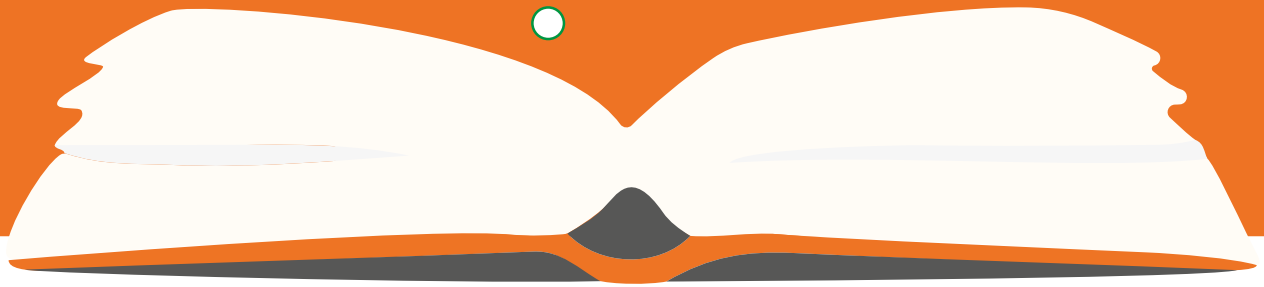
partenariat avec

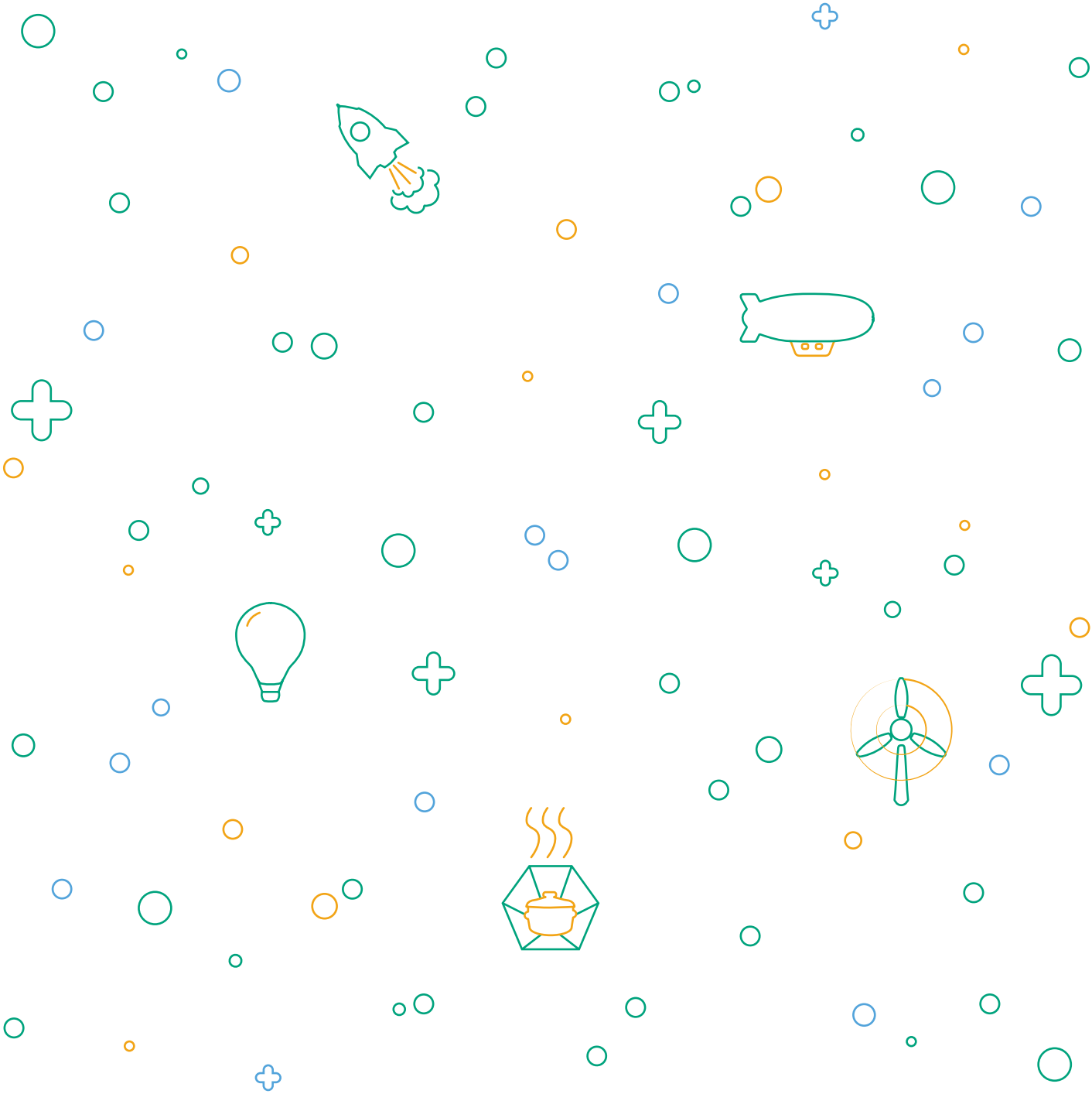
life.augmented

VOYAGER SUR MARS !



GUIDE D'UTILISATION

ROBOT MARTIEN

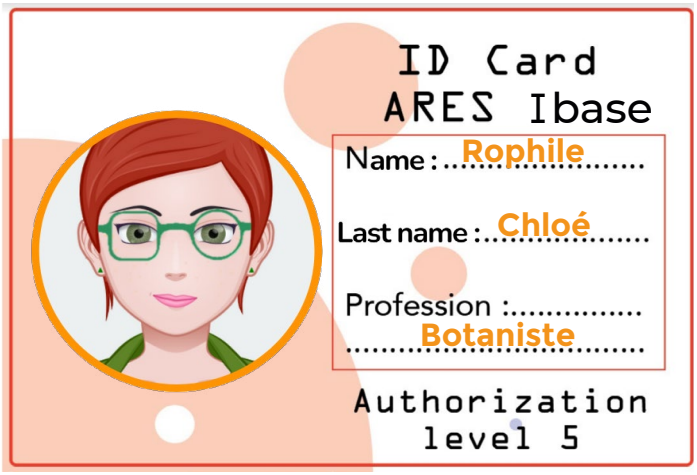


vitta science

LIVRET PÉDAGOGIQUE
ROBOT MARTIEN

Édition rédigée par
@P7i7Plum3 et Damien Vallot
avec le soutien de l'équipe "Survive on Mars"
Mise en page et illustration par
Laura Venezia et Diana Khalipina

• Edition 2021 •
Imprimé en Italie



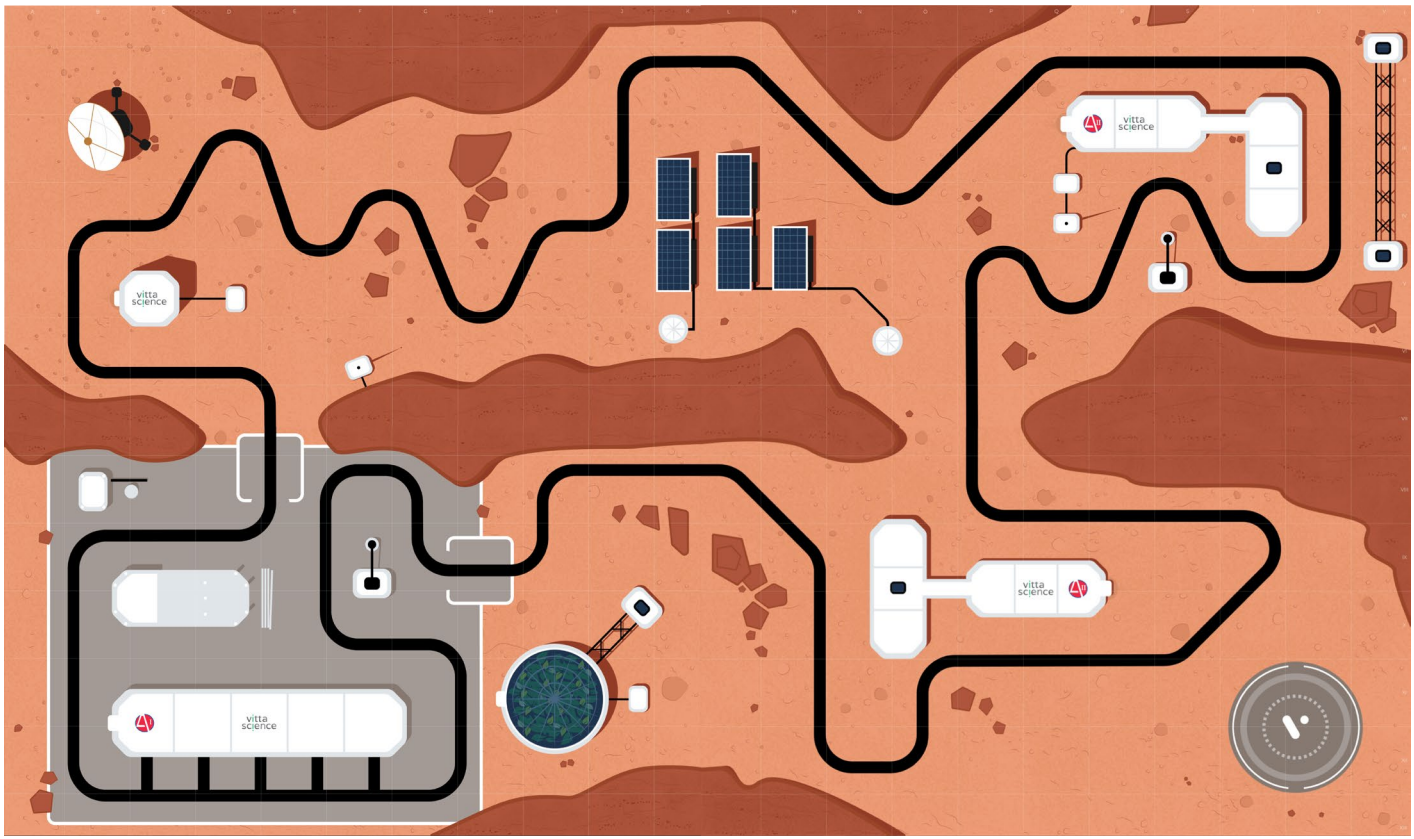
NOS DEUX COLONS MARTIENS PRÊTS À RELEVER TOUS LES DÉFIS !

Nous sommes en 2041. Le succès de la mission sur Mars d'Ingenuity en 2021 est encore dans toutes les mémoires. Ce robot volant de la NASA, déployé aux côtés du rover Perseverance, avait validé le concept de robot attaché à un centre de pilotage sur Mars.

Suite à la réussite de cette mission, de nombreuses autres ont été réalisées avec succès. Depuis l'installation sur Mars de la base souterraine Arès I en 2039, nous cherchons à coloniser la surface.
Pour cela, une nuée de robots a été déployée en 2040 en surface afin de patrouiller, analyser et construire la future base Arès II prévue pour 2042. L'ASM (Agence Spatiale Martienne) basée sur Terre pilote l'ensemble du projet avec l'aide des premiers colons martiens sur place.

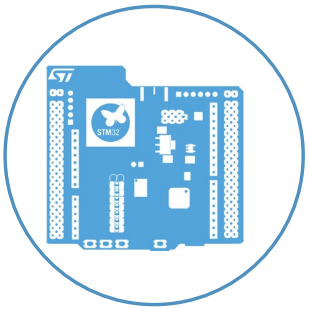
Dans les missions, nous aurons affaire à deux de nos colons les plus émérites de la base Arès I. En effet, suite à un incident survenu sur la base, ceux-ci font face à plusieurs défis !

LE THÉÂTRE DES OPÉRATIONS

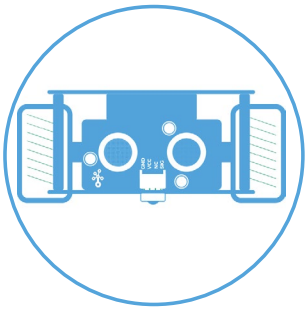


MATÉRIEL NÉCESSAIRE À LA CONSTRUCTION ET À L'UTILISATION DU KIT ROBOT MARTIEN

Contenu du KIT :

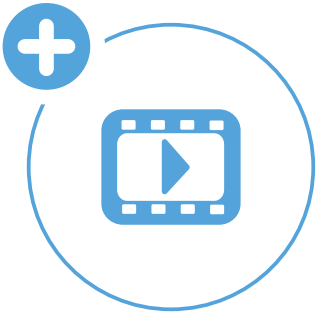


Carte ST NUCLEO-WB55RG



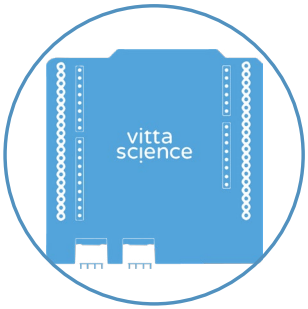
Robot AlphaBot

Piste pour robot

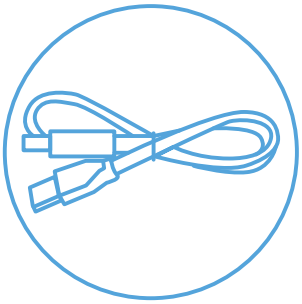


Ressources numériques en ligne

Capteur de distance ST VL53L0X



Nucleo Shield pour Alphabot



Éléments pour le montage

Prévoir un ordinateur

AVERTISSEMENTS CONCERNANT L'UTILISATION DU KIT

ATTENTION !

Présence de petits éléments, ne pas ingérer (risque d'étouffement).

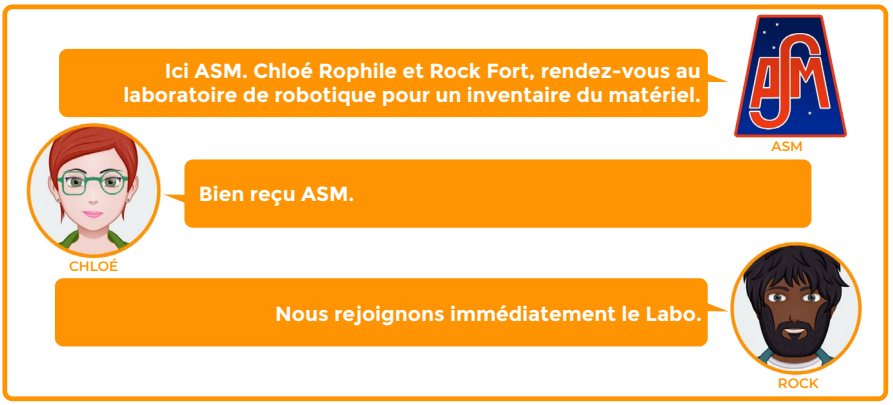
Age minimum

Ne convient pas aux enfants de moins de 7 ans.

