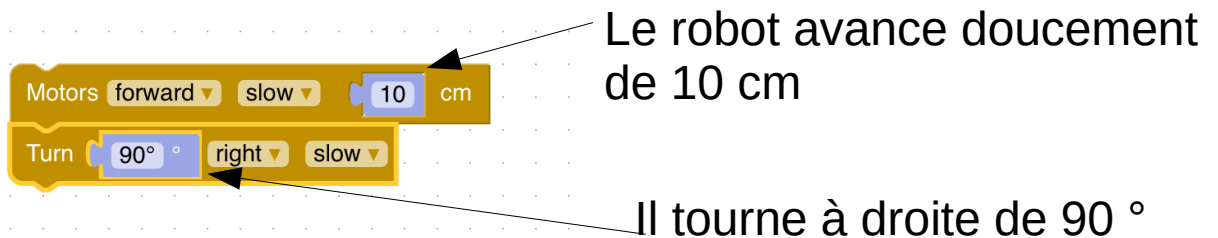
On peut ensuite demander à ce que cela clignote ( on rajoute des lignes où tout est à 0



## 2<sup>ème</sup> séance

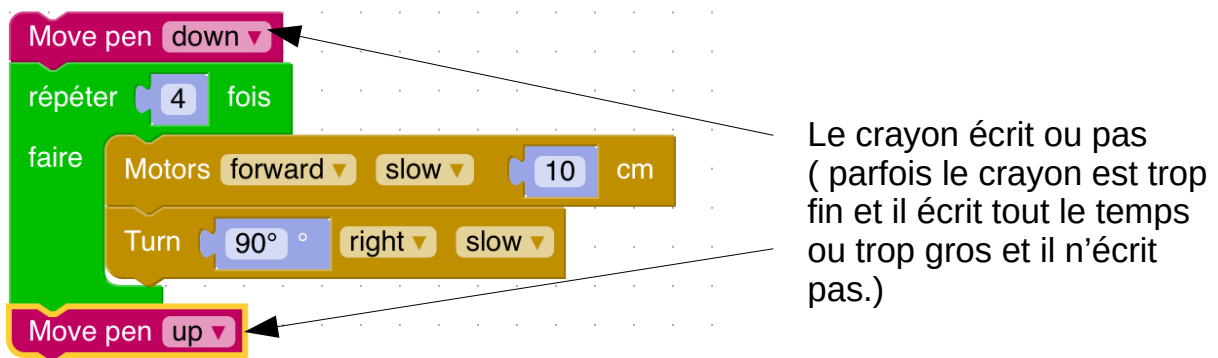
Le robot roule et dessine.

Dans cette séance on explore maintenant les commandes qui concernent les mouvements du Robot.

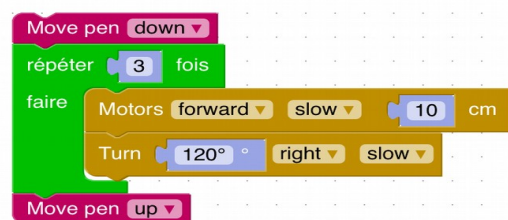
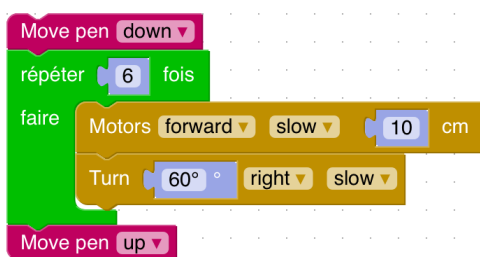


Il faut faire à ce moment faire une aide sur les angles et le cercle trigonométrique. On va devoir faire ensuite un carré et pour faire une figure fermée, il faut que la totalité des angles fasse 360°

Ensuite après avoir constaté qu'on peut mettre un crayon on va trouver le programme qui permet de tracer un carré. Il faudra utiliser une boucle. (4 X 90 = 360)



On peut faire un hexagone ( 6 X 60 = 360) , un triangle (3 X 120 = 360)



On pourrait imaginer quelque chose qui ressemble à un cercle en faisant (360 X 1) et en avançant de 1 cm dans ce cas il faut écrire quand on modifie le degré de l'angle. Mais on peut s'approcher en faisant (36 X 10) ou (72 X 5)

