

Programmer  
avec **Python**  
et Thonny

# SESSION 1

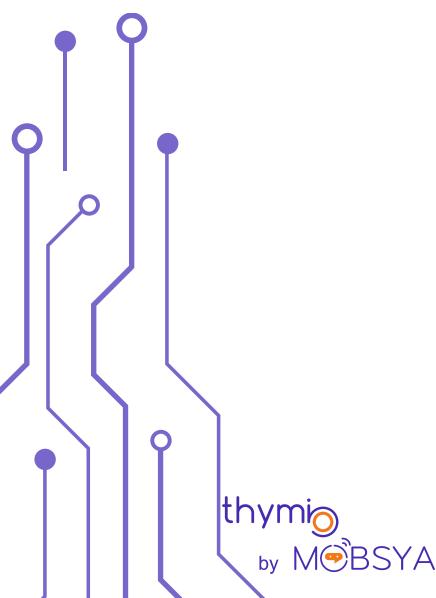
LE CAHIER  
D'ACTIVITÉS

**Les Sujets**

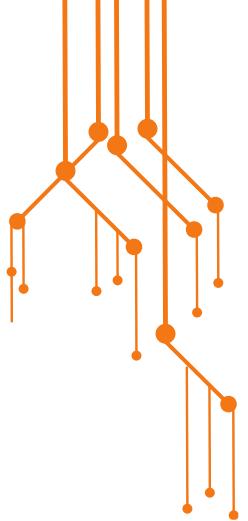


**Les conseils  
et astuces**

**Les solutions**



PAR: **JOËL RIVET**

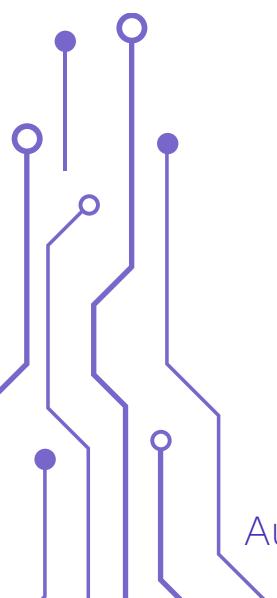


# SESSION 1

LE CAHIER  
D'ACTIVITÉS

## Les Sujets

P1 - P10



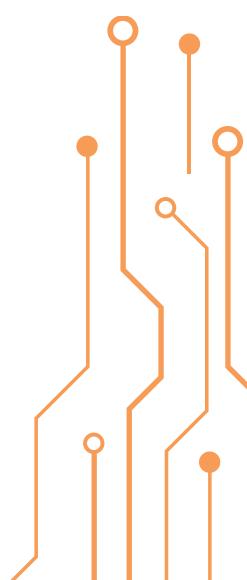
Auteur: JOËL RIVET

# LES SUJETS

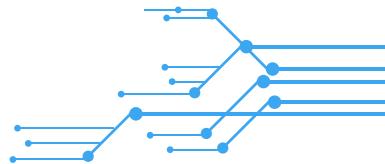
- 01** Thymio le craintif
- 02** Thymio l'explorateur
- 03** Le labyrinthe
- 04** Le jeu du foulard
- 05** Le relais des thymios
- 06** La voiture autonome
- 07** Ne restez pas là!
- 08** Garder la piste
- 09** Atterrissage
- 10** Démarrage codé

## ANNEXE 1

### Faire des murs



# Introduction



## Prérequis

Ces activités sont des applications du cours de la session 1, il est donc fortement conseillé d'avoir vu les chapitres 1 à 5.

## Aide

Si vous éprouvez des difficultés ou si vous voulez comparer vos choix, vous pouvez consulter le document "Session 1 - 10 activités - Aide"

## Solutions

Le document "Session 1 - 10 activités - Solution" contient le code de solutions pour les 10 activités. Il est évidemment tout à fait possible d'élaborer d'autres codes qui répondent aux consignes des sujets.

## Durée

Il est difficile de fixer une durée pour la rédaction du code d'une activité, cela dépend du programmeur, du soin qu'il apportera à son code (clarté, documentation, débogage et robustesse ...), de son expérience, de la difficulté du sujet, de son recours à l'aide ou non ... Pour fixer les idées, une activité pourrait prendre grossièrement de quelques dizaines de minutes à 2 heures.

## Mode filaire ou wireless

Thymio peut être programmé avec un câble USB ou une clé USB sans fil.

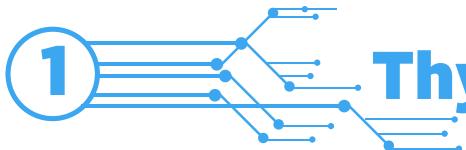
Dans le cas d'une liaison filaire, si on veut, une fois le programme chargé dans Thymio, on peut débrancher la connexion sans que cela empêche Thymio d'exécuter le programme.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets





## Thymio le craintif



1

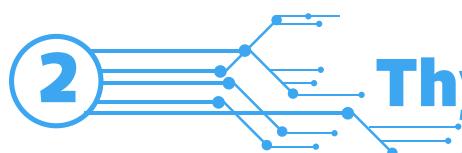
Difficulté

Nombre de robots

Vous connaissez peut-être le comportement préprogrammé rouge dit 'peureux'. Il s'agit ici de reproduire ce comportement avec les précisions suivantes :

- Si aucun obstacle n'est visible du robot, il reste immobile.
- si un objet survient à l'avant gauche (resp. droit), le robot recule en tournant vers la droite (resp. gauche).
- si un objet survient à l'arrière gauche (resp. droit), le robot avance en tournant vers la droite (resp. gauche).
- dans le cas où le robot est cerné de toute part, il se met à tourner sur lui même en rouge et en émettant le son système n°4.

Pour diminuer le nombre de cas possibles, on peut ne pas utiliser les capteurs avant 1 et 3.



## Thymio l'explorateur



1

Difficulté

Nombre de robots

Initialement Thymio est immobile sur une table, entouré d'objets de toutes tailles. Veillez néanmoins à ce que les objets ne soient pas trop rapprochés. Ils doivent être assez lourds pour ne pas être poussés par Thymio ou alors les tenir au passage du robot à la main.

S'il rencontre un obstacle, il continue à progresser en l'évitant par la droite ou la gauche. S'il rencontre le vide (bord de table par exemple), il s'arrête. Si le robot ne peut pas évoluer sur une table, on peut simuler le vide avec une zone de sol noir.