CONSIGNES POUR BIEN TRIER

L'infographie suivante détaille les consignes de tri des différents éléments du kit. Pour plus d'informations, rendez-vous sur le site www.consignesdetri.fr

SOMMAIRE



PAGE
38

PAGE
34

Mission 2 : Pilotage des moteurs

- **Mission 2-1 : Piloter les moteurs**
- **Mission 2-2 : Apprendre à tourner**
- **Mission 2-3 : Suivre une trajectoire**

Mission 3 : Tester le détecteur d'obstacles

- **Mission 3-1 : Découverte du capteur à ultrasons**
- **Mission 3-2 : Découverte du capteur Time of Flight (ToF)**
- **Mission 3-3 : Capteurs de distance et moteurs**

PAGE
41

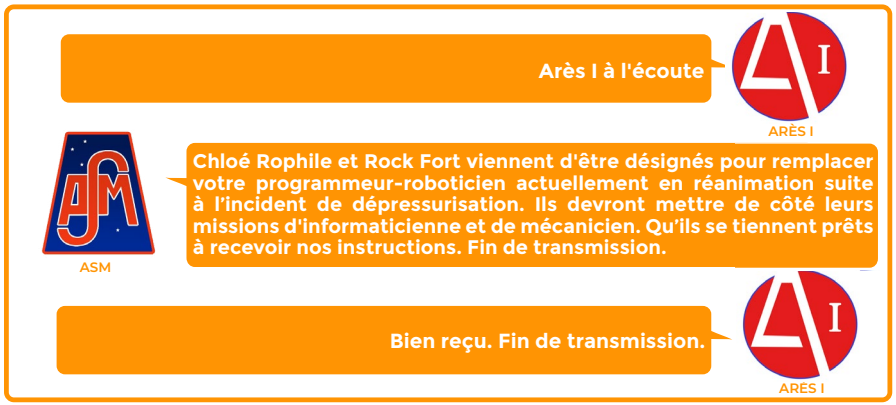
PAGE
44

Mission 4 : Pilotage temps réel

- **Mission 4-1 : La télécommande**
- **Mission 4-2 : Test en conditions réelles “Martiennes”**

Mission 5 : Déplacement en mode case

- **Mission 5-1 : Déplacement en mode cases**
- **Mission 5-2 : Détection d'obstacles avant déplacement**



PAGE 59

PAGE 50

Mission 6 : Suiveur de ligne

- Mission 6-1 : Suivre une ligne
- Mission 6-2 : Suivre une ligne avec 3 capteurs
- Mission 6-3 : Suivre une ligne et éviter les obstacles

Mission 7 : Mise en situation complexe

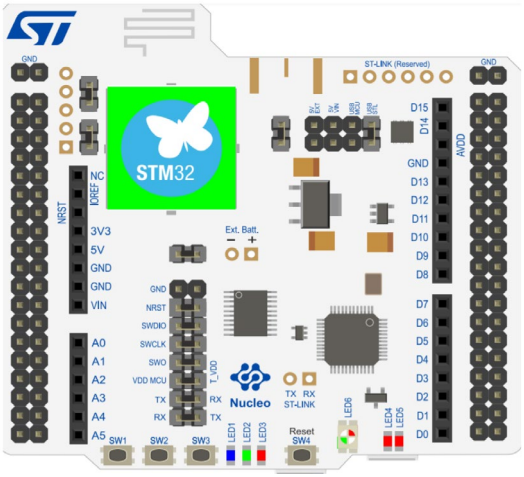
- Mission 7 : Enfin la routine !

Présentation du microcontrôleur et de l'interface Vittascience

Le kit "robot martien" comprend tous les éléments nécessaires pour réaliser le montage d'un robot capable d'explorer Mars de façon autonome.
Il est livré avec une carte ST NUCLEO-WB55RG développée par STMicroelectronics. La carte est équipée d'un microcontrôleur STM32 avec Bluetooth Low Energy (BLE).

• Présentation de la carte NUCLEO-WB55RG ⏱ 15 min

L'illustration suivante montre les entrées et sorties de la carte NUCLEO-WB55RG. Celle-ci peut servir de support à différents montages avec les composants fournis.

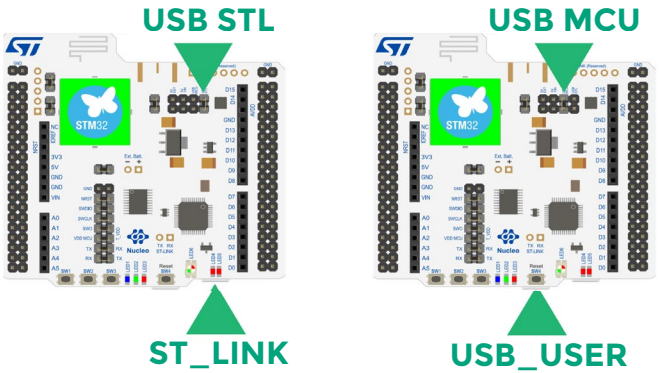


La carte se programme via l'interface Vittascience en MicroPython.
Avant la première utilisation, il est nécessaire de charger un firmware dans la carte NUCLEO-WB55RG afin qu'elle puisse exécuter des programmes en Python (voir encadré).

ENCADRÉ SUR LE CHARGEMENT DU PROGRAMME :

Pour programmer la carte en MicroPython, par code ou par bloc depuis le site Vittascience, il faut charger le firmware adéquat.

(1



Voici les étapes à suivre :

1. Pour flasher la carte, vérifier que le cavalier (pièce métallique entourée de plastique noir) est positionné sur USB STL en haut de la carte au niveau des sources d'alimentation. Si ce n'est pas le cas, déplacer le cavalier sur USB STL. Voir sur l'illustration (1) .
2. Brancher le câble USB sur le port ST _ LINK pour charger le firmware. Les deux LED rouges s'allument.
3. Télécharger le firmware à l'adresse : https://stm32python.gitlab.io/fr/docs/Micropython/Telechargement, puis glisser-déposer celui-ci dans votre carte, qui est apparue comme une clé USB nommée "NOD-WB55". Attention : ne pas décompresser le fichier.
4. Lorsque le téléchargement est terminé, une LED verte s'allume (LED 6 à droite du bouton "Reset").
5. Débrancher le câble USB.
6. Déplacer le cavalier précédemment utilisé (voir le point d'étape n°1) sur USB MCU. Voir sur l'illustration (2) .
7. Rebrancher le câble sur le port USB _ USER (soit l'autre port USB). Voir sur l'illustration (2) .
8. Connecter la carte à l'ordinateur, la LED 5 est allumée en rouge. La carte est donc bien alimentée.
9. Utiliser l'interface Vittascience pour programmer votre carte :
    - utiliser de préférence un navigateur sous Chromium,
    - se rendre sur Vittascience - onglet "Programmer",
    - sélectionner l'interface STM32,
    - cliquer sur “Connecter” et sélectionner la carte,
    - le fichier main.py chargé est exécuté en continu.

Conseil : Un problème, une question ? Nous sommes là pour vous répondre : support@vittascience.com

• Programmation de la carte ⏱ à l'appréciation de l'encadrant

Nous détaillons ici le fonctionnement de l'interface de programmation en ligne Vittascience.com.

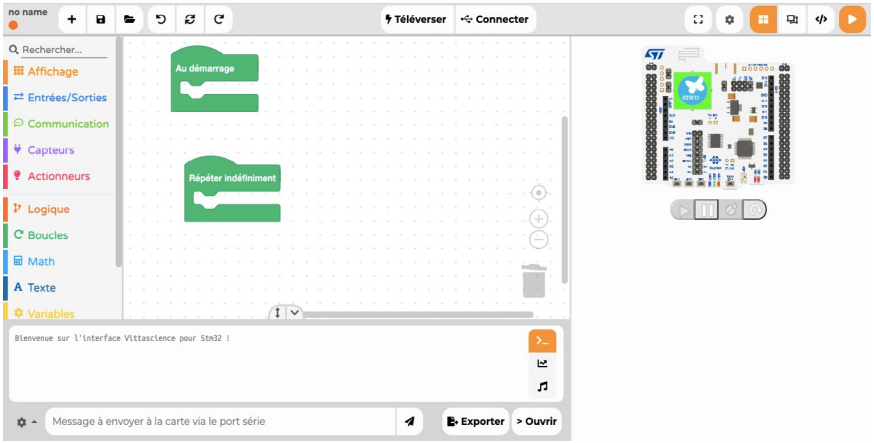
Il est également possible de programmer la carte à l'aide des logiciels Arduino (langage C++). Des tutoriels pour ce logiciel sont disponibles dans la libraire de ressources du site Vittascience.com.

1 • Création d'un compte

Avant tout, nous vous conseillons de vous créer un compte sur notre site. Celui-ci n'est en aucun cas nécessaire pour profiter de votre kit, mais il vous permettra de sauvegarder et partager vos programmes, ressources et retours d'expériences.

Pour cela, rendez-vous sur le site Vittascience.com et cliquez sur l'icône verte en haut à droite pour vous inscrire.

2 • L'interface



L'interface permet de programmer en bloc avec une transcription en parallèle en langage Python.

Attention : La carte ST NUCLEO-WB55RG est nécessaire pour exécuter le programme une fois les capteurs positionnés !
Retrouvez sur le site Vittascience.com des ressources et des programmes pour apprendre à programmer avec la carte.

Sélection du port : Connecter lorsque vous connectez la carte à l'ordinateur, l'interface détecte automatiquement sur quel port elle est connectée. Le menu déroulant permet de sélectionner le bon port si plusieurs cartes sont connectées à l'ordinateur.

Transférer le programme vers la carte : Téléverser le code est exécuté sur la carte dès la fin du transfert.

Annuler ou rétablir : les actions précédentes ou suivantes.

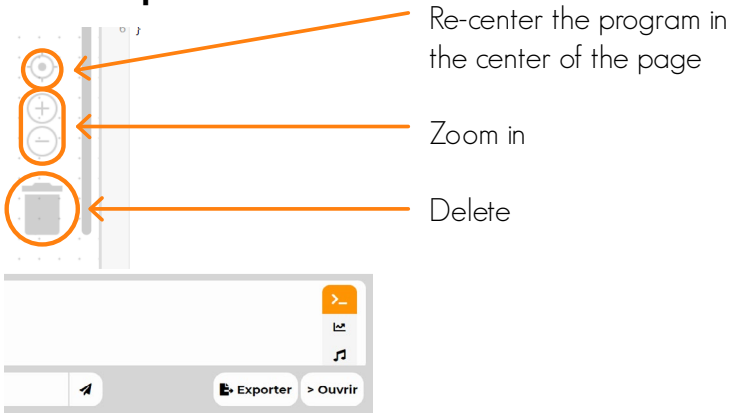
Démarrer un nouveau projet : lancer un nouveau projet vierge, cliquer sur ce bouton.

Sauvegarder le projet : pour enregistrer votre projet afin de le sauvegarder, cliquer sur ce bouton. Pour les personnes disposant d'un compte, il est possible de partager le programme avec la communauté.

Ouvrir un projet existant : Si vous souhaitez ouvrir des programmes déjà réalisés, ou bien faire travailler vos élèves sur une trame que vous avez créée, ils peuvent y accéder en cliquant sur ce bouton.

Coder en Python : si vous souhaitez coder directement en Python.

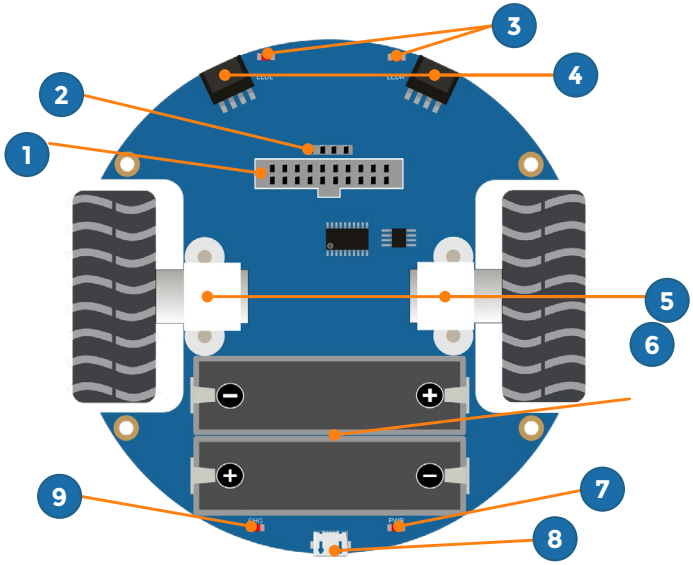
Accès rapide :



Conseil : Cette interface de programmation est conçue pour être très simple d'utilisation, n'hésitez pas à la tester et à la proposer à vos élèves.
Vous disposez sur le site de ressources supplémentaires pour la prendre en main.

Vue d'ensemble de l'AlphaBot2-Base

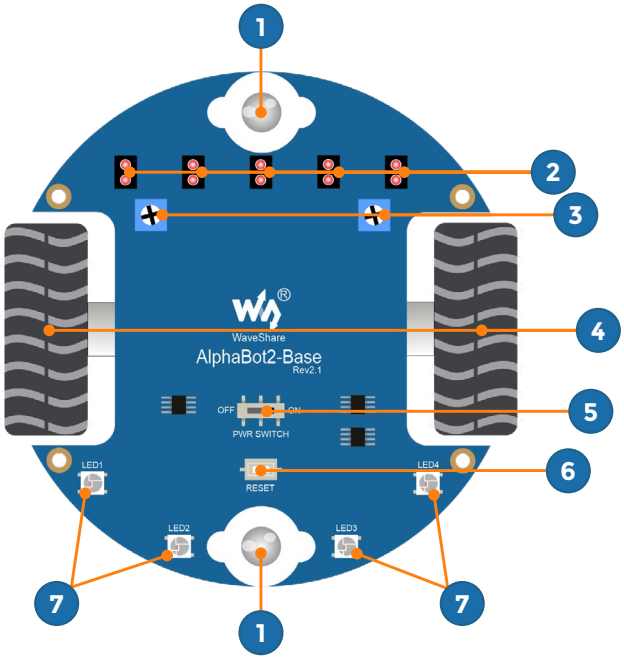
• Figure 1 : Base inférieure, face avant



Source : AlphaBot2-Ar - Waveshare Wiki

1 • Interface de contrôle de l'AlphaBot2 (femelle)
2 • Interface du module ultrasonique
3 • Indicateurs de contournement d'obstacles
4 • Capteur photoélectrique infrarouge réfléchissant, pour l'évitement d'obstacles
5 • Taux de réduction du micro-moteur à engrenages 1:30, 6V/600RPM
6 • Porte-piles pour les piles rechargeables Li-ion 14500. Attention, l'orientation de ce boitier peut être inversée selon les révisions du châssis.
7 • Indicateur d'alimentation
8 • Port USB 5V pour le chargement de la batterie
9 • Indicateur de charge de la batterie

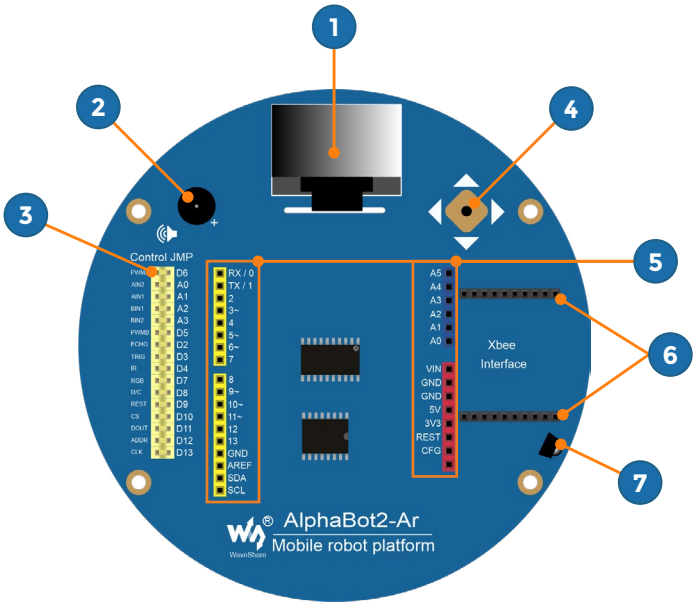
• Figure 2 : Base inférieure, face arrière



Source : AlphaBot2-Ar - Waveshare Wiki

1 • Roue omnidirectionnelle
2 • Capteur photoélectrique infrarouge réfléchissant, pour le suivi de ligne
3 • Potentiomètres pour ajuster la sensibilité du système anticollisions. Vous devrez les ajuster avec un tournevis afin que les LED anticollision (figure 1, point 4) fonctionnent.
4 • Roues en caoutchouc diamètre 42mm, largeur 19mm
5 • Interrupteur d'alimentation
6 • Interrupteur de réinitialisation
7 • LEDs RGB adressables.