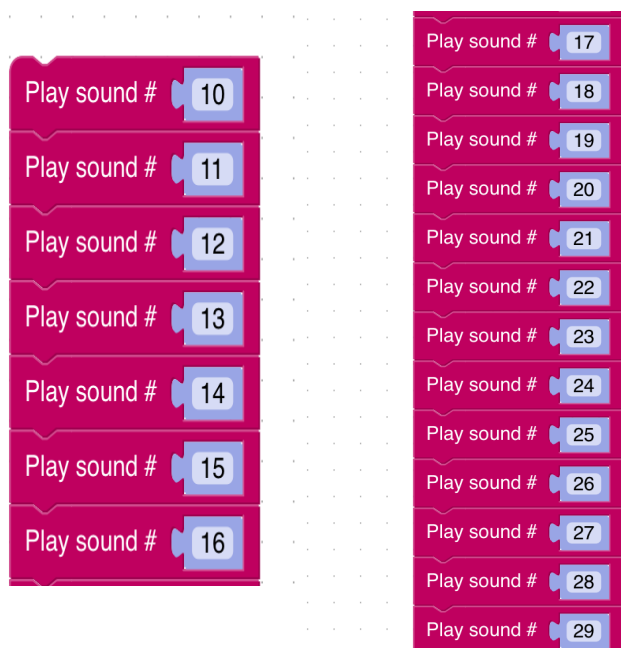
```
Move pen down
répéter 72 fois
faire
  Motors forward slow 2 cm
  Turn 5° right slow
Move pen up
```

On finit la séance en demandant à ce que le robot fasse un carré ou une autre figure en étant allumé en bleu blanc rouge.

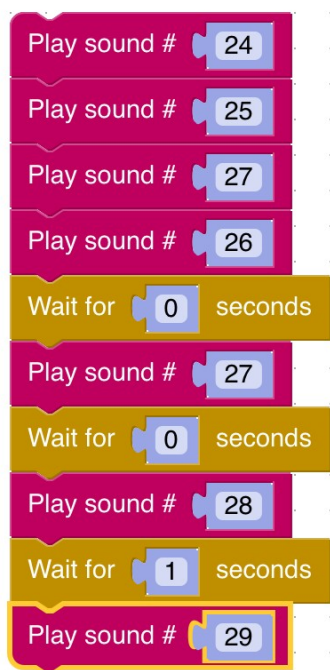
```
répéter 4 fois
faire
  RGB light # 1 red: 0 green: 0 blue: 255
  RGB light # 2 red: 255 green: 255 blue: 255
  RGB light # 3 red: 255 green: 0 blue: 0
  RGB light # 8 red: 0 green: 0 blue: 255
  RGB light # 7 red: 255 green: 255 blue: 255
  RGB light # 6 red: 255 green: 0 blue: 0
  Motors forward slow 10 cm
  Turn 90° right slow
```

### 3<sup>ème</sup> Séance : Son et musique

On va chercher dans les commandes le bloc qui commande le son.



Les sons de 0 à 9 sont des sons de robots  
Et de 10 à 16 ce sont des notes de piano de la gamme  
Et de 17 à 29 ce sont des notes de xylophone