Options possibles :

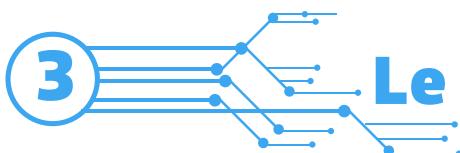
- Associer une couleur différente pour chaque cas.
- démarrer le robot avec l'appui sur un bouton plutôt qu'au lancement du script.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets





## Le Labyrinthe



1

Difficulté

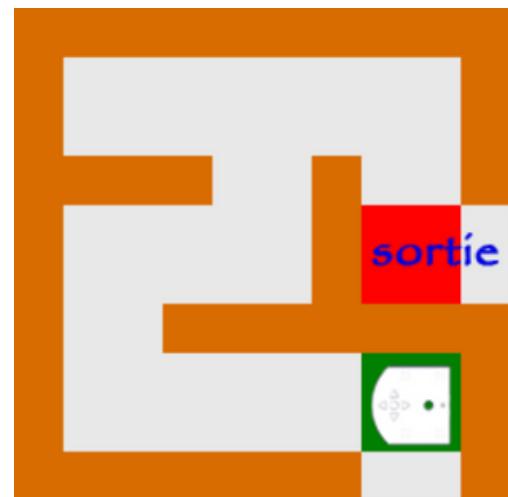
Nombre de robots

### Le défi

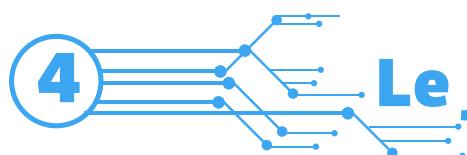
Thymio est dans un labyrinthe et doit en sortir.

### Les conditions matérielles

Construire un labyrinthe réel est très "coûteux" en accessoires (murs). On peut se contenter d'un labyrinthe très simple comme ci-contre ou utiliser le simulateur. La technique de navigation retenue sera la plus simple. Elle est dite "de la main gauche" car il suffit de suivre la continuité de la paroi avec la main gauche pour être sûr de trouver la sortie...



Une restriction, il faut que le labyrinthe soit dit 'parfait' c'est-à-dire qu'il ne comporte pas d'îlot. Il suffira donc à Thymio de suivre un mur ...



## Le jeu du foulard



5 Max

Nombre de robots

### Le défi : le jeu du foulard (un vieux jeu de scoutisme)

Deux adversaires sont face à face. Ils ont chacun dans leur dos un foulard passé dans la ceinture ou le pantalon. Les deux adversaires se tournent autour pour attraper le foulard de l'autre. Le premier qui y arrive a gagné.

Adaptions ce jeu à 2 Thymios. On remplace le foulard par les capteurs arrière. Le Thymio qui voit un de ses capteurs arrières s'allumer a perdu.

### Déroulement du jeu

L'environnement peut être ouvert ou fermé par des murs, mais l'espace interne doit être assez large. Si murs il y a, un Thymio qui voit un de ses capteurs allumé par un mur a perdu. En pratique, jouer avec des murs est préférable car l'expérience montre que 2 joueurs peuvent se tourner autour pendant ...très longtemps sans s'atteindre.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets



Les Thymios sont contrôlés par une télécommande. Il y a plusieurs modèles mais elles utilisent les mêmes codes.

Si l'activité est partagé sur plusieurs robots (5 max), ou peut créer une arène en y mettant tous les robots. Un robot qui "perd son foulard" est retiré de l'arène. Le vainqueur est le dernier restant.

En cas de combat à plusieurs, il faut un maître de séance qui :

- donne le démarrage de tous les robots,
- surveille qu'aucun joueur n'intervient avec les mains, sauf pour retirer son robot
- vérifie que les robots perdants sont bien retirés

### Contraintes sur la programmation.

Les touches de télécommandes sont utilisées pour démarrer le robot, l'arrêter et contrôler sa direction.

Pour utiliser la télécommande, ajouter un événement :

```
@onevent
def rc5():
```



puis utiliser les codes des touches avec la variable `rc5_command`. Par exemple, if `rc_command == 85` se produit si on appuie sur la touche de gauche.

Par ailleurs, la télécommande ancienne nous impose une limitation importante : une même télécommande commande tous les robots présents dans la pièce. Pour contrôler des robots différents, il faut donc utiliser des touches différentes. Avec les touches de la télécommande de Thymio, on peut envisager de contrôler 3 robots indépendants (voir figure précédente).

Pour la télécommande plus récente (modèle en image ci-dessus), il est possible de commander 5 robots avec 5 télécommandes de manière indépendante.

Tableau des codes des touches :

mouvement	Robot 1	Robot 2	Robot 3
avancer	87	2	8
tourner à gauche	85	1	7
tourner à droite	86	3	9
reculer	81	5	0
s'arrêter	80	6	16

Dans le cas où un des capteurs arrière est touché (capte un obstacle), le robot concerné doit impérativement :

- mettre sa variable vitesse (`speed`) à 0
- arrêter les moteurs avec `motor_..._target = 0`
- émettre un son aigu de durée 3 secondes

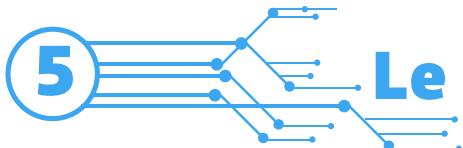
Très important : le seuil de détection d'un obstacle à l'arrière doit être le même pour tous les robots. Ce ne sera pas nécessairement 0, il faudra faire des essais.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets





# Le relais des Thymios



Plusieurs

Difficulté

Nombre de robots

## Le défi

Faire collaborer des Thymios dans un relais

Dans ce défi, il a été décidé qu' une certaine distance doit être parcourue, non pas un seul, mais plusieurs robots, qui devront donc collaborer. Comment faire pour rejoindre la ligne d'arrivée ?"

## Mise en oeuvre

L'idée est d'utiliser plusieurs robots en relais, comme dans les courses à pied, pour atteindre la ligne d'arrivée. Il y a donc un Thymio à l'arrêt sur la ligne de départ. Les autres sont répartis équitablement entre la ligne de départ et la ligne d'arrivée.