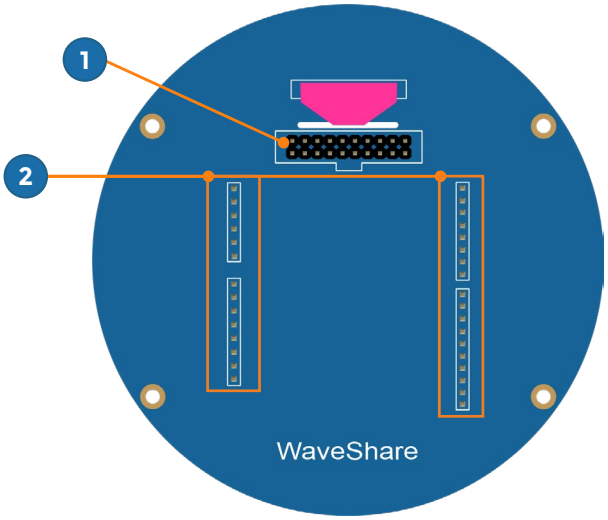
• Figure 3 : Base supérieure, face avant



Source : AlphaBot2-Ar - Waveshare Wiki

1 • Écran OLED 128x64
2 • Buzzer
3 • Connecteurs de périphériques Arduino
4 • Manette de commande
5 • Connecteur d'extension Arduino (femelle)
6 • Connecteur Xbee (femelle), non utilisé dans notre kit.
7 • Récepteur IR (pour la télécommande)

• Figure 4 : Base supérieur, face arrière

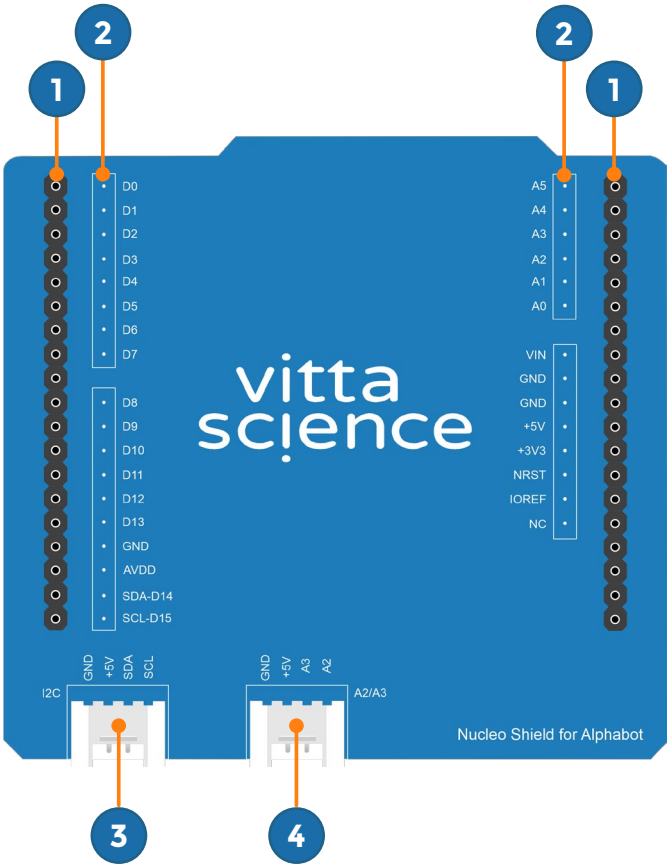


Source : AlphaBot2-Ar - Waveshare Wiki

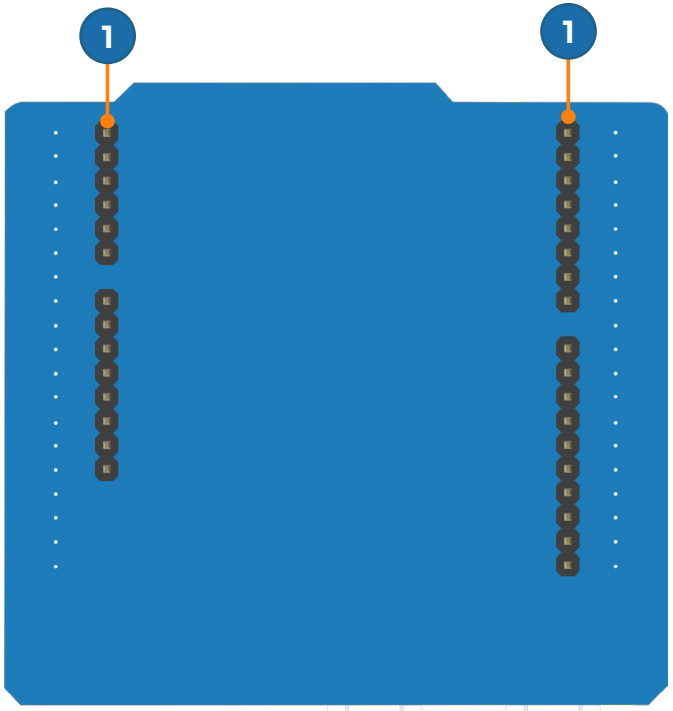
1 • Interface de contrôle AlphaBot2 (mâle)
2 • Connecteur d'extension Arduino (mâle)

Vue d'ensemble de Nucleo Shield pour Alphabot

• Figure 5 : Nucleo Shield pour Alphabot, face avant



• Figure 6 : Nucleo Shield pour Alphabot, face arrière

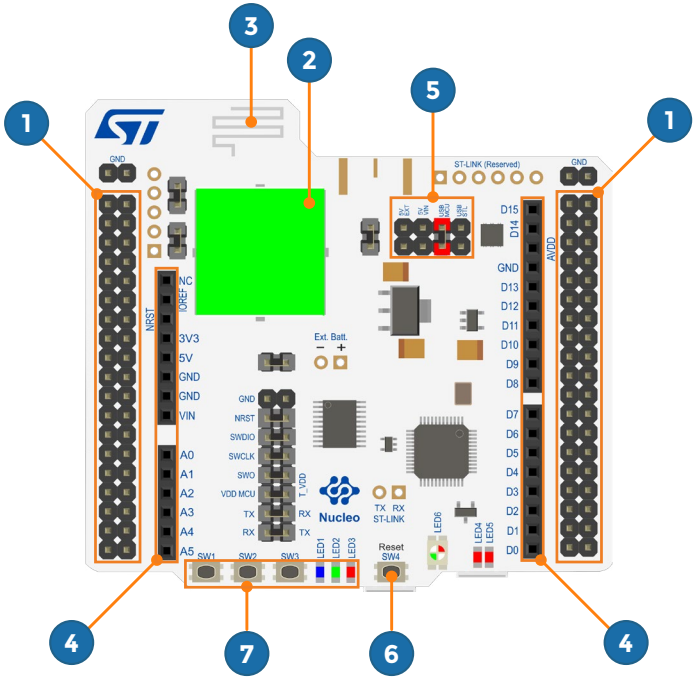


1 • Connecteur femelle pour Nucleo
2 • Libellés des extensions Arduino
3 • Connecteur Grove vers le support I2C
4 • Connecteur Grove vers l'entrée analogique

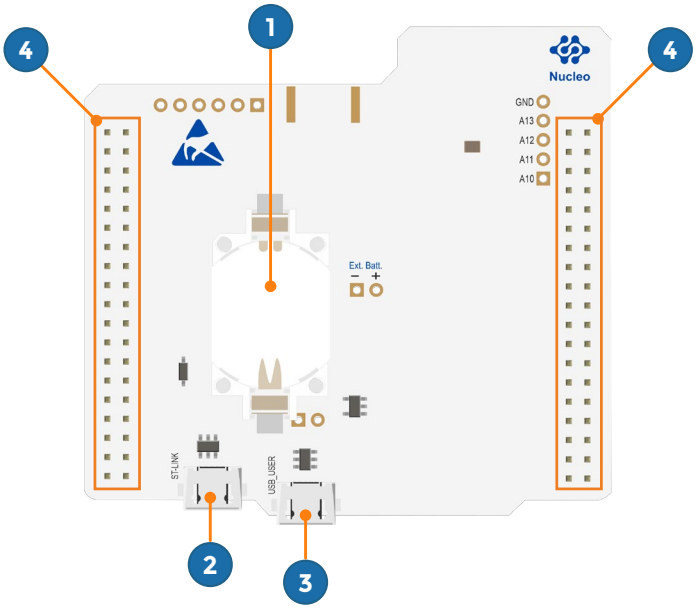
1 • Connecteur d'extension Arduino (mâle)

Vue d'ensemble de la carte Nucleo-WB55

• **Figure 7 : Nucleo-WB55 face avant**



• **Figure 8 : Nucleo-WB55 face arrière**



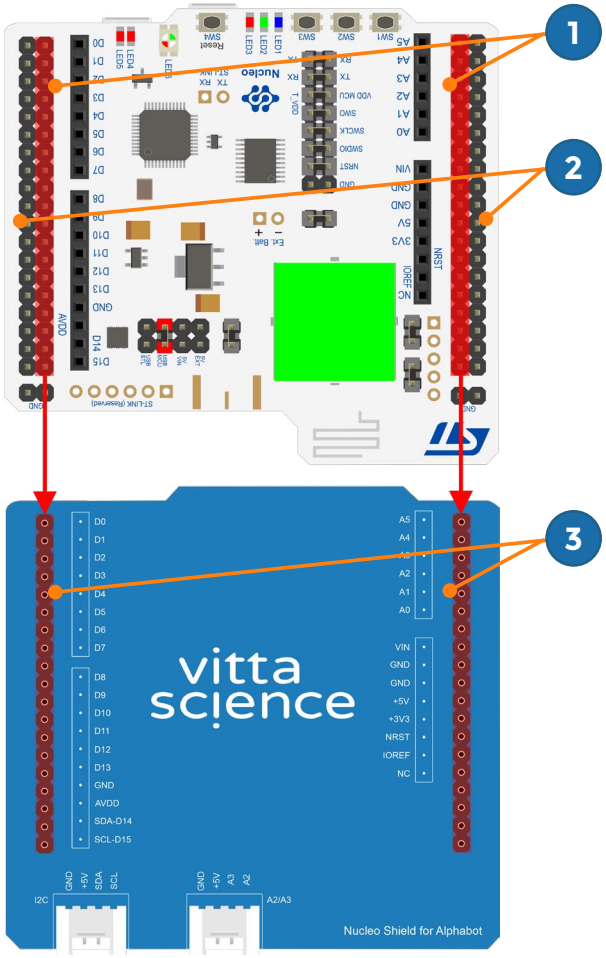
1 • Connecteur pour extension ST-Morpho (mâle)
2 • Puce STM32 WB55 intégrant la radio Bluetooth à faible énergie
3 • Antenne Bluetooth à faible énergie
4 • Connecteur d'extension Arduino (femelle)
5 • Connecteur d'alimentation et de programmation (mâle)
6 • Interrupteur de réinitialisation
7 • Interrupteurs utilisateur (3) et LED utilisateur (3)

1 • Prise pour pile CR2023 (selon les versions)
2 • Connecteur Micro-USB vers ST-LINK (programmation du firmware)
3 • Connecteur Micro-USB vers USB _ USER (lien vers MicroPython REPL)
4 • Connecteur ST-MORPHO (mâle)

Instructions de montage ⏱ à l'appréciation de l'encadrant

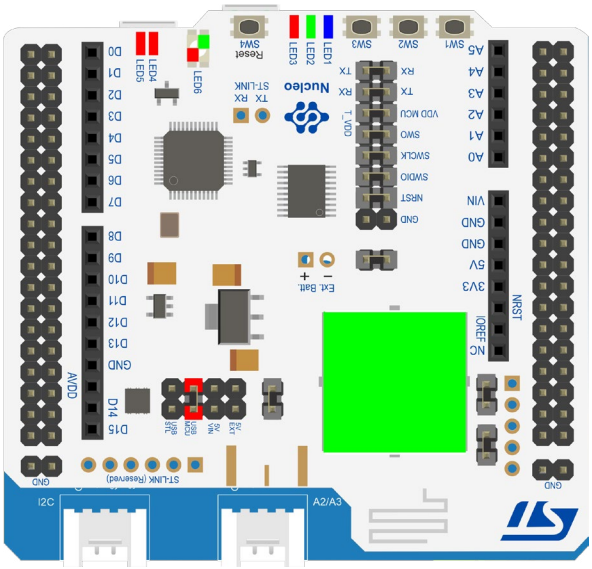
Connexion du Nucleo-WB55 au shield Nucleo pour Alphabot

• **Figure 9 : Nucleo-WB55 et shield Nucleo pour la connexion des connecteurs Alphabot**



1 • Colonnes intérieures du connecteur d'extension ST-Morpho
2 • Colonnes extérieures du connecteur d'expansion ST-Morpho
3 • Nucleo - vers - Arduino connecteur

• Figure 10 : Carte Nucleo-WB55 connectée au shield Nucleo-vers-AlphaBot2 (vue de dessus)



• Tout d'abord, branchez la carte Nucleo-WB55 sur le dessus du Nucleo Shield pour Alphabot.

Pour ce faire, les deux colonnes intérieures du connecteur d'extension ST-Morpho Nucleo-WB55 doivent s'insérer dans le connecteur du pont Nucleo-to-Arduino. Ceci est illustré sur la figure 9.

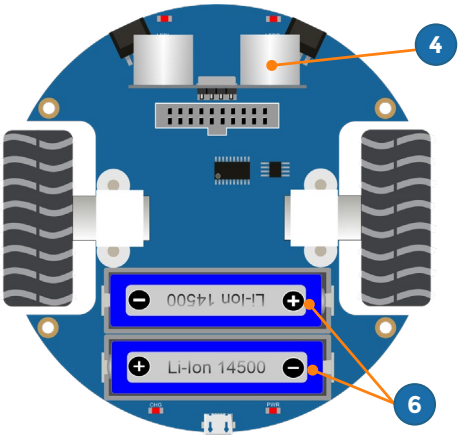
En faisant cela :

- Soyez attentif à la bonne orientation de la Nucleo-WB55 ;
- Assurez-vous que les connecteurs sont correctement alignés et que toutes les broches sont complètement enfoncées ;
- Veuillez noter qu'une fois l'opération terminée, les broches des colonnes externes des connecteurs ST-Morpho doivent flotter parallèlement aux ponts Nucleo – vers – Arduino du shield.

Si vous avez correctement réalisé cette opération, votre système doit ressembler à la figure 10.

Installation de la base Alphabot

• Figure 11 : Installation de la base inférieure d'Alphabot (vue de dessus)

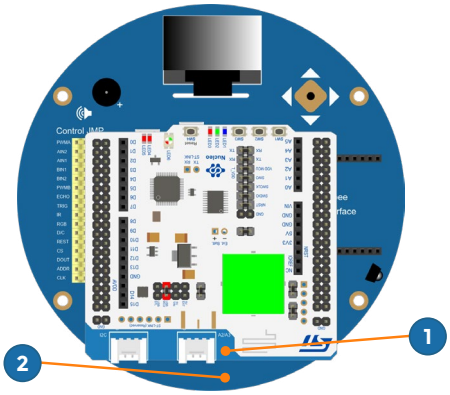


1 • Capteur ultrasonique branché
2 • Piles rechargeables Li-ion 14500 branchées.
Attention, les polarités sont peut-être inversées sur votre robot !

• Préparez la base inférieure Alphabot

Pour ce faire, branchez les deux batteries rechargeables Li-ion 14500 dans le support de batteries et le capteur à ultrasons dans son connecteur. Ceci est illustré sur la figure 11.
En faisant cela :
• Assurez-vous utiliser le bon type de piles, c'est-à-dire des piles rechargeables Li-ion 14500.
Notamment, bien qu'elles s'insèrent parfaitement dans les logements du châssis, les piles AA NE PERMETTENT PAS au robot de fonctionner..

• Figure 12 : Installation de la base supérieure d'Alphabot (vue de dessus)



1 • Shield Nucleo pour Alphabot branché sur le connecteur d'extension Arduino, sur le dessus de la base supérieure d'Alphabot.
2 • Base supérieure d'Alphabot (en dessous)

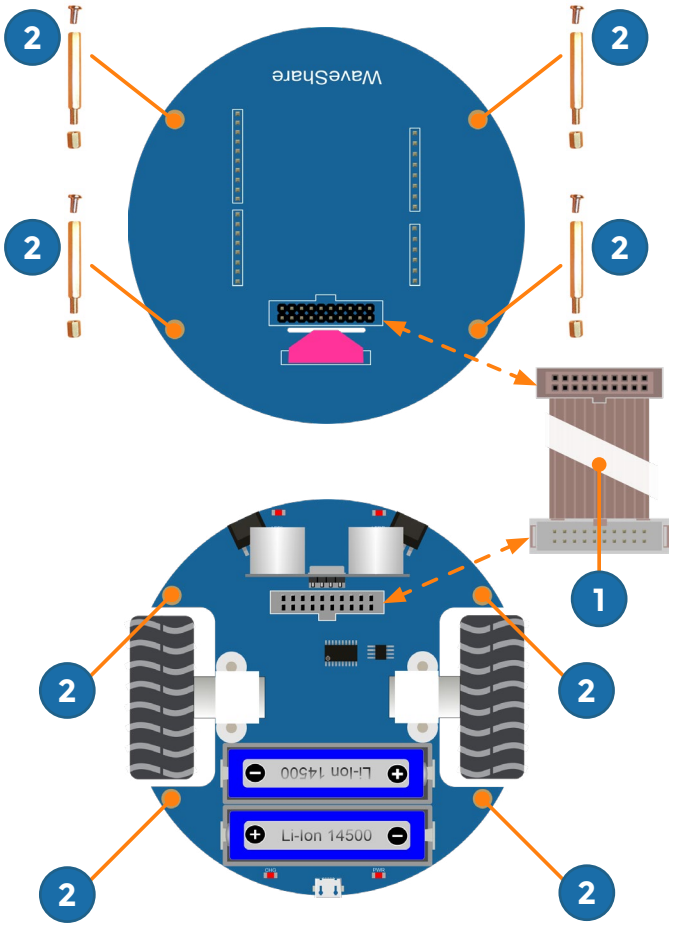
• Préparez la base supérieure d'Alphabot

Pour ce faire, branchez les connecteurs Arduino femelles situés à l'arrière du shield Nucleo pour Alphabot dans le connecteur d'extension Arduino situé à l'avant de la base supérieure d'Alphabot.