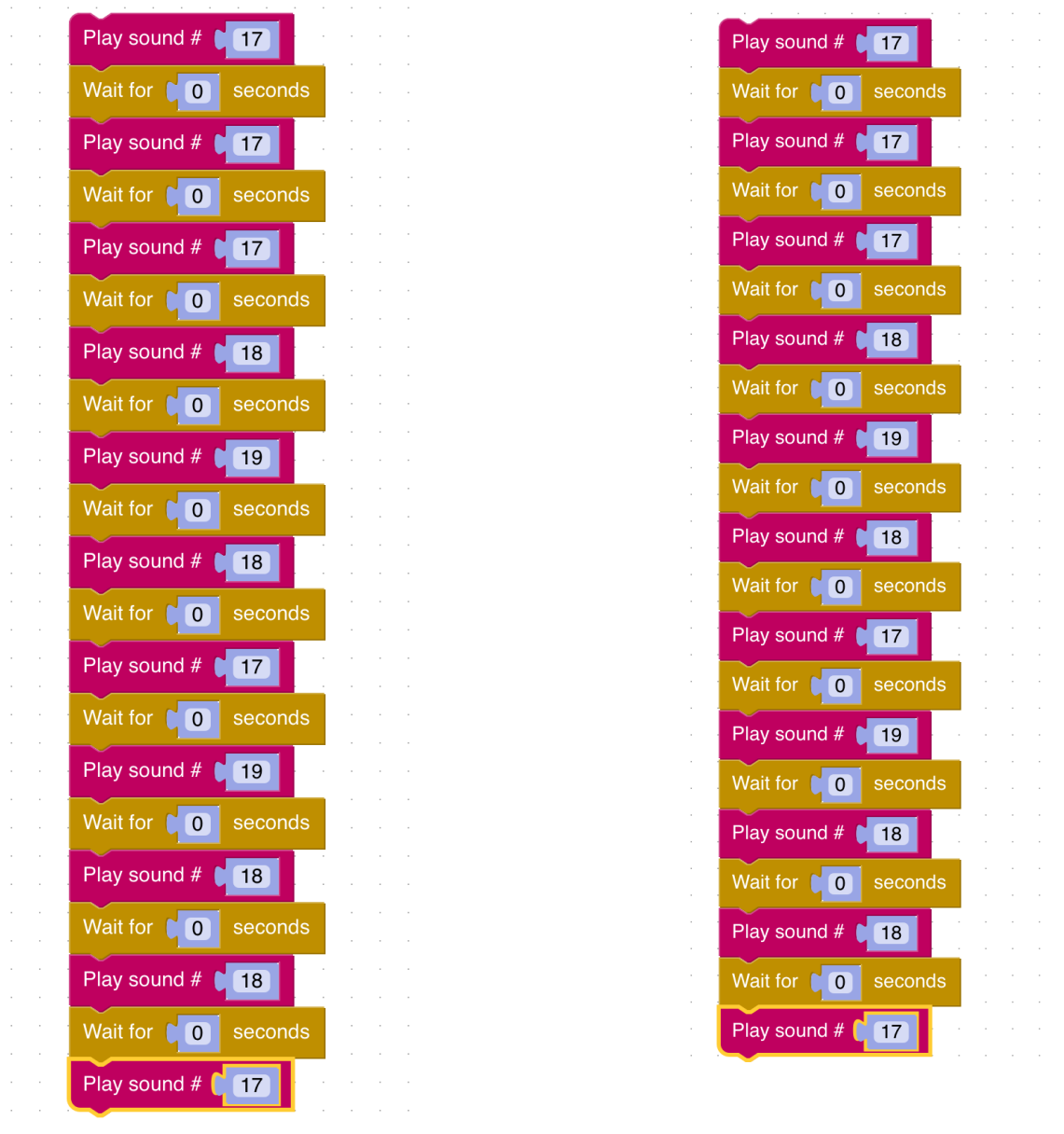


Comme les notes sont vite jouées on peut mettre des pauses

On peut demander à ce que le robot joue au clair de la lune (début) ou la marseillaise début

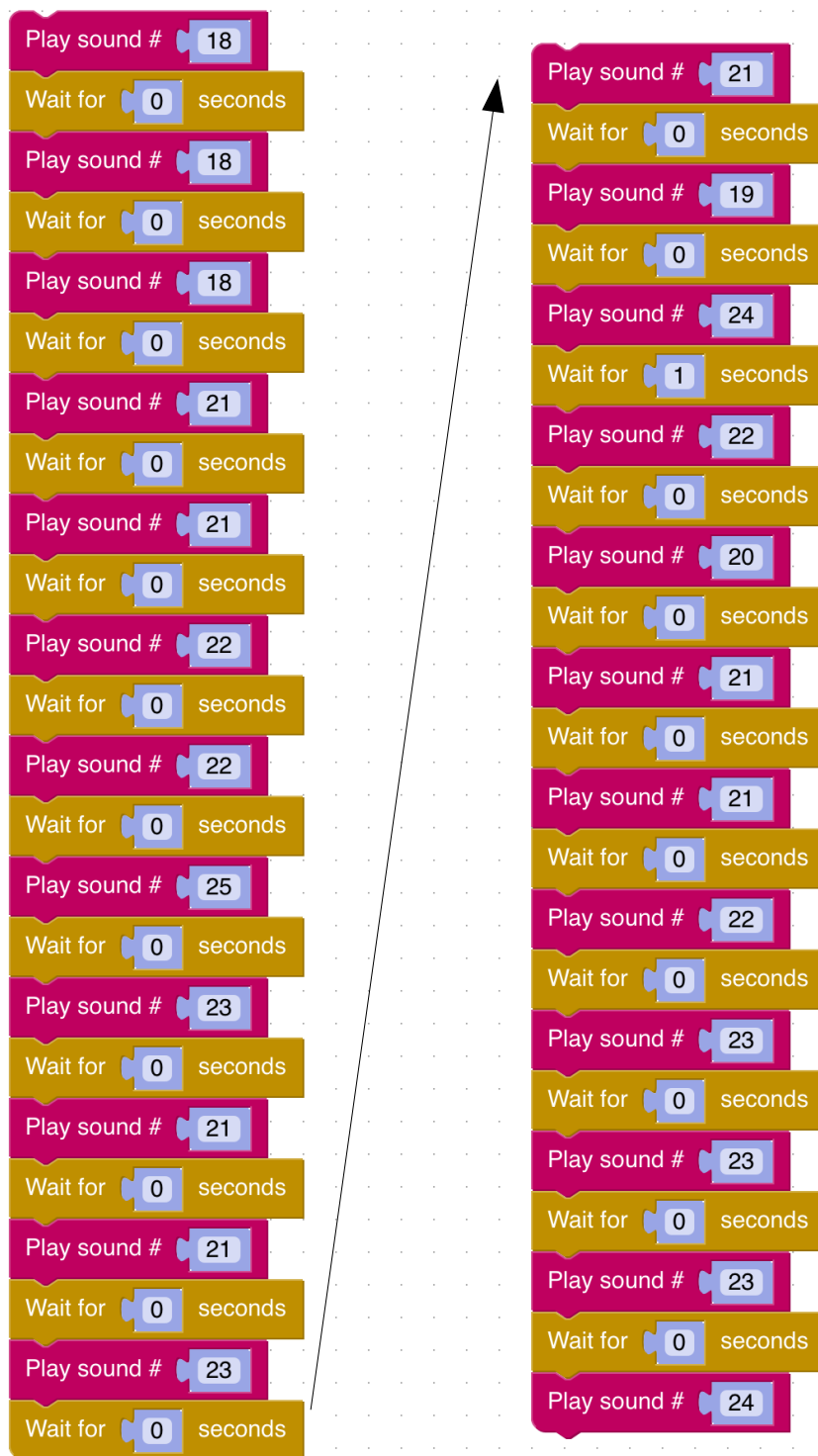
Au clair de la lune : Les notes sont : do do do ré mi ré do  
mi ré ré do