Evitez de consulter l'aide. Si vous êtes plusieurs équipes ayant un Thymio, vous aurez intérêt à collaborer pour que chacun puisse apporter une idée et élaborer ensemble une solution commune. Il va falloir trouver comment un Thymio peut déclencher le mouvement d'un autre Thymio.

Il y a devra y avoir deux programmes solution légèrement différents:

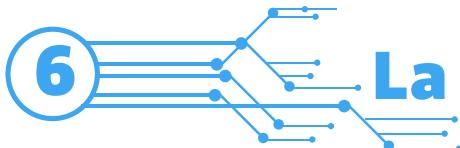
- thymio-relais 1 pour le 1er robots
- thymio-relais n pour les autres



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets





# La voiture autonome



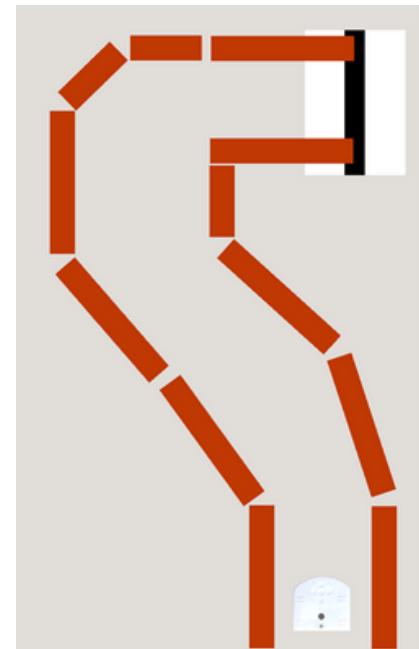
1

Nombre de robots

Une voiture autonome doit être capable de se déplacer en toute sécurité dans un environnement routier.

Pour cela, elle est équipée d'un grand nombre de capteurs qui lui communiquent en temps réel des données rendant compte non seulement de ce qui est immobile (bâtiments, végétation, superstructure en tout genre) mais également de ce qui est mobile (êtres vivants, météorologie, véhicules classiques, autres robots...)

Notre modèle de voiture autonome que nous nous proposons de programmer utilisera seulement ses 5 capteurs avant pour se déplacer. Le parcours sera un fragment de rue sinuuse bordée de murs (figure ci-contre). La fin de la rue sera matérialisée par une bande noire au sol. Thymio l'utilisera pour faire demi-tour.



De plus, un objet représentant un piéton pourra être placé dans la rue. Thymio devra s'arrêter devant et attendre que l'objet soit retiré pour repartir.

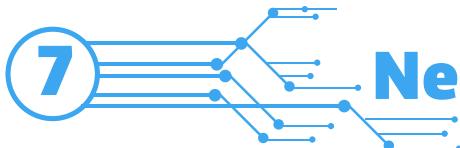
*Les murs peuvent être réalisés avec des feuilles A4 un peu épaisses pliées en 4 dans le sens de la longueur (voir Annexe 1).*



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets





# Ne restez pas là!



1

Difficulté

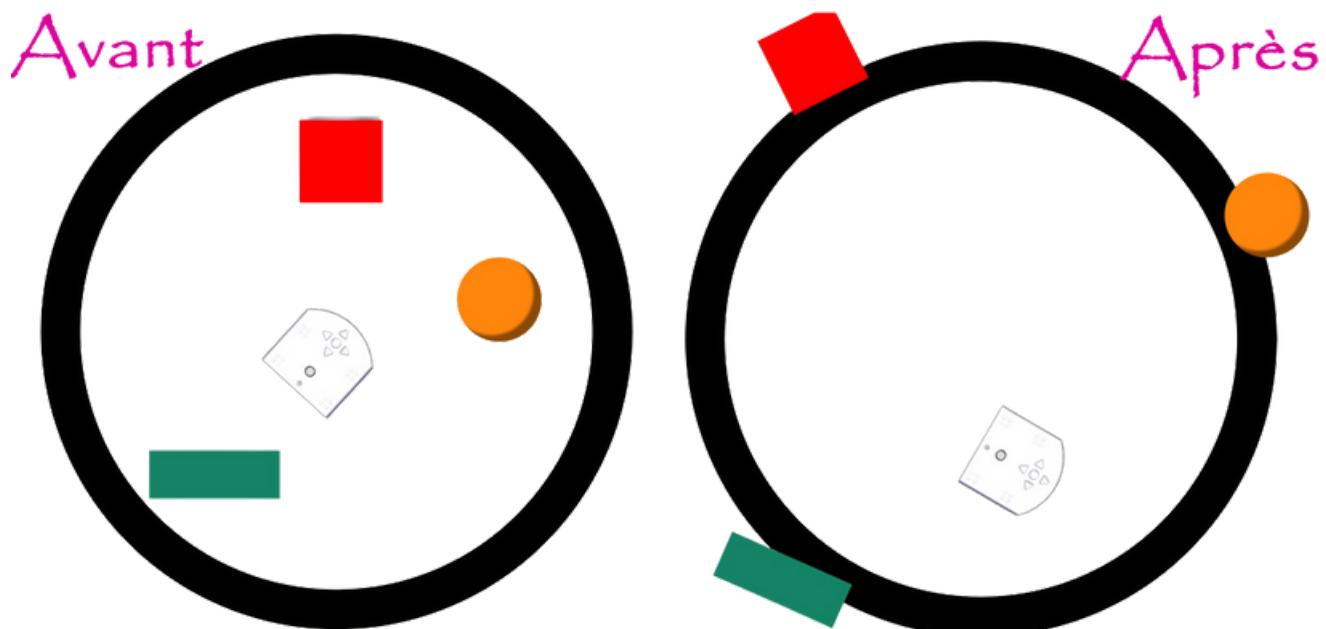
Nombre de robots

## Le défi

Sur une feuille de papier d'environ 70 cm de côté, tracer un cercle noir qui prend presque toute la surface du papier et d'environ 4 cm de large.

Placer Thymio dedans ainsi que 3 ou 4 objets pas trop lourds, de couleurs pas trop foncées et de largeur comprise entre 3 et 10 cm.