En faisant cela :
• Assurez-vous que les connecteurs sont correctement alignés.
Si vous avez effectué cette étape correctement, le résultat, vue du dessus, devrait ressembler à la figure 12.

Raccordement de la base inférieure à la base supérieure d'Alphabot

• Figure 13 : Relier la base inférieure à la base supérieure



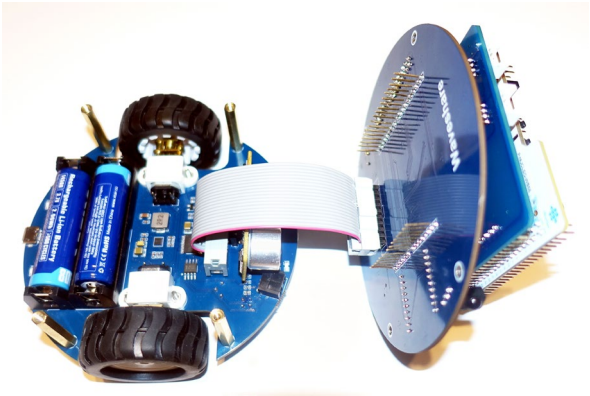
1 • Carte de connecteurs flexible.
2 • Trous et vis d'écartement
3 • Trous pour les vis d'écartement

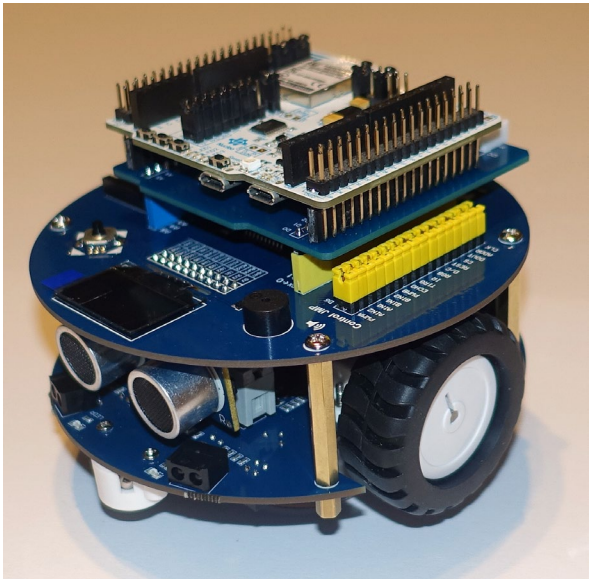
• Connecter les couches Alphabot

Pour terminer,
1 • Branchez l'extrémité femelle de la carte de connexion flexible sur l'interface de commande de l'AlphaBot2, à l'arrière de la base supérieure.
2 • Branchez l'extrémité mâle de la carte de connexion flexible à l'interface de commande de l'AlphaBot2 sur le côté avant de la base inférieure.
3 • Terminez l'assemblage de la base supérieure et de la base inférieure en plaçant les 4 vis d'écartement dans les trous correspondants.

En faisant cela :
• Assurez-vous tirer fermement les extrémités de la carte flexible à l'intérieur des connecteurs des bases supérieure et inférieure.
La figure 13 illustre cette étape.

Photos du robot martien assemblées





• Photo 1 : Raccordement de la base inférieure d'Alphabot à la base supérieure.

• Photo 2 : Robot martien assemblé.

• Photos du robot assemblé

Les photos 1 et 2 vous donneront une confirmation visuelle que vous n'avez pas fait d'erreur en suivant les instructions de montage précédentes.

Maintenant, il est temps de partager les derniers points techniques pour utiliser correctement le robot !

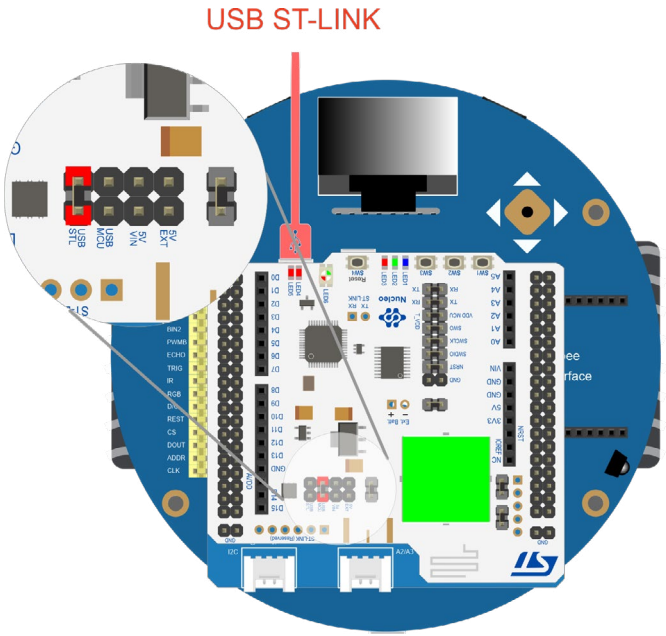
Programmation du robot martien

⏱ à l'appréciation de l'encadrant

Configuration de la mise à jour du firmware de MicroPython

1 • Placer l'interrupteur de mise sous tension situé sous le robot sur « OFF ».
2 • Positionnez le cavalier indiqué en rouge sur la figure 14 en position « USB STL ».
3 • Reliez la Nucleo-WB55 à votre ordinateur par son connecteur USB _ STL.
4 • Depuis votre ordinateur, faites un glisser-déplacer du nouveau firmware dans le disque USB virtuel associé à la Nucleo-WB55.
5 • Patientez jusqu'à ce que l'opération de copie soit terminée.

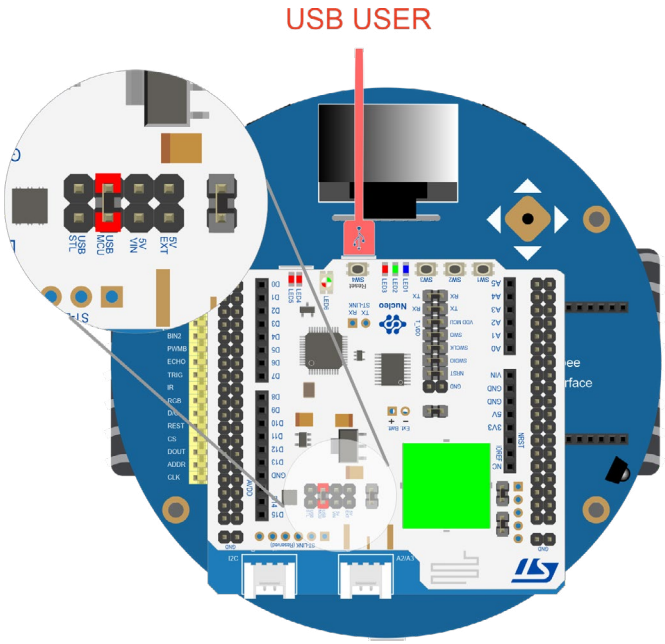
Afin que le robot fonctionne correctement, il est nécessaire qu'il dispose du firmware dans une révision 1.17 ou plus récente. Nous vous recommandons de télécharger la dernière version validée du firmware exclusivement depuis le site de Vittascience.



Configuration pour la programmation en MicroPython

1 • Mettez l'interrupteur sous le robot sur « OFF »
2 • Mettre le cavalier (en rouge) sur la figure ci dessus sur la position « USB MCU ».
3 • Connectez votre carte à votre ordinateur en utilisant le connecteur « USB USER » de la carte Nucleo-WB55.
4 • Connectez le câble USB à votre ordinateur et démarrez l'interface de programmation de Vittascience : vittascience.com/stm32

Vous pouvez maintenant vous connecter à l'interface Vittascience et télécharger vos programmes dans le Nucleo-WB55 qui pilote le robot.

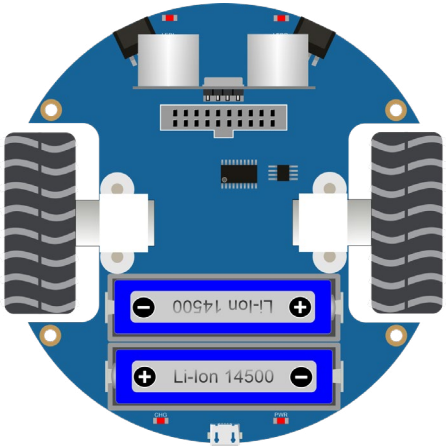


Faire fonctionner le robot martien

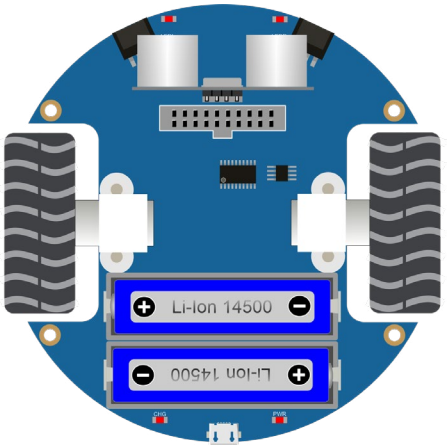
⏱ à l'appréciation de l'encadrant

Chargement des batteries du robot

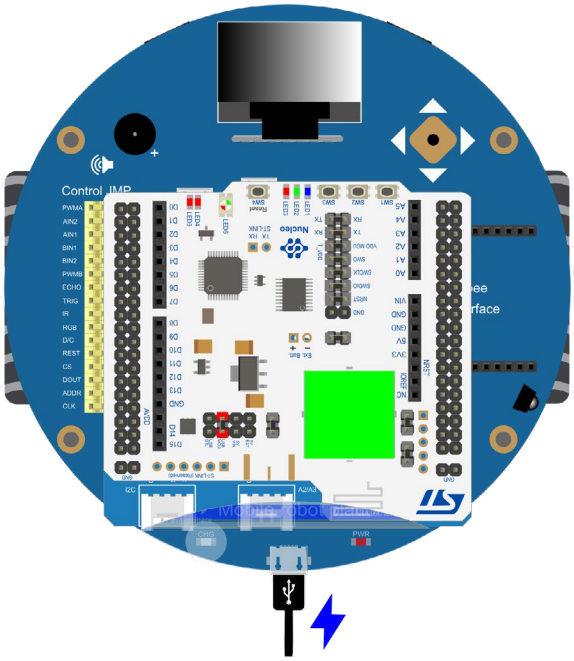
1 • Si l'orientation des accumulateurs pour votre robot correspond à la figure 16-a, placez l'interrupteur de mise sous tension situé sous le robot sur « OFF ».
2 • Si l'orientation des accumulateurs pour votre robot correspond à la figure 16-b, placez l'interrupteur de mise sous tension situé sous le robot sur « ON ».
3 • Connectez un seul câble USB sur le robot, sur le port USB 5V de la base de l'AlphaBot2.
4 • La LED CHG doit rester allumée jusqu'à ce que les piles soient complètement chargées. Si la LED clignote rapidement, veuillez vérifier que le câble USB et les piles sont bien branchés.
5 • Lorsque les piles sont pleines, le voyant CHG est éteint. Votre robot est maintenant prêt à démarrer, il suffit de mettre l'interrupteur d'alimentation situé sous le robot sur « ON ».



• Figure 16-a



• Figure 16-b



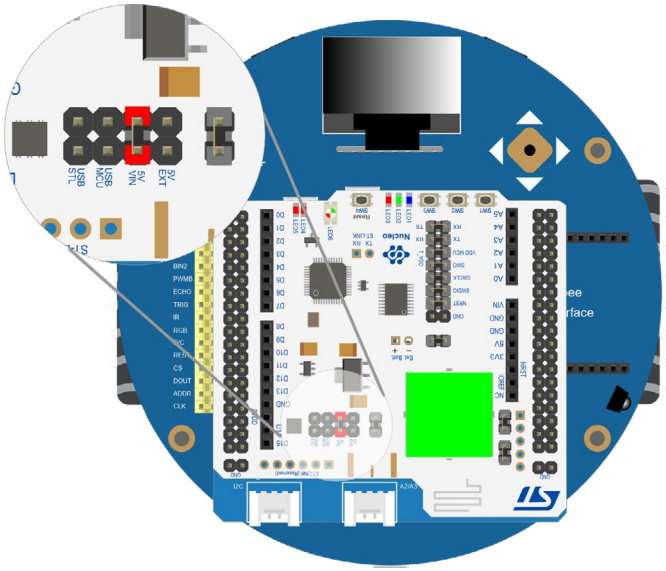
• Figure 17

Démarrage du robot

1 • Débranchez tout câble USB connecté à votre robot.
2 • Mettre le cavalier (en rouge) sur la figure ci dessus sur la position «5V VIN». Maintenant, la carte Nucleo-WB55 va tirer son énergie des deux batteries de la base Alphabot.
3 • Placez l'interrupteur d'alimentation sous le robot sur « ON », la LED PWR sur sa base doit s'allumer.

Si les batteries de l'Alphabot sont suffisamment chargées, la LED PWR de la base du robot s'allumera et le robot commencera à exécuter le dernier programme que vous avez téléchargé dans le Nucleo-WB55 en utilisant l'interface de programmation de Vittascience.

Si le comportement de votre robot n'est pas celui que vous attendiez, vérifiez votre programme et assurez vous que les accumulateurs sont assez chargés..



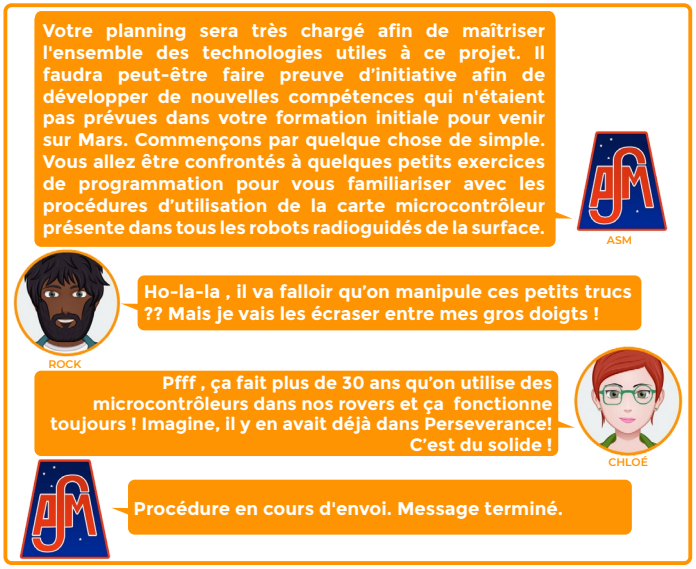
Mission • 1 ⏱ à l'appréciation de l'encadrant

Découverte et utilisation du matériel

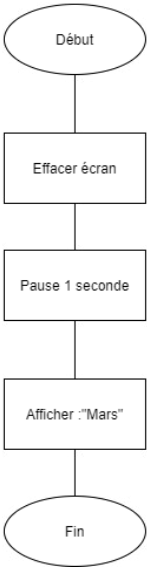
Découverte de la carte contrôleur du futur mini rover (mission très simple pour maîtriser le transfert de programme et l'interface de la plateforme Vittascience).

• Mission 1-1 : Afficher un texte sur l'écran OLED

Pour débuter cette première mission nous allons afficher le texte "Mars" sur l'écran OLED du robot.