La correspondance sera donc



Pour la marseillaise le robot devra jouer ré ré ré sol sol la la ré\* si sol sol mi do\* la on s'arrête ici car il faut faire un fa#  
Ou on accepte le fa ( mais cela fait une fausse note.

On la joue qu avec les notes du xylophone ( il faut 2 octaves)

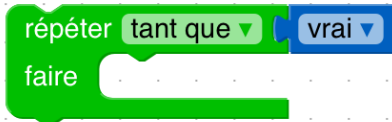


Ensuite on va demander à ce que le robot s'allume dessine et joue de la musique dans le même programme.

## 4<sup>ème</sup> séance les capteurs et les phares



On va apprendre que cette commande permet d'allumer les phares de manière indépendante.



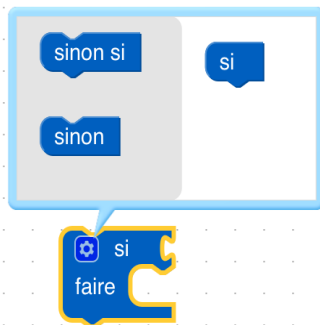
Si on rajoute cette commande ils peuvent rester tout le temps allumer

Ensuite on va voir la commande

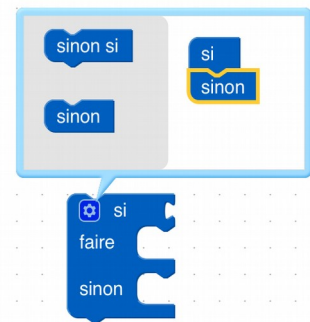


Elle commande les capteurs (avant droit ↗ avant gauche ↖  
arrière droit ↘ arrière gauche ↙ )

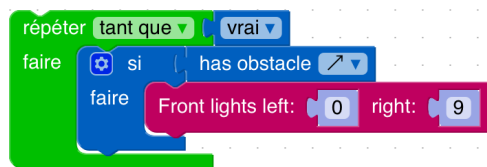
On va aussi introduire la commande de condition



Cela permet de mettre une condition et que faire si cette condition n'est pas remplie.