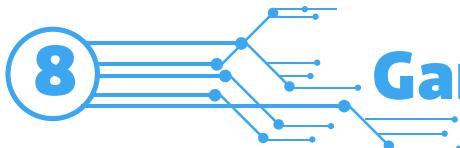
Thymio doit les pousser en dehors du cercle les uns après les autres. Il n'est pas prévu que Thymio s'arrête tout seul une fois la place dégagée.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets





# Garder la piste

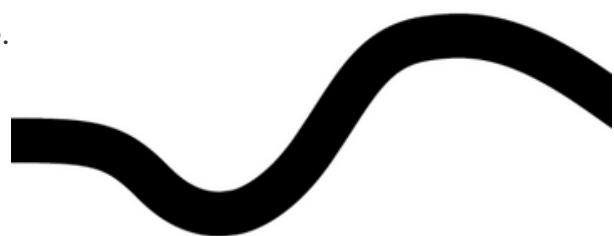


1

Nombre de robots

**Le défi : Thymio doit parcourir une ligne noire de plus en plus vite sans "dérailler".**

Pourquoi les trains ont-ils des rails ? Car cela permet d' aller beaucoup plus vite sans sortir de la trajectoire. Les rails servent de guide très efficace. Pas question de mettre Thymio sur des rails mais on va lui donner une ligne noire à la place. Le déplacement sur une ligne noire a déjà été abordé dans l'exercice de synthèse à la fin du chapitre 5.



La ligne noire ne doit pas être fermée mais doit se terminer par 2 espaces de sol blanc. C'est en détectant ces espaces blancs que Thymio entamera un demi-tour pour re-parcourir la piste.

Si le robot ne se plante jamais, même à la vitesse maximale de 500, refaire une piste aux virages plus serrés ou rétrécir la piste.

En guise de jeu, plusieurs autres versions peuvent être imaginées :

- le robot qui tient le plus longtemps sur la piste sans se planter.
- un nombre maximal d' aller-retours en un temps donné.
- faire une piste fermé sinuuse, mettre 2 robots à l'opposé et regarder celui qui gagne du terrain.

## Propositions sur la programmation.

L'augmentation de vitesse pourra se faire en ajoutant 1 dans le bloc qui fait tourner le robot sur les parties blanches.

Petit piège : nappelez pas votre variable d'augmentation de vitesse step car c'est un mot réservé mais Thonny ne vous affichera pas de message explicite.

Il peut être intéressant de mettre un compteur d'aller-retours mais c'est un peu plus difficile.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets





## Atterrissage



1

Nombre de robots

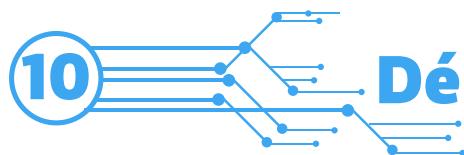
### Le scénario

On tient Thymio horizontalement en l'air (50 cm du sol ou de la table environ) puis on descend verticalement doucement. Au cours de la descente, la couleur rouge apparaît puis s'intensifie, une note (le la3) se fait entendre, suivie d'autres notes toujours plus aiguës.

Quand Thymio entre en contact avec le sol (événement tap), pendant 2 secondes on doit entendre une note grave, la lumière passe au vert, attestant que l'atterrissement s'est bien passé.

Le point délicat de ce script est de calculer l'augmentation de la hauteur du son et de l'intensité du rouge en fonction de la distance au sol qui diminue.

De plus il y a deux phases distinctes, la phase d'approche et celle de l'état posé du robot sur le sol. Il faudra une variable pour gérer les 2 phases.



## Démarrage codé



1

Nombre de robots

Il s'agit d'écrire un script qui n'autorise le démarrage de Thymio que si on tape un mot de passe. Ce 'mot de passe' prendra la forme d'une séquence de boutons à taper dans un ordre précis. On se limitera à 4 frappes parmi les 5 boutons possibles (bouton central et les 4 flèches).

En pratique, on laisse l'utilisateur taper 4 touches (ce peut-être plusieurs fois la même). A la fin de la 4ième frappe :

- si le code est correct, le robot démarre en vert avec une note aiguë assez brève,
- si le code n'est pas bon, Thymio passe en rouge avec le son système 1 (son d'extinction).

Pour faciliter la programmation, on décide que chaque touche aura un numéro.

0 : touche centrale

1 : touche avant

2 : touche droite

3 : touche arrière

4 : touche gauche

Remarque : si vous décidez de stocker les codes dans des listes, les restrictions d'aseba ne permettent pas de comparer 2 listes avec une seule égalité. Il faudra faire des comparaisons membre à membre dans une boucle.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets



# Annexe 1 : Faire des murs et des lignes

---

## Murs en papier ou en carton fin

Plier une feuille A4 3 fois dans le sens de la longueur et la replier suivant les plis. On obtient un tube de section carrée.

L'aplatir et coller les 2 côtés opposés avec un morceau d'adhésif, puis lui redonner son volume.

La dimension est alors de 297 x 52 x 52 mm. L'expérience montre que ces murs sont faciles à ranger, à transporter et à réutiliser.

Pourquoi ne pas prendre du papier déjà utilisé ? Ou faire dessiner des tags par des élèves ?

## Murs en bois

Couper des tasseaux en sections de 30 cm, de hauteur minimum 5cm pour les que rayons des capteurs de Thymio ne passent pas au-dessus.

## Faire des lignes noires

Les lignes doivent avoir une largeur minimale de 4 cm pour un suivi de ligne. La marque Posca propose des marqueurs adaptés. De la peinture peut également convenir, le danger étant que le papier, trop mouillé se déforme.

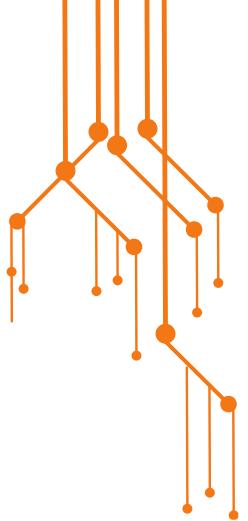
Pour un résultat plus propre, on peut utiliser du ruban adhésif noir de largeur 5 cm.