Voici dans l'ordre l'algorigramme puis le code de programmation par bloc puis enfin le code Python.




```
1  import machine
2  from stm32_ssd1306 import SSD1306, SSD1306_I2C
3  import utime
4
5  oled = SSD1306_I2C(128, 64, machine.I2C(1))
6
7  while True:
8    oled.text('Mars ', 0, 0)
9    oled.show()
10   utime.sleep(1)
11   oled.fill(0)
12   oled.show()
```

Dans l'interface de programmation de Vittascience, on pourra au choix programmer soit en Python soit en code bloc. Dans un premier temps et en fonction du niveau, il est conseillé d'utiliser le code bloc qui est plus simple pour débuter.

Les bons blocs sont dans le menu de gauche, pour commencer seuls les deux suivants seront utiles :

Robots | Entrées/Sorties

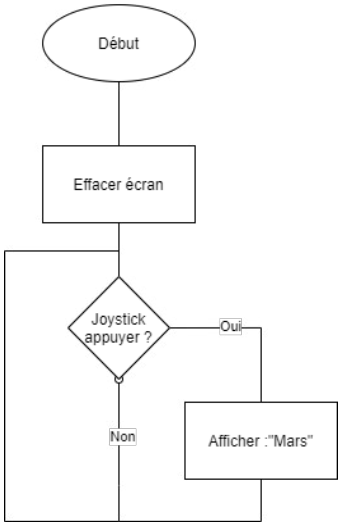
• Mission 1-2 : Utilisation du joystick

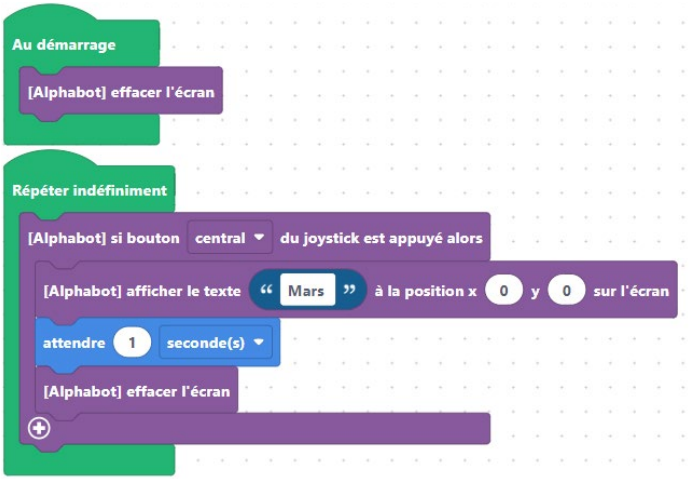
Nous venons d'apprendre comment afficher un texte avec la carte. Étudions à présent comment interagir avec l'écran à l'aide du joystick.

Nous allons garder "Mars" comme message de base, mais faire en sorte qu'il s'affiche lorsque nous appuyons sur le joystick du robot. Nous aurons besoin pour ce faire de nouveaux blocs qui se trouvent dans les menus suivants :

Robots | Entrées/Sorties

Boucles



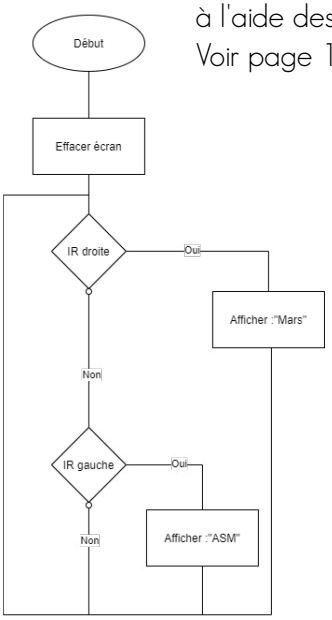


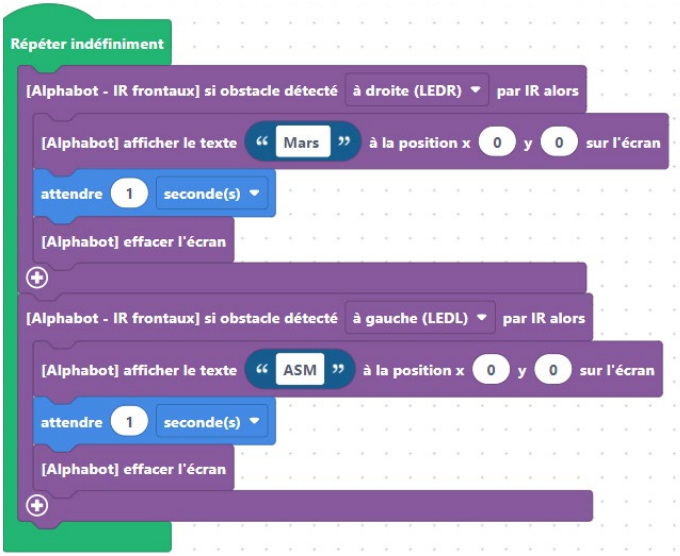

```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  joystickButton = alphabot.getJoystickValue()
  if joystickButton == "center":
    oled.text('Mars ', 0, 0)
    oled.show()
    utime.sleep(1)
    oled.fill(0)
    oled.show()
```

• Mission 1-3 : Utilisation des capteurs infrarouges frontaux

À la place du joystick, qui n'est qu'un capteur de pression basique (appuyé ou pas) nous allons utiliser les capteurs infrarouges frontaux du robot.
Pour cette mission, nous allons compléter le code précédent avec la fonction suivante : « Si je passe la main devant le capteur de droite, alors l'écran OLED affiche "Mars" pendant 1 seconde. Si je passe la main devant le capteur de gauche, alors il affiche "ASM". » (Là encore l'information reçue sera basique « je capte ou pas »).
Voici, dans l'ordre, l'algorigramme, le code de programmation par blocs et enfin le code Python.
Celui-ci contient un bloc "effacer écran" au démarrage qui permet de s'assurer que l'écran est bien initialement vide.

Attention : Si les capteurs infrarouges ne semblent pas fonctionner, ajuster leur sensibilité à l'aide des potentiomètres sous le robot. Voir page 18, figure 2, point 3.



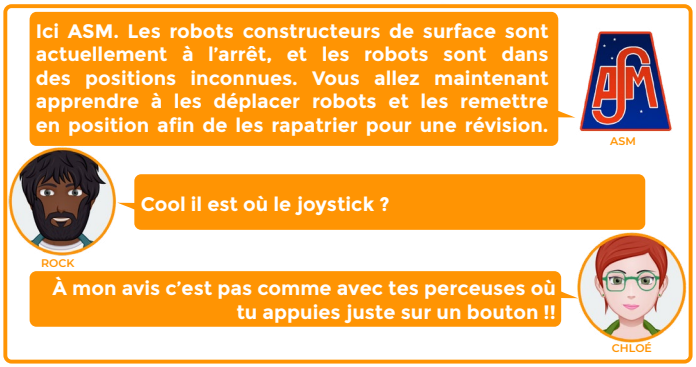



```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  detection = alphabot.readInfrared()
  if detection == alphabot.RIGHT_OBSTACLE:
    oled.text('Mars ', 0, 0)
    oled.show()
    utime.sleep(1)
    oled.fill(0)
    oled.show()
  detection = alphabot.readInfrared()
  if detection == alphabot.LEFT_OBSTACLE:
    oled.text('ASM', 0, 0)
    oled.show()
    utime.sleep(1)
    oled.fill(0)
    oled.show()
```

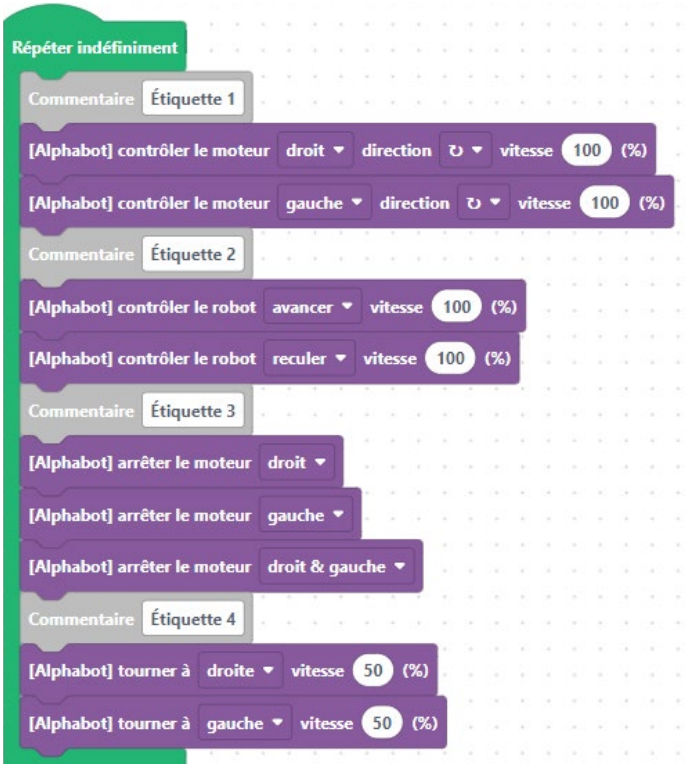
Mission • 2 ⏱ à l'appréciation de l'encadrant
Pilotage des moteurs

• Mission 2-1 : Piloter les moteurs

Il est temps de mettre en mouvement notre petit robot. Commençons par les mouvements simples "avancer" et "tourner".



Le roues du robot peuvent être contrôlées de plusieurs façons, à l'aide de trois blocs de code :

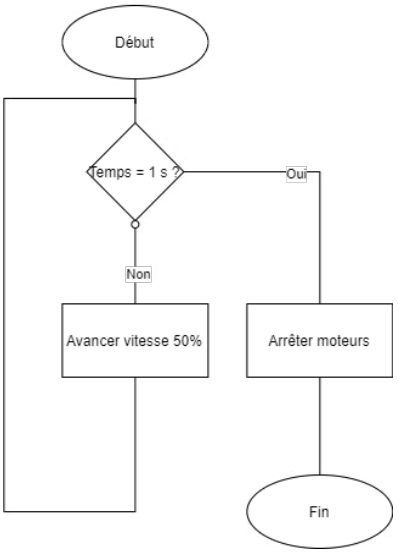


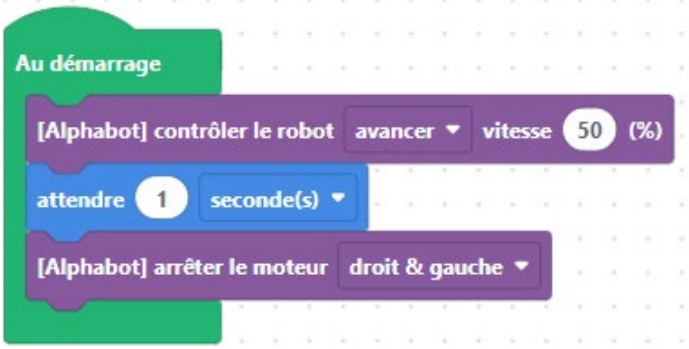
Bloc 1 : Permet de contrôler chaque moteur droit ou gauche de manière indépendante, de contrôler le sens de rotation et enfin de contrôler la vitesse de rotation.
Bloc 2 : Permet de contrôler simultanément le sens et la vitesse de rotation des deux moteurs.
Bloc 3 : Permet d'arrêter l'un des deux moteurs seulement, ou bien les deux simultanément.
Bloc 4 : Permet d'activer le moteur pour tourner à droite ou à gauche et de contrôler sa puissance.

Pour notre premier programme nous allons nous contenter de faire avancer le robot en marche avant pendant 1 seconde.

Pour ce faire, nous utiliserons des blocs qui se trouvent dans :

Voici, dans l'ordre, l'algorigramme, le code de programmation par blocs et enfin le code Python :






```python
import machine
from stm32_alphabot_v2 import AlphaBot_v2
import utime

alphabot = AlphaBot_v2()

alphabot.moveForward(50)
utime.sleep(1)
alphabot.stop()

while True:
  pass
```

• Mission 2-2 : apprendre à tourner

Un robot qui avance tout droit c'est bien mais c'est dangereux, nous devons lui apprendre à tourner ! Pour cela il est nécessaire contrôler indépendamment chaque moteur.

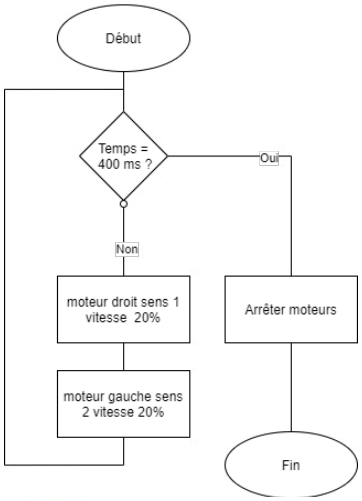
Deux choix s'offrent à nous pour tourner : la méthode "char d'assaut" ou la méthode "voiture". Le char d'assaut fait tourner ses chenilles droite et gauche en sens opposés, ce qui lui permet d'effectuer une rotation sur place. La voiture avec le système de différentiel a généralement une roue qui tourne plus vite que l'autre pour suivre une trajectoire courbe.

Chaque technique présente ses avantages et ses inconvénients. La technique "char d'assaut" permet des virages plus serrés au détriment de la vitesse globale de déplacement. Pour la méthode "voiture", c'est l'inverse.

Dans notre cas nous, allons nous déplacer dans un milieu hostile (Mars) donc nous nous privilégierons la manœuvrabilité du robot avec la méthode "char d'assaut" pour réaliser des virages serrés à 90°.

Pour ce faire, nous utiliserons des blocs qui se trouvent dans :

Voici, dans l'ordre, l'algorigramme, le code de programmation par blocs et enfin le code Python :

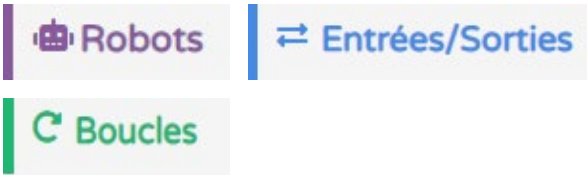



```python
import machine
from stm32_alphabot_v2 import AlphaBot_v2
import utime

alphabot = AlphaBot_v2()

alphabot.setMotors(right=20)
alphabot.setMotors(left=-20)
utime.sleep_ms(400)
alphabot.stop()

while True:
  pass
```

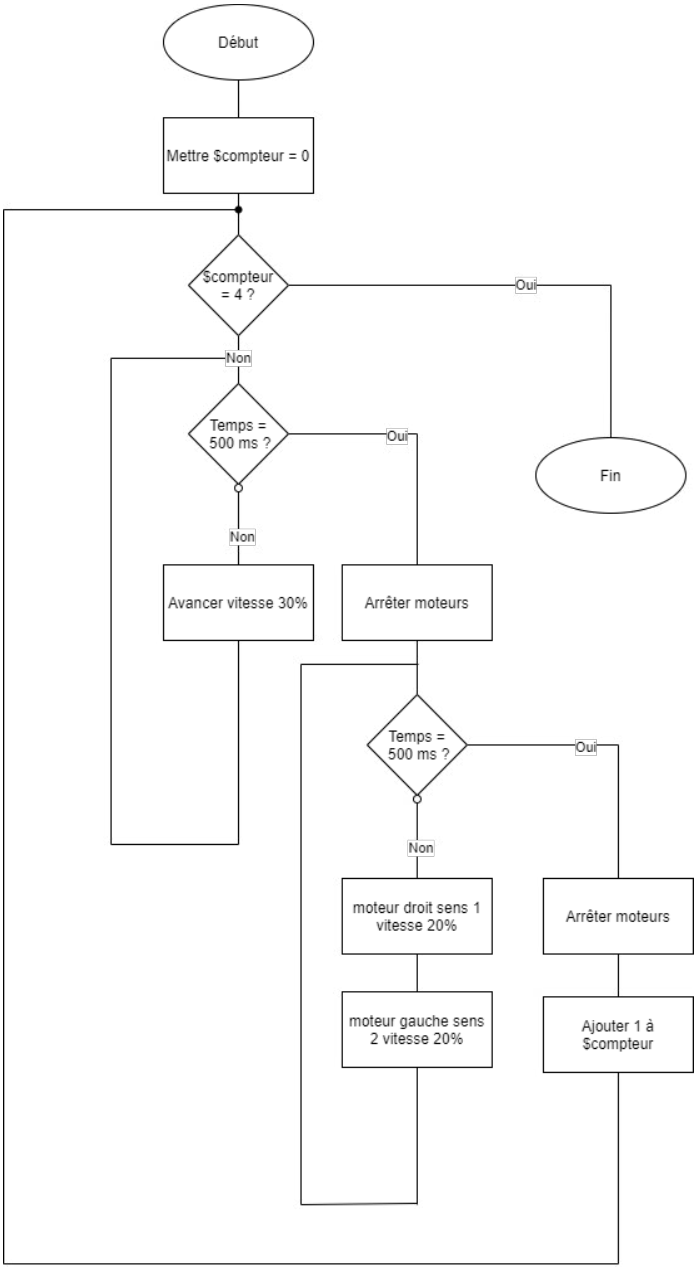
Ce programme ne produira probablement pas un virage à 90° parfait. Pour ce faire vous devrez ajuster les temporisations soit par essais et erreurs, soit à l'aide d'un tableau de proportionnalité. Ne pas hésiter à réduire la vitesse afin de mieux contrôler la direction.
Nous contrôlons maintenant parfaitement notre robot et pouvons le déplacer où nous voulons.