On va donc tester les capteurs avant avec un programme de ce type



Quand on approche un objet du capteur avant droit, le phare droit s'allume et reste allumé.

```

répéter tant que vrai
faire
  si has obstacle
  faire
    Front lights left: 0 right: 9
    Front lights left: 0 right: 0

```

Ici le phare droit s'étendra une fois qu'il n'y aura plus d'obstacle mais on peut faire varier phare droit ou gauche ou les deux.

```

répéter tant que vrai
faire
  si has obstacle et has obstacle
  faire
    Front lights left: 0 right: 9
    Front lights left: 0 right: 0

```

Même programme avec les 2 capteurs qui doivent détecter ensemble

```

répéter tant que vrai
faire
  si has obstacle ou has obstacle
  faire
    Front lights left: 0 right: 9
    Front lights left: 0 right: 0

```

Ici c'est l'un ou l'autre

```

répéter tant que vrai
faire
  si has obstacle ou has obstacle
  faire
    Front lights left: 0 right: 9
    Front lights left: 0 right: 0
    Play sound # 0
    RGB light # All red: 255 green: 0 blue: 0

```

Ici on rajoute du son et lumières de couleur on peut faire varier tous les paramètres

```

répéter tant que vrai
faire
  si has obstacle ou has obstacle
  faire
    Front lights left: 0 right: 9
    Front lights left: 0 right: 0
    Play sound # 0
    RGB light # All red: 255 green: 0 blue: 0
    RGB light # All red: 0 green: 0 blue: 0

```

Tout s'éteint à la fin

```

    répéter tant que vrai
    faire
      si has obstacle ou has obstacle
      faire
        Motors reverse slow 5 cm
        Front lights left: 0 right: 9
        Front lights left: 0 right: 0
        Play sound # 0
        RGB light # All red: 255 green: 0 blue: 0
        RGB light # All red: 0 green: 0 blue: 0
  
```

Maintenant quand il y a un obstacle il recule.

```

    répéter tant que vrai
    faire
      si has obstacle ou has obstacle
      faire
        Front lights left: 0 right: 9
        RGB light # All red: 255 green: 0 blue: 0
        Play sound # 0
        Front lights left: 0 right: 0
        RGB light # All red: 0 green: 0 blue: 0
        Motors reverse slow 10 cm
        Turn 90° right slow
      sinon
        Motors forward slow 10 cm
  
```

Avec « si et sinon » le robot recule et tourne s'il trouve un obstacle (en plus de ce qu'il faisait avant).

```

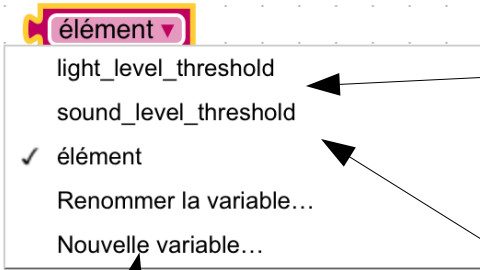
    répéter tant que vrai
    faire
      si Distance < 25
      faire
        Stop Moving
        Play sound # 1
        Move 10 cm backward at speed 10
        Turn 90 right slow
      sinon
        Move forward slow
  
```

Le robot roule. S'il trouve un obstacle à 25 cm, il s'arrête, il recule de 10 cm tourne de 90° et s'il n'y a plus d'obstacle, il repart.

Pour finir on peut imaginer des situations en combinant tout ce que l'on sait faire.

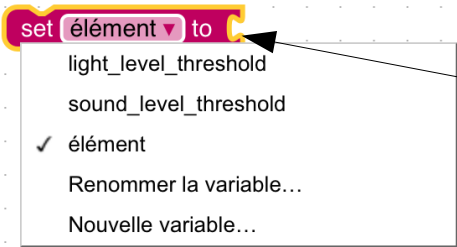


# Pour aller plus loin on peut définir des éléments



Ici on a défini le seuil de lumière ( light\_level\_threshold) ou le seuil de son ( sound\_level\_threshold)

Pour cela on a défini une nouvelle variable



Cela permet ensuite de créer des programme en fonction des seuils qu'on a défini

```
set sound_level_threshold to 30
set light_level_threshold to 25
répéter tant que vrai
faire
  si Sound Level ≥ sound_level_threshold
  faire
    RGB light # 4 red: 0 green: 255 blue: 0
    RGB light # 5 red: 0 green: 255 blue: 0
  si Sound Level < sound_level_threshold
  faire
    RGB light # 4 red: 0 green: 0 blue: 0
    RGB light # 5 red: 0 green: 0 blue: 0
  si Light Level < light_level_threshold
  faire
    Front lights left: 10 right: 10
  si Light Level ≥ light_level_threshold
  faire
    Front lights left: 0 right: 0
```

Voici un exemple avec le niveau de son et de lumière qui sera défini avant l'exécution du programme par la machine. Ce programme est enregistré par défaut dans l'application.