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les sujets



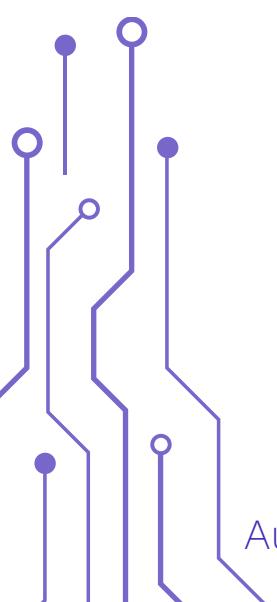


# SESSION 1

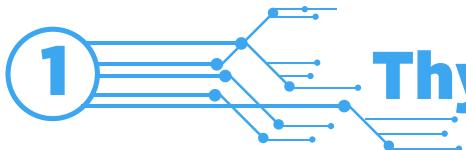
LE CAHIER  
D'ACTIVITÉS

**Les conseils  
et astuces**

P11 - P15



Auteur: JOËL RIVET



# Thymio le craintif

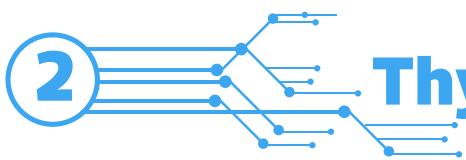
Tout le code peut se placer dans une fonction prox():

on peut traiter successivement les cas :

- un objet à gauche (capteur 0 > 2000) avec un if
- un objet à droite (capteur 4)
- un objet arrière à droite
- un objet arrière à gauche
- un objet sur les 2 capteurs arrière
- un objet sur les 2 capteurs arrière et en plus sur le capteur avant

Il y a d'autres cas possibles mais ce qu'on a vu est suffisant

Il reste le cas où il n'y a rien : on peut mettre les moteurs à 0 dans la fonction directement sans condition



# Thymio l'explorateur

## Aide 1

Ne pas tenir compte des capteurs arrière.

Pour prendre en compte les 4 capteurs de côté, et éviter qu'ils ne se bloquent devant un obstacle, on peut recourir à des variables intermédiaires.

```
p_left = prox_horizontal[0] + prox_horizontal[1]  
p_right = prox_horizontal[3] + prox_horizontal[4]  
p_center = prox_horizontal[2]
```

Puis pour les tests droite et gauche, tester si **p\_left > p\_right** ou le contraire.

## Aide 2

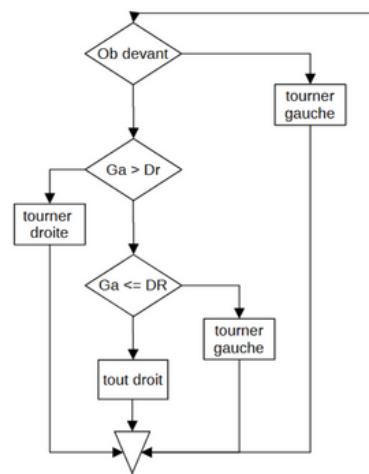
Envisager tous les cas possibles avec les 5 capteurs devant donne  $2^5 = 32$  possibilités. Trop compliqué de les explorer toutes.

Un arbre de décision possible ci-dessous

Légende :

Ob : objet

Ga > Dr : les 2 capteurs de gauche reçoivent des signaux supérieurs à ceux de droite.





3

## Le labyrinthe

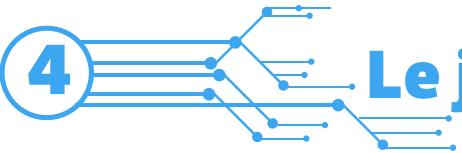
Il faut longer toujours le même mur sans se soucier de l'autre côté.

Si on suit le mur de gauche, seuls les capteurs gauche et centre seront à utiliser.

Il y a finalement peu de chose à faire. Si le robot s'écarte du mur, on l'en rapproche doucement, et inversement.

Si il y a une mur en face, on amorce un virage vers la droite.

Le succès de la mission sera dans les choix des seuils de détection et choix des vitesses des moteurs.



4

## Le jeu du foulard

On peut contrôler les déplacements du robot de 2 manières :

- en utilisant la télécommande qui procure un confortable contrôle à distance (limité à 3 ou 5 joueurs).
- en tapant sur les boutons du robot mais la présence trop proche du joueur peut perturber le jeu et les autres joueurs.

De plus, on peut agir sur les moteurs de 2 manières :

- l'envoi d'une commande met les moteurs à 0 ou à la vitesse choisie (dite vitesse nominale)
- l'envoi d'une commande augmente ou diminue la vitesse d'un moteur, cela procure un meilleur contrôle.

Pour le choix d'une couleur aléatoire, vous pouvez inclure dans le code la fonction suivante et se contenter de l'appeler au début du code :

```
def color_random():
    """ génère 1 couleur aléatoire parmi 12 """
    global leds_top

    n = math_rand() // 5462 + 6
    if n == 0:
        leds_top = [32, 32, 32] # white
    elif n == 1:
        leds_top = [0, 16, 32] # sky blue
    elif n == 2:
        leds_top = [0, 32, 0] # green
    elif n == 3:
        leds_top = [0, 32, 8] # emerald green
    elif n == 4:
        leds_top = [0, 32, 32] # cyan
    elif n == 5:
        leds_top = [8, 32, 0] # fluo green
```



Auteur: JOËL RIVET

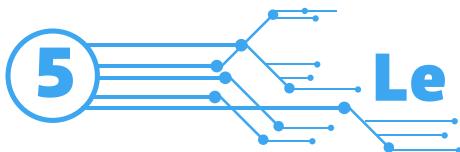
Session 1 - Le cahier d'activités - Les conseils et astuces



```

        leds_top = [8, 32, 0] # fluo green
    elif n == 6:
        leds_top = [32, 0, 0] # red
    elif n == 7:
        leds_top = [32, 0, 8] # fushia
    elif n == 8:
        leds_top = [32, 0, 32] # magenta
    elif n == 9:
        leds_top = [32, 16, 0] # orange
    elif n == 10:
        leds_top = [32, 32, 0] # yellow
    elif n == 11:
        leds_top = [12, 4, 0] # maroon

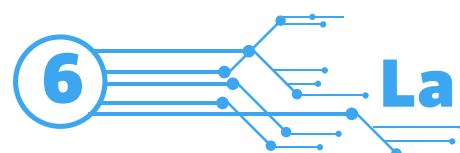
```



## Le relais de Thymio

Utiliser le capteur horizontaux avant et les 2 capteurs arrières pour chaque Thymio. Le premier robot avance, s'arrête derrière le second qui capte l'arrivée du premier par l'un de ses capteurs arrières. Le second démarre, s'arrête derrière le troisième, ainsi de suite. On pourra mettre une ligne noire pour arrêter le dernier.

Toute la difficulté est de bien choisir les seuils de détection avant et arrière.