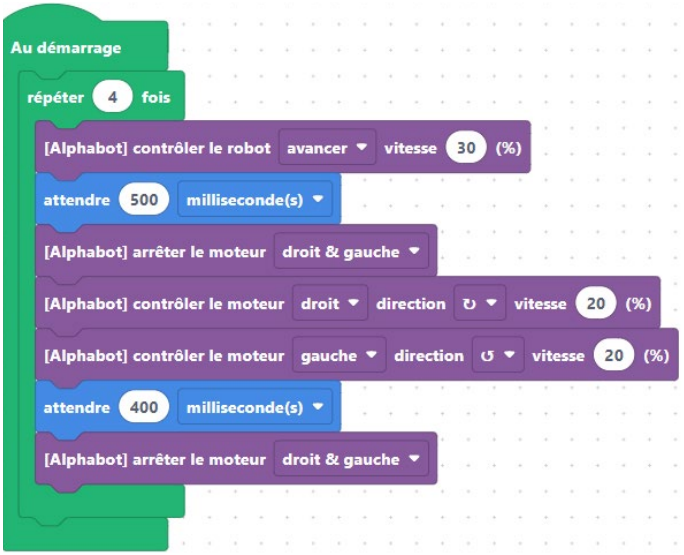
• Mission 2-3 : suivre une trajectoire

Vous savez avancer, reculer et tourner à 90°. Essayez à présent de suivre une trajectoire carrée !
Pour ce faire, nous utiliserons des blocs qui se trouvent dans :



Attention : l'algorigramme devient plus compliqué en écriture à cause de la boucle qui va compter combien on a fait de fois une série d'actions.
On va donc y voir apparaître une variable compteur.



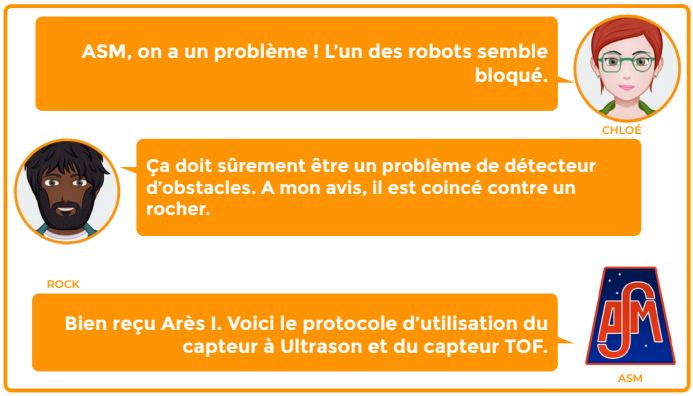



```python
import machine
from stm32_alphabot_v2 import AlphaBot_v2
import utime

alphabot = AlphaBot_v2()

for count in range(4):
  alphabot.moveForward(30)
  utime.sleep_ms(500)
  alphabot.stop()
  alphabot.setMotors(right=20)
  alphabot.setMotors(left=-20)
  utime.sleep_ms(400)
  alphabot.stop()

while True:
  pass
```

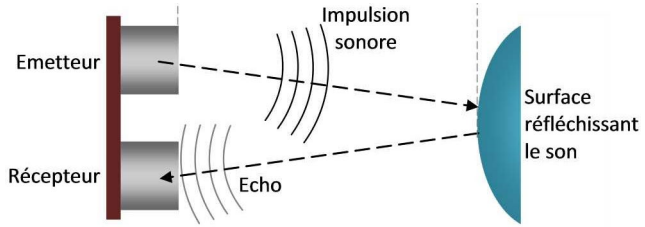
Mission • 3 ⏱ à l'appréciation de l'encadrant

Tester le détecteur d'obstacles

• Mission 3-1 : Découverte du capteur à ultrasons



Un capteur à ultrasons détecte les obstacles devant lui et détermine leur distance.

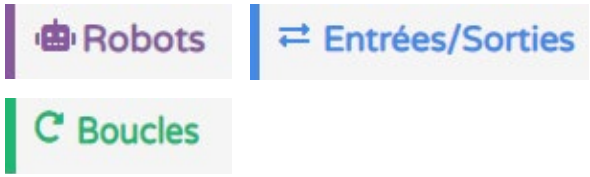


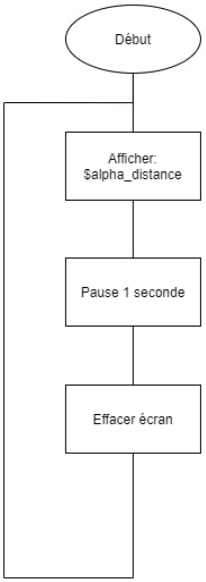
Son principe est le suivant :

- Un émetteur produit une impulsion sonore.
- Cette impulsion se propage jusqu'à rencontrer un obstacle (à la distance d) qui la réfléchit partiellement vers un récepteur (écho).
- Lorsque l'écho atteint le récepteur, le son a parcouru approximativement la distance 2d (inconnue) pendant la durée t (connue), à la vitesse V (connue) du son dans l'air. On en déduit d.

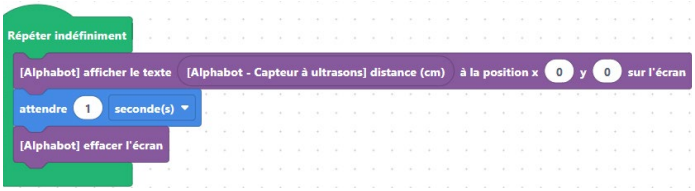
Pour cette mission nous allons afficher la distance d'un objet en permanence sur l'écran OLED.

Pour ce faire, nous utiliserons des blocs qui se trouvent dans :



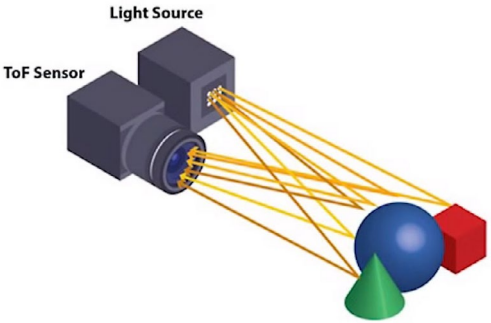



```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  oled.text(str(alphabot.readUltrasonicDistance()), 0, 0)
  oled.show()
  utime.sleep(1)
  oled.fill(0)
  oled.show()
```



• Mission 3-2 : Découverte du capteur Time of Flight (ToF)

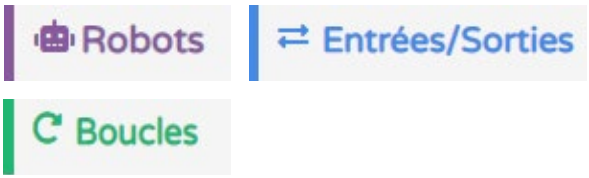
Un capteur "Time of Flight" détecte les obstacles devant lui et détermine leur distance.

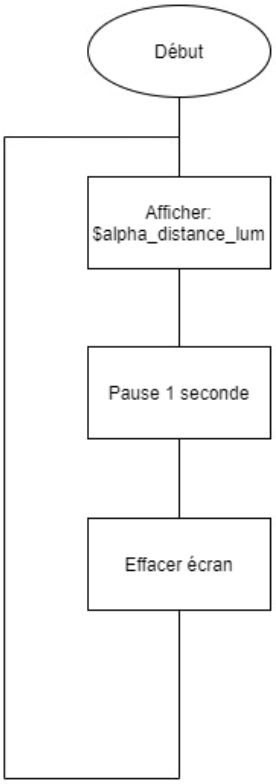


Son principe de fonctionnement est exactement le même que celui du capteur à ultrasons, mais il est plus rapide et précis. Le ToF envoie des ondes lumineuses (laser infrarouge) pour sonder l'espace devant lui, dont on connaît également la vitesse de propagation (celle de la lumière dans l'air).

Le capteur "Time of Flight" doit être branché sur le port Grove I2C du shield Nucleo.
Pour cette mission nous allons afficher la distance d'un objet en permanence sur l'écran OLED.

Pour ce faire, nous utiliserons des blocs qui se trouvent dans :

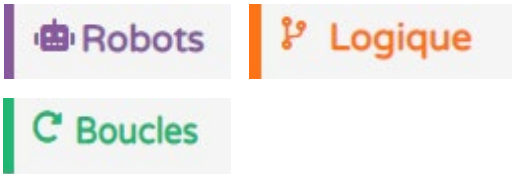




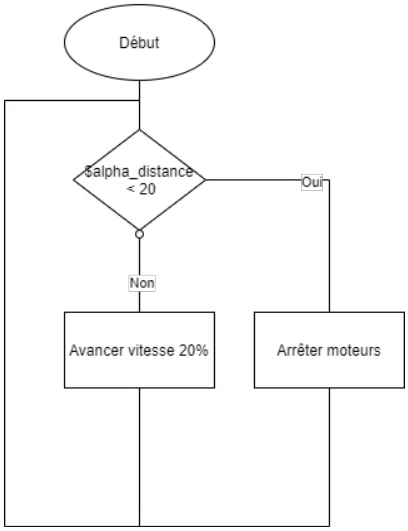
• Mission 3-3 : Capteurs de distance et moteurs

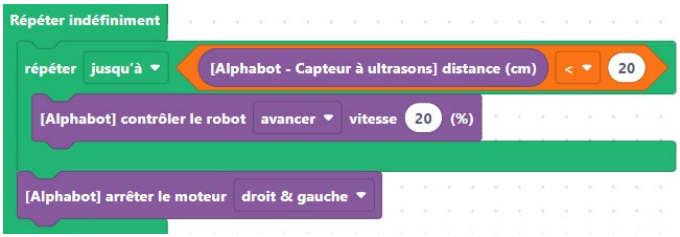
Nous allons tout de suite exploiter ce capteur à ultrasons pour le déplacement du robot. Celui-ci devra se déplacer jusqu'à ce que sa distance à un mur ait une valeur choisie (par exemple 20 cm au plus).

Pour ce faire, nous utiliserons des blocs qui se trouvent dans :



Nous allons utiliser dans cette mission les comparateurs logiques, qui permettent de tester si une information est vraie ou fausse. Selon le résultat, on décidera de réaliser telle ou telle action. Dans notre cas, la question sera : Est ce que la distance entre mon robot et le mur est inférieure à 20 cm ?





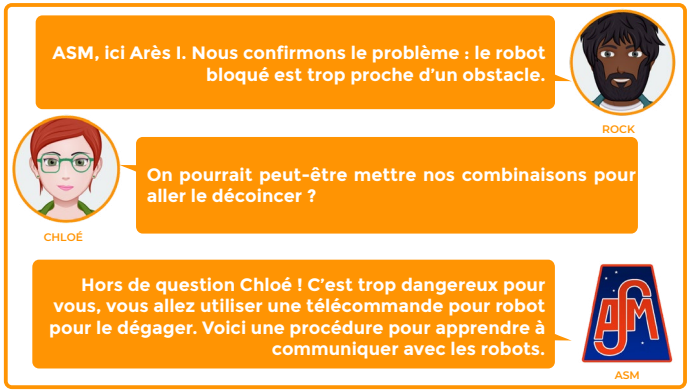

```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  while not alphabot.readUltrasonicDistance() < 20:
    alphabot.moveForward(20)
  alphabot.stop()
```

Si vous mesurez la distance obtenue après déplacement du robot, il se peut qu'il y ait une légère différence possiblement pour deux raisons :

- Une erreur de protocole : il ne faut pas mesurer au niveau des roues mais plutôt au niveau du capteur de distance.
- L'inertie du robot qui ne peut pas s'arrêter instantanément; plus sa vitesse sera grande, plus longue sera sa distance de freinage.
- Le microcontrôleur de la NUCLEO-WB55 a besoin de fractions de secondes pour traiter l'information de distance et envoyer l'ordre de freiner, qui ne sera pas exécuté instantanément non plus. Ces temps de latence peuvent se traduire par un excès de quelques millimètres parcourus lorsque la vitesse du robot est élevée.

Il est tout à fait possible d'utiliser le capteur ToF à la place du capteur à ultrasons, le programme sera identique excepté pour le nom du capteur utilisé.

Mission • 4 ⏱ à l'appréciation de l'encadrant
Pilotage temps réel



Nous allons à présent piloter le robot à l'aide de sa télécommande.

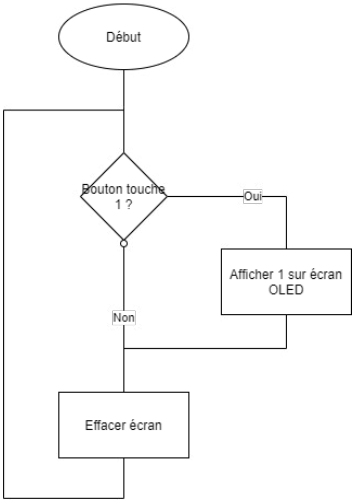
L'idée est que le robot reçoive des informations et réagisse en fonction des informations qu'il reçoit.

• Mission 4-1 : La télécommande

Dans une première mission nous souhaitons afficher sur l'écran OLED du robot une information en correspondance avec la touche appuyée de la télécommande. Par exemple, si nous appuyons sur la touche 1 l'écran devrait afficher 1.

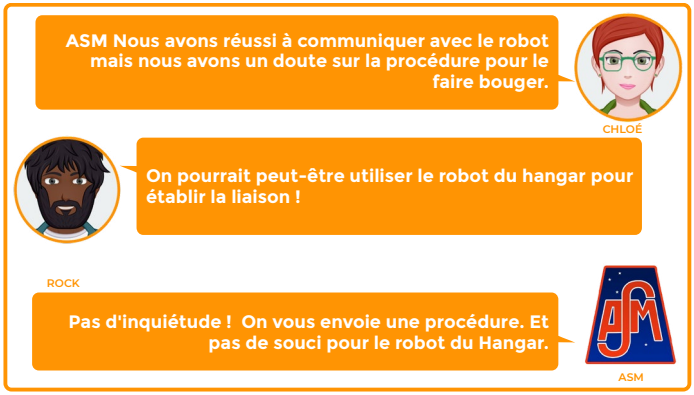
Pour ce faire, nous utiliserons des blocs qui se trouvent dans :

Robots | Boucles



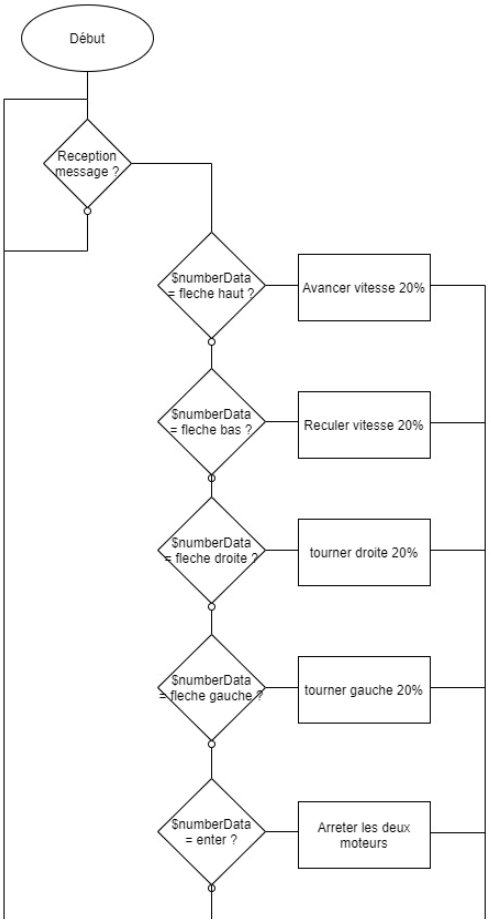

```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime
from stm32_nec import NEC_8, NEC_16
import gc

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()
ir_current_remote_code = None

def remoteNEC_basicBlack_getButton(hexCode):
  if hexCode == 0x0c: return "1"
  elif hexCode == 0x18: return "2"
  elif hexCode == 0x5e: return "3"
  elif hexCode == 0x08: return "4"
  elif hexCode == 0x1c: return "5"
  elif hexCode == 0x5a: return "6"
  elif hexCode == 0x42: return "7"
  elif hexCode == 0x52: return "8"
  elif hexCode == 0x4a: return "9"
  elif hexCode == 0x16: return "0"
  elif hexCode == 0x40: return "up"
  elif hexCode == 0x19: return "down"
  elif hexCode == 0x07: return "left"
  elif hexCode == 0x09: return "right"
  elif hexCode == 0x15: return "enter_save"
  elif hexCode == 0x0d: return "back"
  elif hexCode == 0x45: return "volMinus"
  elif hexCode == 0x47: return "volPlus"
```

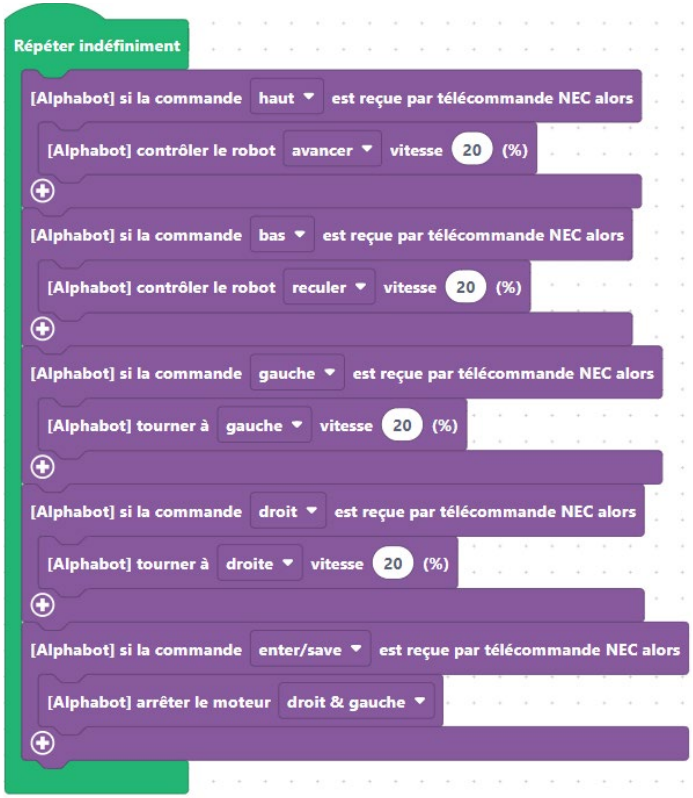
```
  elif hexCode == 0x46: return "play_pause"
  elif hexCode == 0x44: return "setup"
  elif hexCode == 0x43: return "stop_mode"
  else: return "NEC remote code error"

def remoteNEC_callback(data, addr, ctrl):
  global ir_current_remote_code
  if data < 0:  # NEC protocol sends repeat codes.
    print('Repeat code.')
  else:
    print('Data {:02x} Addr {:04x} Ctrl {:02x}'.format(data, addr, ctrl))
    ir_current_remote_code = remoteNEC_basicBlack_getButton(data)

classes = (NEC_8, NEC_16)
ir_remote = classes[0](alphabot.pin_IR, remoteNEC_callback)

oled.fill(0)
oled.show()

while True:
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "1":
    oled.text('1', 0, 0)
    oled.show()
  utime.sleep(1)
  oled.fill(0)
  oled.show()
```



• Mission 4-2 : Test en conditions réelles "martiennes"

Ce nouvel exercice va consister à piloter notre petit robot dans le hangar. Pour cela nous utiliserons les programmes réalisés dans les exercices qui précèdent.
L'objectif sera de faire avancer le robot avec la flèche du haut, de le faire tourner avec les flèches droite et gauche, de le faire reculer avec la flèche du bas et de l'arrêter avec le bouton "Entrée" (Enter sur la télécommande).



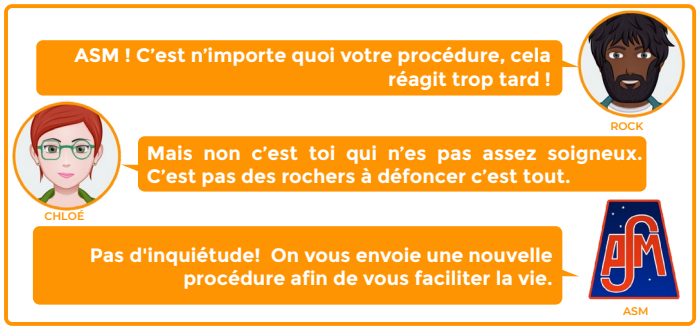



```
while True:
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "up":
    alphabot.moveForward(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "down":
    alphabot.moveBackward(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "left":
    alphabot.turnLeft(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "right":
    alphabot.turnRight(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "enter_save":
    alphabot.stop()
```

Le code Python est volontairement incomplet, tout le début du programme est identique au précédent et on retrouve ici seulement les appels de fonctions.

Suite à cette simulation, on constate qu'il est est très complexe de piloter le robot avec la télécommande. Une solution plus efficace consiste à le programmer pour qu'il gère ses déplacements de façon autonome sur la base de la vitesse de rotation de ses moteurs mais aussi d'informations sur son environnement, obtenues par les capteurs dont il est équipé.
Sur commande, le robot devra s'adapter pour accomplir au mieux sa tâche. Par exemple, si on lui donne la consigne "Allez à 100 mètres au Nord !" il devra grâce à sa programmation relier le point de rendez-vous qui se situe 100 mètres au nord.

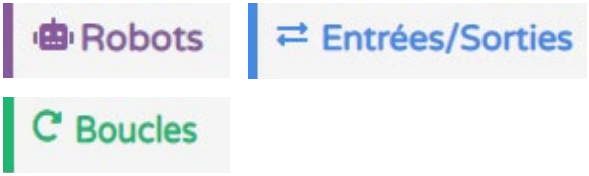
Mission • 5 ⏱ à l'appréciation de l'encadrant
Déplacement en mode cases



• Mission 5-1 : Déplacement en mode case

Pour cette mission nous allons programmer les déplacements du robot en mode cases. Le robot fera de petits déplacements comme s'il était sur un échiquier ou un plateau de dames. Cela permettra de simplifier la programmation pour débuter et de lui demander, par exemple, de se déplacer de 3 cases en avant et de 2 cases vers la droite.
Pour commencer nous allons exploiter les missions précédentes avec la rotation à 90° qui conviendra parfaitement pour un déplacement sur case. Ensuite, nous mesurerons combien de temps en déplacement tout droit il faut pour avancer d'une case.
Nous allons fixer la taille de la case à 10 cm ce qui permettra de faire un quadrillage de 4 cases par 3 cases sur une feuille A3.
Le code de mission que nous allons utiliser est celui de la 2-1 et de la 2-2, ainsi que celui de la mission 4-2.

Nous allons éviter de tout programmer à nouveau et améliorer le code déjà écrit comme c'est souvent le cas en informatique.
Pour ce faire, nous utiliserons des blocs qui se trouvent dans :

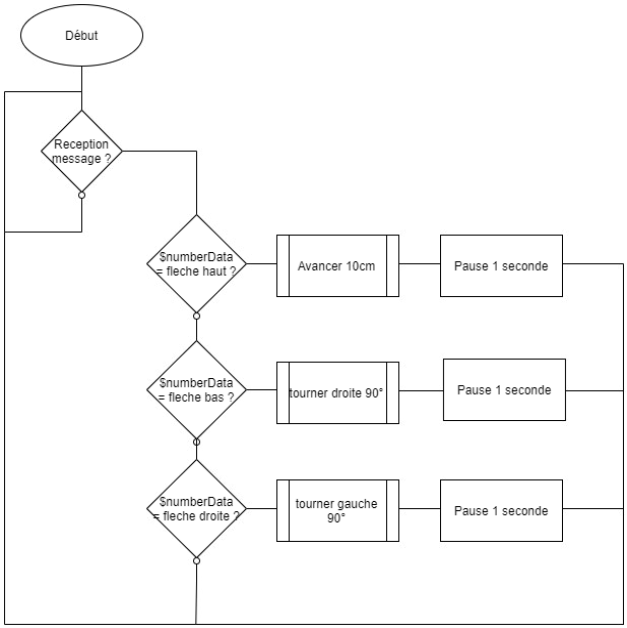


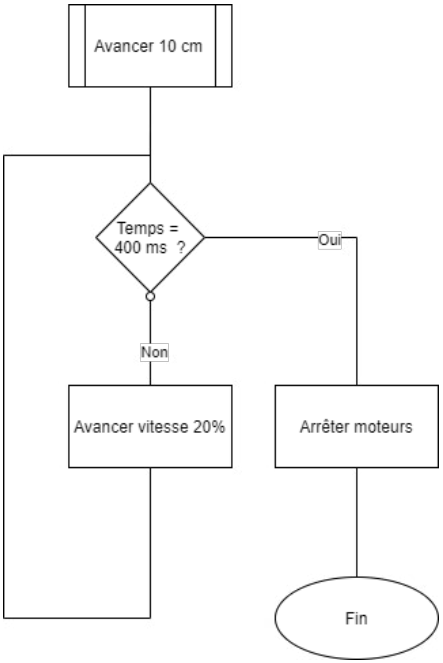
On commence en créant trois fonctions : une pour avancer, une pour tourner à droite et une pour tourner à gauche.

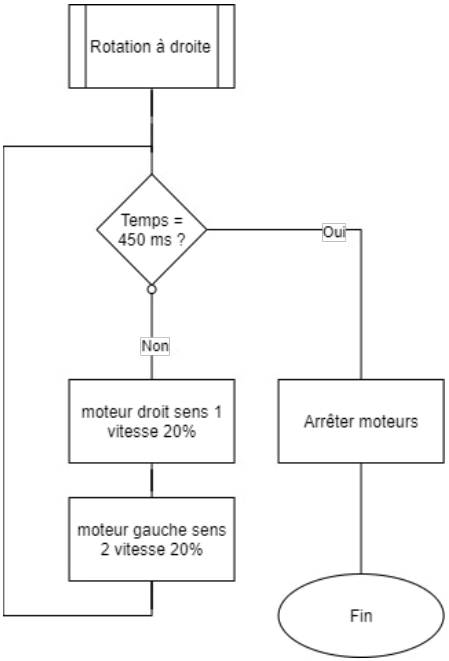
Il n'y a pas de changement par rapport à la mission 2-2 sauf qu'au lieu de mettre les blocs dans la boucle "Répéter indéfiniment" ou "Au démarrage", nous allons les mettre dans un bloc "Fonction" auquel nous donnerons un nom reconnaissable afin de ne pas nous perdre dans le code.

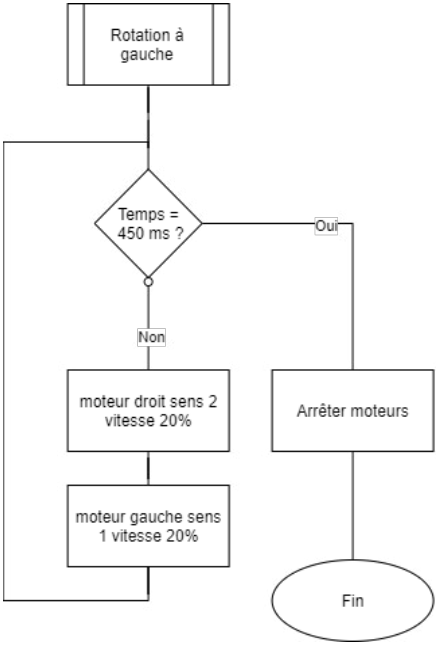
Nous rajouterons un bloc de conditions afin que le robot exécute la fonction lorsque l'on appuie sur le bon bouton de la télécommande, ainsi qu'une seconde de délai afin d'avoir le temps d'appuyer sur une touche de la télécommande en zone morte (touche non affectée).

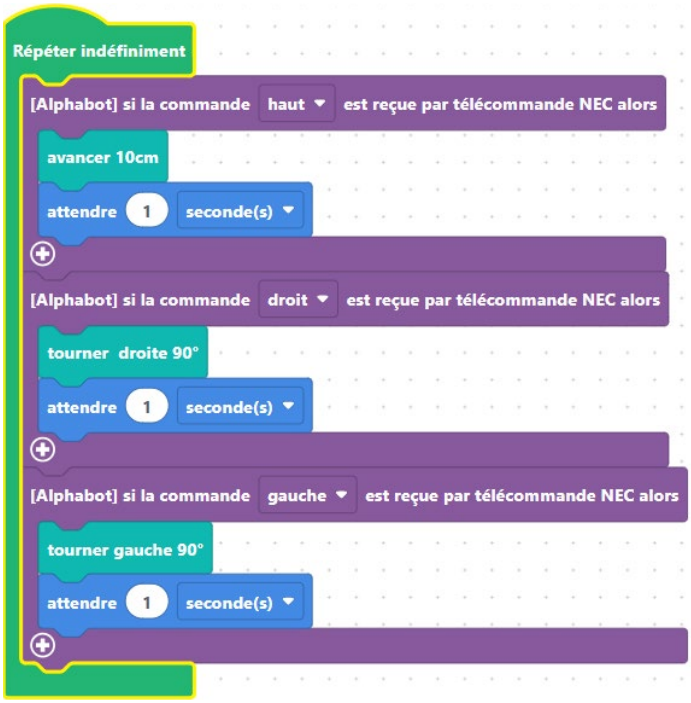
Cela donne les algorigrammes présentés sur les pages suivantes :


Début
Reception message ?
$numberData = fleche haut ?
Avancer 10cm
Pause 1 seconde
$numberData = fleche bas ?
tourner droite 90°
Pause 1 seconde
$numberData = fleche droite ?
tourner gauche 90°
Pause 1 seconde

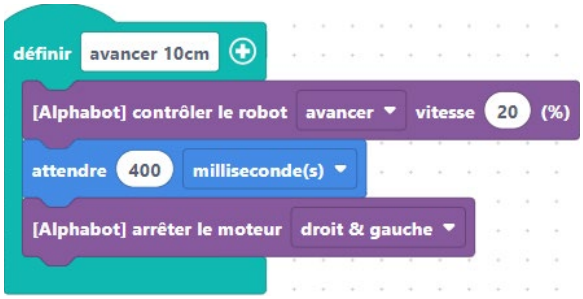

Avancer 10 cm
Temps = 400 ms ?
Oui
Non
Avancer vitesse 20%
Arrêter moteurs
Fin


Rotation à droite
Temps = 450 ms ?
Oui
Non
moteur droit sens 1 vitesse 20%
moteur gauche sens 2 vitesse 20%
Arrêter moteurs
Fin


Rotation à gauche
Temps = 450 ms ?
Oui
Non
moteur droit sens 2 vitesse 20%
moteur gauche sens 1 vitesse 20%
Arrêter moteurs
Fin


Répéter indéfiniment
[Alphabot] si la commande haut est reçue par télécommande NEC alors
avancer 10cm
attendre 1 seconde(s)
[Alphabot] si la commande droit est reçue par télécommande NEC alors
tourner droite 90°
attendre 1 seconde(s)
[Alphabot] si la commande gauche est reçue par télécommande NEC alors
tourner gauche 90°
attendre 1 seconde(s)


définir tourner droite 90°
[Alphabot] contrôler le moteur droit direction ↻ vitesse 20 (%)
[Alphabot] contrôler le moteur gauche direction ↺ vitesse 20 (%)
attendre 450 milliseconde(s)
[Alphabot] arrêter le moteur droit & gauche


définir avancer 10cm
[Alphabot] contrôler le robot avancer vitesse 20 (%)
attendre 400 milliseconde(s)
[Alphabot] arrêter le moteur droit & gauche


définir tourner gauche 90°
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 20 (%)
[Alphabot] contrôler le moteur droit direction ↺ vitesse 20 (%)
attendre 450 milliseconde(s)
[Alphabot] arrêter le moteur droit & gauche

```
45 def tourner_gauche_90_C2_B0():
46   alphabot.setMotors(left=20)
47   alphabot.setMotors(right=-20)
48   utime.sleep_ms(450)
49   alphabot.stop()
50
51 def tourner__droite_90_C2_B0():
52   alphabot.setMotors(right=20)
53   alphabot.setMotors(left=-20)
54   utime.sleep_ms(450)
55   alphabot.stop()
56
57 def avancer_10cm():
58   alphabot.moveForward(20)
59   utime.sleep_ms(400)
60   alphabot.stop()
```

```
62 while True:
63   utime.sleep_ms(150)
64   gc.collect()
65   if ir_current_remote_code == "up":
66     avancer_10cm()
67     utime.sleep(1)
68   utime.sleep_ms(150)
69   gc.collect()
70   if ir_current_remote_code == "right":
71     tourner__droite_90_C2_B0()
72     utime.sleep(1)
73   utime.sleep_ms(150)
74   gc.collect()
75   if ir_current_remote_code == "left":
76     tourner_gauche_90_C2_B0()
77     utime.sleep(1)
78
```

Les fonctions simplifient la programmation. On peut les appeler dans le programme principal pour qu'elles s'exécutent d'une traite.