## La voiture autonome

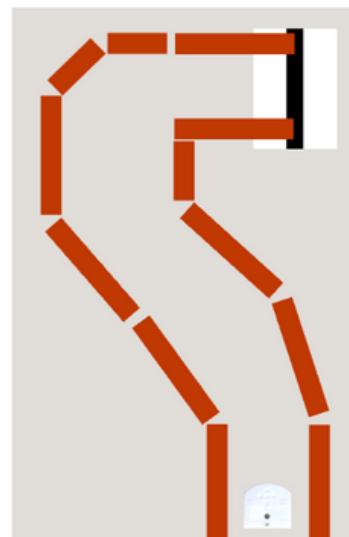
Les murs seront détectés avec les capteurs 0 et 4.

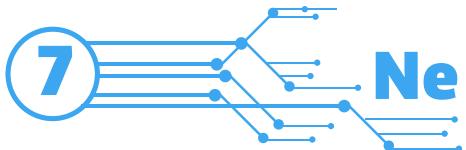
Le demi-tour sur la ligne noire sera géré par un timer pour fixer la durée du pivotement.

Un problème risque d'apparaître au moment du demi-tour si le robot continue de détecter les murs. Il faut donc ajouter une variable qui bloque la détection des murs.

Par exemple, au début du script, mettre une variable `half_turn` à **False**, puis mettre la ligne `if not half_turn :` avant de gérer les prox.

Au moment de la détection de la ligne noire, mettre cette variable à **True**, puis la remettre à **False** à la fin de la rotation (dans la fonction `timer0()`).

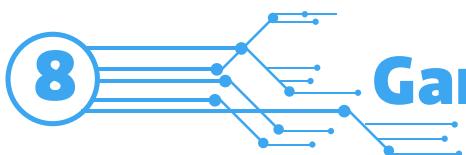




## 7 Ne restez pas là !

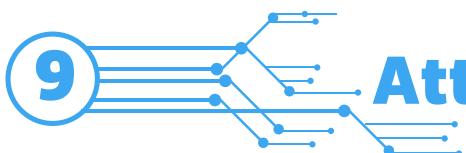
Il faut bien distinguer les cas où :

1. Thymio est à intérieur du cercle. Si il rencontre un obstacle sur côté, il pivote vers ce côté pour être en face de l'objet à pousser. Sinon, il avance tout droit, qu'il y ait quelque chose à pousser ou non.
2. Thymio est sur la ligne noire. Il pivote pendant quelques secondes (avec un timer) sans se soucier des capteurs horizontaux pour revenir à l'intérieur du cercle. Pour qu'il ne se soucie pas des capteurs horizontaux, il faut utiliser un variable qui bloque les capteurs le temps voulu.



## 8 Garder la piste

Pas d'aide spécifique (voir chapitre 5)



## 9 Atterrissage

Pour l'augmentation du son, vous pouvez partir d'une note donnée (ex : la 440) et ajouter le delta fourni par les capteurs sol.

Pour la couleur il faudra faire des essais avec une fonction affine couleur (entre 0 et 32) =  $f(\text{delta})$ .





# Démarrage codé

Pour évaluer le code de l'utilisateur, on peut comparer deux tableaux :

- le premier qui contiendra 4 entiers du code voulu par le programmeur.
- le deuxième qui contiendra la séquence de 4 entiers tapée par l'utilisateur.

Il y aura donc 2 parties dans le code :

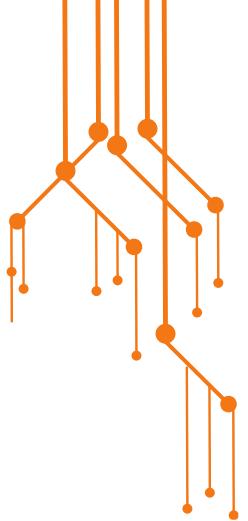
- la partie consacrée à la saisie de l'utilisateur qui comportera un compteur pour compter le nombre de boutons tapés.
- la partie consacrée à la comparaison des 2 tableaux. Attention, les restrictions d'aseba ne permettent pas de comparer 2 listes avec une seule égalité. Il faudra faire des comparaisons membre à membre dans une boucle.

## Les boutons

- Ne pas hésiter à revoir le §3.5 du chapitre 4.
- Il reste une petite difficulté : pour chaque appui sur un bouton, l'événement correspondant est déclenché 2 fois, une fois quand le doigt appuie sur le bouton, une 2ième fois quand le doigt se retire du bouton.
- Par exemple, si l'utilisateur appuie sur le bouton central, la fonction `def button_center()` sera exécutée 2 fois.

Pour ne pas tenir compte du passage où on retire le doigt, on peut utiliser le test `if button_center == 1 : return` à placer au début de la fonction.



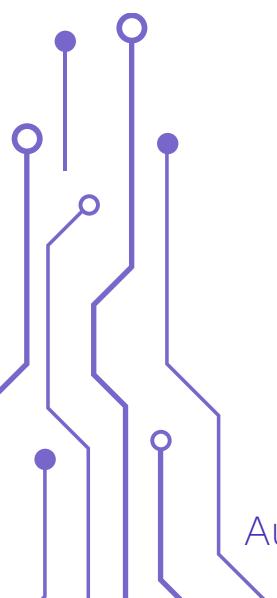


# SESSION 1

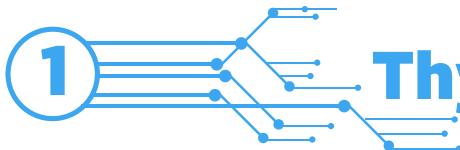
LE CAHIER  
D'ACTIVITÉS

**Les solutions**

P16 - P27



Auteur: JOËL RIVET



# Thymio le craintif

```
speed = 200

@onevent
def prox():
    global motor_left_target, motor_right_target, leds_top

    # rien autour
    motor_left_target = 0
    motor_right_target = 0

    leds_top = BLACK
    if prox_horizontal[0] > 2000: # qqchose avant gauche
        motor_left_target = -speed
        motor_right_target = 0

    if prox_horizontal[4] > 2000: # qqchose avant droit
        motor_left_target = 0
        motor_right_target = -speed

    if prox_horizontal[6] > 2000:
        if prox_horizontal[5] < 1000: # qqchose arrière droit seulement
            motor_left_target = 0
            motor_right_target = speed
        else:                      # qqchose arrière gauche et droit
            motor_left_target = speed
            motor_right_target = speed

    if prox_horizontal[5] > 2000: # qqchose arrière gauche
        motor_left_target = speed
        motor_right_target = 0

    if prox_horizontal[2] > 2000:
        if prox_horizontal[5] > 2000 or prox_horizontal[6] > 2000: # qqchose avant et arrière
            motor_left_target = 0
            motor_right_target = 0
            nf_sound_system(4)
            leds_top = RED
        else:# qqchose avant seulement
            motor_left_target = -speed
            motor_right_target = -speed
```





## Thymio l'explorateur

```
stop = True
speed = 150
motor_left_target = 0
motor_right_target = 0

@onevent
def button_forward():
    global motor_left_target, motor_right_target, stop
    stop = False
    motor_left_target = speed
    motor_right_target = speed

@onevent
def button_center():
    """bouton ajouté pour arrêter le robot"""
    global motor_left_target, motor_right_target, stop
    stop = True
    motor_left_target = 0
    motor_right_target = 0

@onevent
def prox():
    global motor_left_target, motor_right_target, speed, leds_top, stop

    if stop: return

    p_left = prox_horizontal[0] + prox_horizontal[1]
    p_right = prox_horizontal[3] + prox_horizontal[4]
    p_center = prox_horizontal[2]

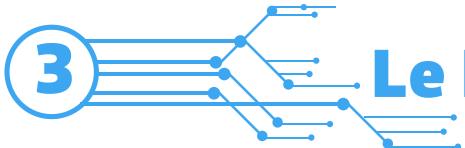
    if p_center > 3500:
        motor_left_target = -speed // 2 # tourne gauche
        motor_right_target = speed

    else:
        if p_left > p_right:
            motor_left_target = speed # tourne droite
            motor_right_target = -speed // 2

        elif p_left < p_right:
            motor_left_target = -speed // 2 # tourne gauche
            motor_right_target = speed
        else:
            motor_left_target = speed
            motor_right_target = speed

    if prox_ground_delta[0] < 400:
        speed = 0
        motor_left_target = speed
        motor_right_target = speed
        leds_top = BLACK
```





## Le Labyrinthe

```
threshold = 2000 # seuil de détection d'un mur
speed = 150
motor_left_target = speed
motor_right_target = speed

@onevent
def prox():
    global motor_left_target, motor_right_target

    if prox_horizontal[0] > threshold:
        motor_left_target = speed
        motor_right_target = speed // 2
    else:
        motor_left_target = speed // 2
        motor_right_target = speed

    if prox_horizontal[2] > threshold:
        motor_left_target = speed
        motor_right_target = 0
```





# Le jeu du foulard

```
speed = 100
gap = 10

leds_top = [16,16, 12]
# couleur aléatoire du robot, pour les distinguer les uns des autres
color_random()
leds_top = [12,4, 0]

@onevent
def button_center():
    global motor_left_target, motor_right_target
    motor_left_target = 0
    motor_right_target= 0

@onevent
def button_forward():
    global motor_left_target, motor_right_target, speed
    speed += gap
    motor_left_target = speed
    motor_right_target= speed

@onevent
def button_backward():
    global motor_left_target, motor_right_target, speed
    speed += gap
    motor_left_target = -speed
    motor_right_target= -speed

@onevent
def button_left():
    global motor_left_target, motor_right_target
    motor_left_target = 0
    motor_right_target = speed

@onevent
def button_right():
    global motor_left_target, motor_right_target
    motor_left_target = speed
    motor_right_target= 0

@onevent
def prox():
    global speed, motor_left_target, motor_right_target

    if prox_horizontal[5] > 2500 or prox_horizontal[6] > 2500: # perdu
        speed = 0
        motor_left_target = 0
        motor_right_target= 0
        nf_sound_freq(1000, 180)
```



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les solutions

```

@onevent
def rc5():
    global motor_left_target, motor_right_target

    if rc5_command==85:          #left remote button
        motor_left_target = 0   #turn on the spot to the left
        motor_right_target = speed
    if rc5_command==87:#center remote button
        motor_left_target = 0   #stop
        motor_right_target = 0
    if rc5_command==86:          #right remote button
        motor_left_target = speed #turn on the spot to the right
        motor_right_target = 0
    if rc5_command==80:          #up remote button
        motor_left_target = speed #move forward
        motor_right_target = speed
    if rc5_command==81:          #down remote button
        motor_left_target = -speed #move backward
        motor_right_target = -speed

def color_random():
    global leds_top

    n = math_rand() // 5462 + 6
    if n == 0:
        leds_top = [32, 32, 32] # white
    elif n == 1:
        leds_top = [0, 16, 32] # sky blue
    elif n == 2:
        leds_top = [0, 32, 0] # green
    elif n == 3:
        leds_top = [0, 32, 8] # emerald green
    elif n == 4:
        leds_top = [0, 32, 32] # cyan
    elif n == 5:
        leds_top = [8, 32, 0] # fluo green
    elif n == 6:
        leds_top = [32, 0, 0] # red
    elif n == 7:
        leds_top = [32, 0, 8] # fushia
    elif n == 8:
        leds_top = [32, 0, 32] # magenta
    elif n == 9:
        leds_top = [32, 16, 0] # orange
    elif n == 10:
        leds_top = [32, 32, 0] # yellow
    elif n == 11:
        leds_top = [12,4, 0] # maroon

```





# Le relais des Thymios

## Premier robot

```
speed = 100
nf_leds_top(32,16,0)

@onevent
def button_forward():
    global motor_left_target, motor_right_target
    motor_left_target = speed
    motor_right_target = speed

@onevent
def prox():
    global motor_left_target, motor_right_target, speed

    if prox_horizontal[2] > 3000:
        speed = 0
        motor_left_target = 0
        motor_right_target = 0
        nf_leds_top(0,0,0)
```

## Deuxième robot

```
speed = 100
nf_leds_top(0,0,0)

@onevent
def prox():
    global motor_left_target, motor_right_target, speed

    if prox_horizontal[5] > 2000 or prox_horizontal[6] > 2000:
        motor_left_target = speed
        motor_right_target = speed
        nf_leds_top(32,0,16)

    if prox_horizontal[2] > 2000:
        motor_left_target = 0
        motor_right_target = 0
        nf_leds_top(0,0,0)

    if prox_ground_delta[0] < 400:
        motor_left_target = 0
        motor_right_target = 0
        nf_leds_top(0,0,0)
```





# La voiture autonome

```
speed = 250
half_turn = False # Thymio ne fait pas son demi-tour et il tient compte des murs

@onevent
def prox():
    global motor_left_target, motor_right_target, timer_period, half_turn

    if not half_turn:
        if prox_horizontal[2] < 4000: # piéton
            if prox_horizontal[0] == 0 and prox_horizontal[4] == 0:
                motor_left_target = speed
                motor_right_target = speed
            else:
                if prox_horizontal[0] > 1500:
                    motor_left_target = speed
                    motor_right_target = speed // 2
                if prox_horizontal[4] > 1500:
                    motor_left_target = speed // 2
                    motor_right_target = speed
        else:
            motor_left_target = 0
            motor_right_target = 0

    if prox_ground_delta[0] < 400 or prox_ground_delta[1] < 400: # ligne noire de
demi-tour
        half_turn = True
        timer_period[0] = 1800
        motor_left_target = speed
        motor_right_target = -speed

@onevent
def timer0():
    global timer_period, half_turn

    half_turn = False # le demi-tour est fini, la détection des murs peu reprendre
    timer_period[0] = 0
```





## Ne restez pas là!

```
turning = False
threshold = 2500
speed = 150
motor_left_target = speed
motor_right_target = speed

@onevent
def prox():
    global motor_left_target, motor_right_target, timer_period, leds_top, turning

    if prox_ground_delta[0] < 400 and prox_ground_delta[1] < 400: # contact ligne
        noire
        turning = True
        timer_period[0] = 3000
        leds_top = RED
        motor_left_target = 100
        motor_right_target = -100

    else:
        if turning == False:
            if prox_horizontal[0] > threshold or prox_horizontal[1] > threshold:
                leds_top = GREEN
                motor_left_target = 0
                motor_right_target = speed
            else:
                if prox_horizontal[3] > threshold or prox_horizontal[4] > threshold:
                    leds_top = BLUE
                    motor_left_target = speed
                    motor_right_target = 0
                else:
                    leds_top = [32,32,0]
                    motor_left_target = speed
                    motor_right_target = speed

        @onevent
        def timer0():
            global timer_period, turning, leds_top
            timer_period[0] = 0
            turning = False # turning n'est remis à false qu'au bout de 3s pour le robot ait
            le temps de tourner tranquillement
            leds_top = BLACK
```





# Garder la piste

```
leds_top = [0,0,0]
speed = 150
motor_left_target = speed
motor_right_target = speed
gap = 1

@onevent
def prox():
    global motor_left_target, motor_right_target, leds_top, speed

    if prox_ground_delta[0] < 400:
        if prox_ground_delta[1] < 400: # noir noir --> tout droit, bleu
            motor_left_target = speed
            motor_right_target = speed
            leds_top = [0, 0, 32]
        else:                      # noir blanc --> rot à gauche, vert
            motor_left_target = 0
            motor_right_target = speed
            leds_top = [0, 32, 0]
    else:
        if prox_ground_delta[1] < 400: # blanc noir --> rot à droite, jaune
            motor_left_target = speed
            motor_right_target = 0
            leds_top = [32, 32, 0]
        else:                      # blanc blanc --> demi-tour, rouge
            motor_left_target = speed
            motor_right_target = -speed
            leds_top = [32, 0, 0]
            speed += gap
            print(speed)
```



**9**

# Atterissage

```
delta = 0
color = 0
height = 0
contact = 0
threshold = 4 # seuil pour éviter que le rouge ne s'allume trop tôt

@onevent
def prox():
    global delta, color, height, contact, prox_ground_delta, leds_top

    if contact == 0:
        delta = (prox_ground_delta[0] + prox_ground_delta[1]) // 2
        if delta > threshold:
            color = delta // 30 + 7
            nf_leds_top(color, 0, 0)

        height = 440 + delta
        nf_sound_freq(height, 10)

@onevent
def tap():
    global contact, leds_top
    contact = 1
    leds_top = [0, 32, 0]
    nf_sound_freq(55, 150)
```



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les solutions



# Démarrage codé

```
# prgm : demarrage code
# auteur : Joël Rivet
# licence : CC-BY-NC
# maj : 15/10/23

# le robot ne démarre que si une suite correct de 4 appuis sur les boutons est respectée.
# par exemple : devant, gauche, droite, centre
# code des touches : 0 centre, 1 avant, 2 droite, 3 arrière, 4 gauche

code = [1, 2, 2, 0] # code secret au choix du programmeur
user = [-1] * 6 # tableau des choix du joueur
index = 0
```

```
index_max = 3
leds_top = BLUE
nf_sound_system(-1) # coupe par défaut des boutons

@onevent
def button_center():
    global user, index

    if button_center == 1: return # détecte le 2ieme passage
    print(index, "ce")
    user[index] = 0
    if index == index_max:
        evaluate()
    else:
        index += 1

@onevent
def button_forward():
    global user, index

    if button_forward == 1: return # détecte le 2ieme passage
    print(index, "fo")

    user[index] = 1
    if index == index_max:
        evaluate()
    else:
        index += 1
```



Auteur: JOËL RIVET

Session 1 - Le cahier d'activités - Les solutions

```

@onevent
def button_right():
    global user, index

    if button_right == 1: return # détecte le 2ieme passage
    print(index,"ri")

    user[index] = 2
    if index == index_max:
        evaluate()
    else:
        index += 1

@onevent
def button_backward():
    global user, index

    if button_backward == 1: return # détecte le 2ieme passage
    print(index,"bw")

    user[index] = 3
    if index == index_max:
        evaluate()
    else:
        index += 1

@onevent
def button_left():
    global user, index

    if button_left == 1: return # détecte le 2ieme passage
    print(index,"lf")

    user[index] = 4
    if index == index_max:
        evaluate()
    else:
        index += 1

def evaluate():
    global leds_top, motor_left_target, motor_right_target, index

    print("eval")
    success = True
    for i in range(4):
        if code[i] != user[i]:
            success = False

    if success: # gagné
        nf_sound_freq(880,30)
        leds_top = GREEN
        motor_left_target = 150
        motor_right_target = 150

    else: # perdu
        leds_top = RED
        nf_sound_system(1)

```