Comme on peut le constater, les fonctions rendent le programme beaucoup plus lisible en nous évitant de répéter un grand nombre de fois leurs lignes de code dans le programme principal.

• Mission 5-2 : Détection d'obstacle avant déplacement

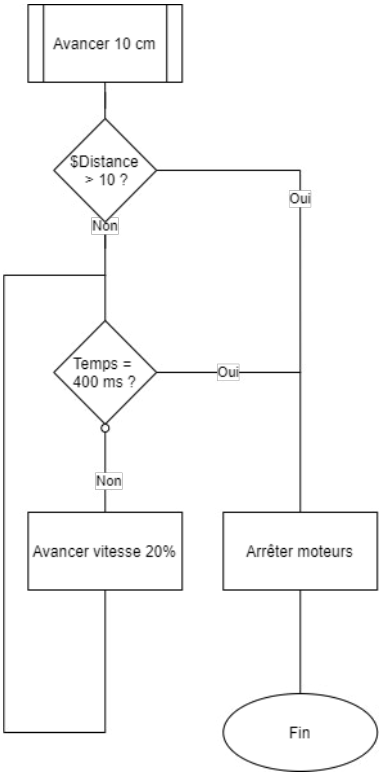
Dans cette mission nous allons rajouter les éléments de la mission 3. En effet, se déplacer sans regarder où l'on va risque de rapidement poser des problèmes. Dans la vie de tous les jours notre vue en 3 dimensions nous permet de repérer les obstacles et d'évaluer leur distance. Pour notre robot, ce seront le capteur à ultrasons ou le capteur Time of Flight qui joueront le rôle de nos yeux.

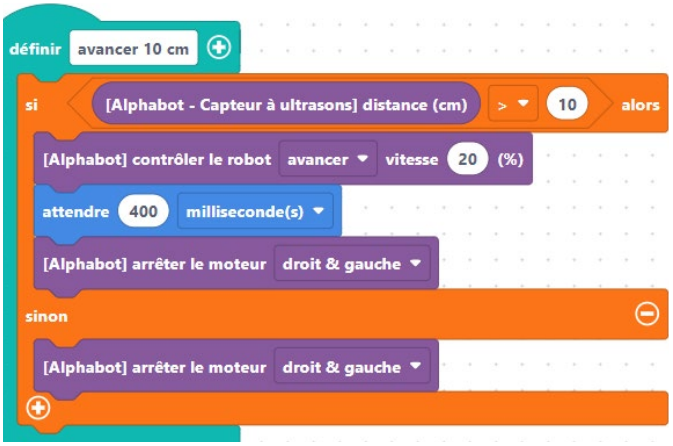
L'objectif de cette mission sera de se déplacer sur le damier en vérifiant avant chaque mouvement si la case de destination est libre. Dans le cas contraire, on décidera de ne pas avancer sur la case encombrée et d'afficher un "X" sur l'écran OLED du robot.

Nous allons uniquement modifier la fonction avancer pour que celle-ci détecte les obstacles.

Pour ce faire, nous utiliserons des blocs qui se trouvent dans :

Ici, seuls les changement dans la fonction « Avancer 10 cm » sont présentés.

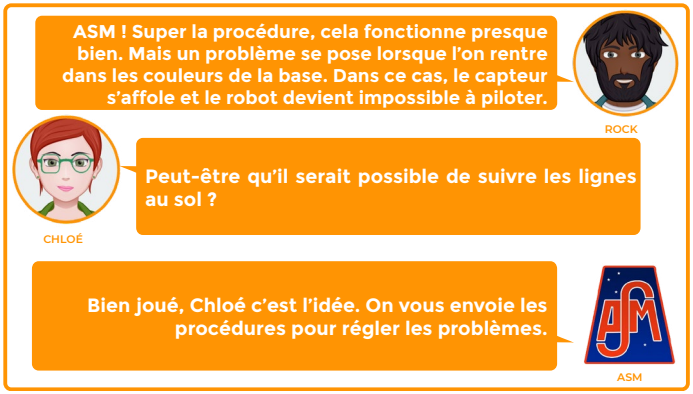




```
45 def avancer_10_cm():
46   if alphabot.readUltrasonicDistance() > 10:
47     alphabot.moveForward(20)
48     utime.sleep_ms(400)
49     alphabot.stop()
50   else:
51     alphabot.stop()
```

La fonction Python relative au au capteur à ultrasons n'est pas ajoutée ici. Si nécessaire, elle est disponible dans la mission 3.

Nous voilà déjà bien avancés, notre robot est maintenant capable de se déplacer seul en évitant les obstacles ou plutôt en les ignorant !



Mission • 6 ⏱ à l'appréciation de l'encadrant
Suiveur de ligne

• Mission 6-1 : Suivre une ligne

Le suivi de ligne est assuré par cinq capteurs constitués chacun d'une LED (émetteur) et d'un photo-transistor (récepteur), situés sous le châssis du robot. La LED émet un signal et le photo-transistor détecte sa réflexion. Le noir absorbant le rayonnement infrarouge, le récepteur ne détecte pas le signal émis lorsque le capteur est au-dessus de la ligne, celui-ci retourne donc 0. Dans le cas contraire, le signal est réfléchi par le blanc, le capteur retourne 1.

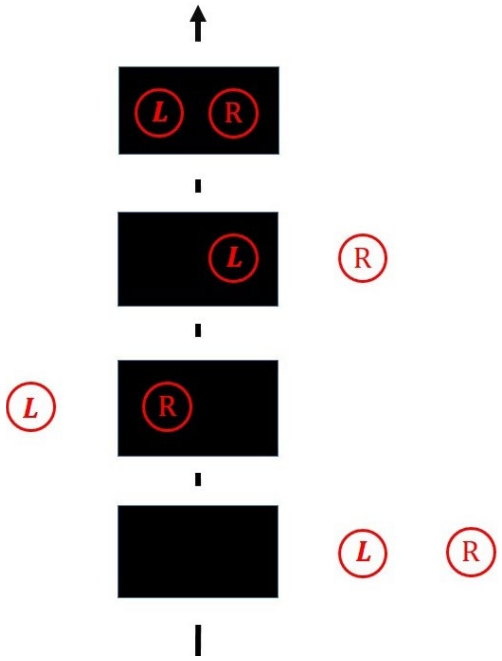
Dans un premier temps nous n'allons utiliser que deux capteurs pour suivre la ligne.

• **Cas 1:** Les 2 capteurs sont au-dessus de la ligne noire.
-> Retourne 0,0

• **Cas 2:** Line-L est au-dessus de la ligne et la Line-R est à l'extérieur.
-> Retourne 0,1

• **Cas 3:** Line-L est à l'extérieur et Line-R est au-dessus de la ligne.
-> Retourne 1,0

• **Cas 4:** Les 2 capteurs sont à l'extérieur de la ligne noire.
-> Retourne 1,1



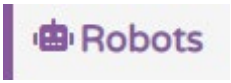
La figure ci-dessus illustre le fonctionnement des capteurs et l'information que l'on récupère au niveau de la NUCLEO-WB55.

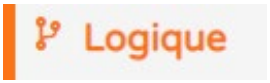
Sur la base de ces informations, nous allons écrire un algorithme pour gérer les moteurs.
Pour faciliter la tâche, nous allons utiliser un tableau logique dans lequel figurent deux colonnes pour les capteurs et deux colonnes pour les moteurs. Nous utiliserons le code 0 pour "moteur éteint" et couleur blanche et 1 pour "moteur actif" et couleur noire.

Capteur gauche	Capteur droite	Moteur gauche	Moteur droite
0 / blanc	0 / blanc	?	?
0 / blanc	1 / noir	1 / activer	0 / éteint
1 / noir	0 / blanc	0 / éteint	1 / activer
1 / noir	1 / noir	1 / activer	1 / activer

On constate que dans le premier cas, la ligne n'est plus détectée et qu'il faut mettre en place une stratégie pour la retrouver. Ce pourrait être par exemple être par exemple « avancer de 5 cm puis tourner en rond jusqu'à détection de ligne » ou alors simplement « reculer » car on est allé trop vite et on a perdu la ligne.

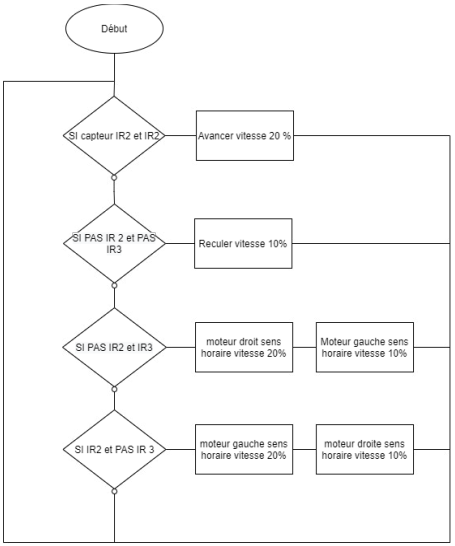
Programmons cela !
Pour ce faire, nous utiliserons des blocs qui se trouvent dans :



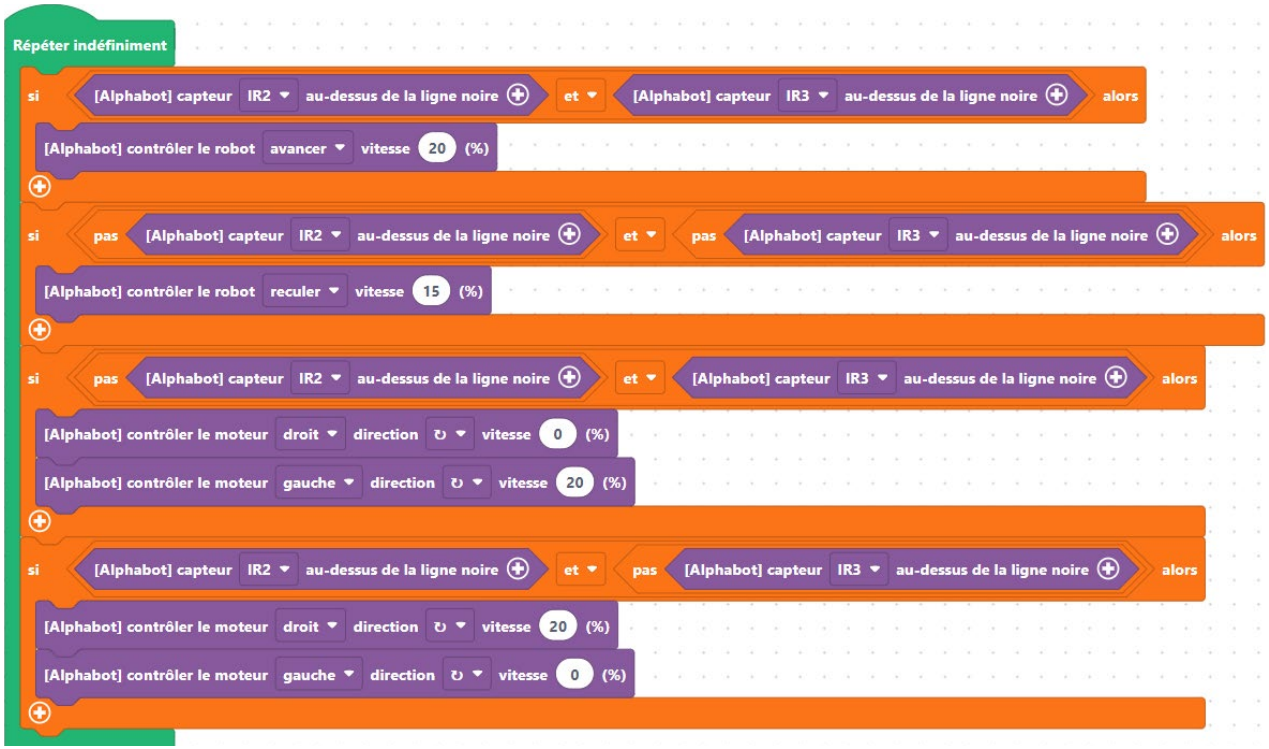


On constate que le bloc “Au démarrage” du programme contient un bloc “arrêter le moteur”. C'est une mini initialisation, cela permet d'être sûr que l'on commence moteur éteint. Dans cette initialisation, on peut rajouter de nombreuses instructions pour être sûr de démarrer le programme en mettant le robot dans un état déterminé ; par exemple un « effacer l'écran » pour être sûr que rien ne reste sur l'écran. On peut y ajouter une pause pour avoir le temps de bien positionner le robot avant que celui-ci ne démarre.

• 52




```python
while True:
  if isSensorAboveLine(alphabot, sensor='IR2') and isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.moveForward(20)
  if not isSensorAboveLine(alphabot, sensor='IR2') and not isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.moveBackward(15)
  if not isSensorAboveLine(alphabot, sensor='IR2') and isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.setMotors(right=0)
    alphabot.setMotors(left=20)
  if isSensorAboveLine(alphabot, sensor='IR2') and not isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.setMotors(right=20)
    alphabot.setMotors(left=0)
```

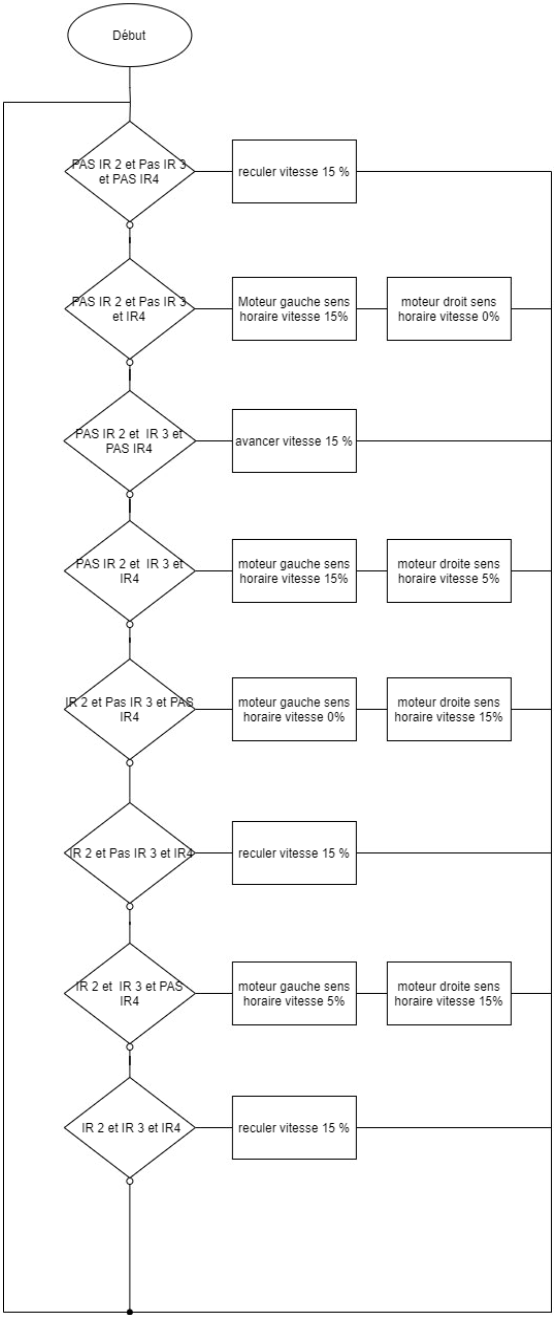


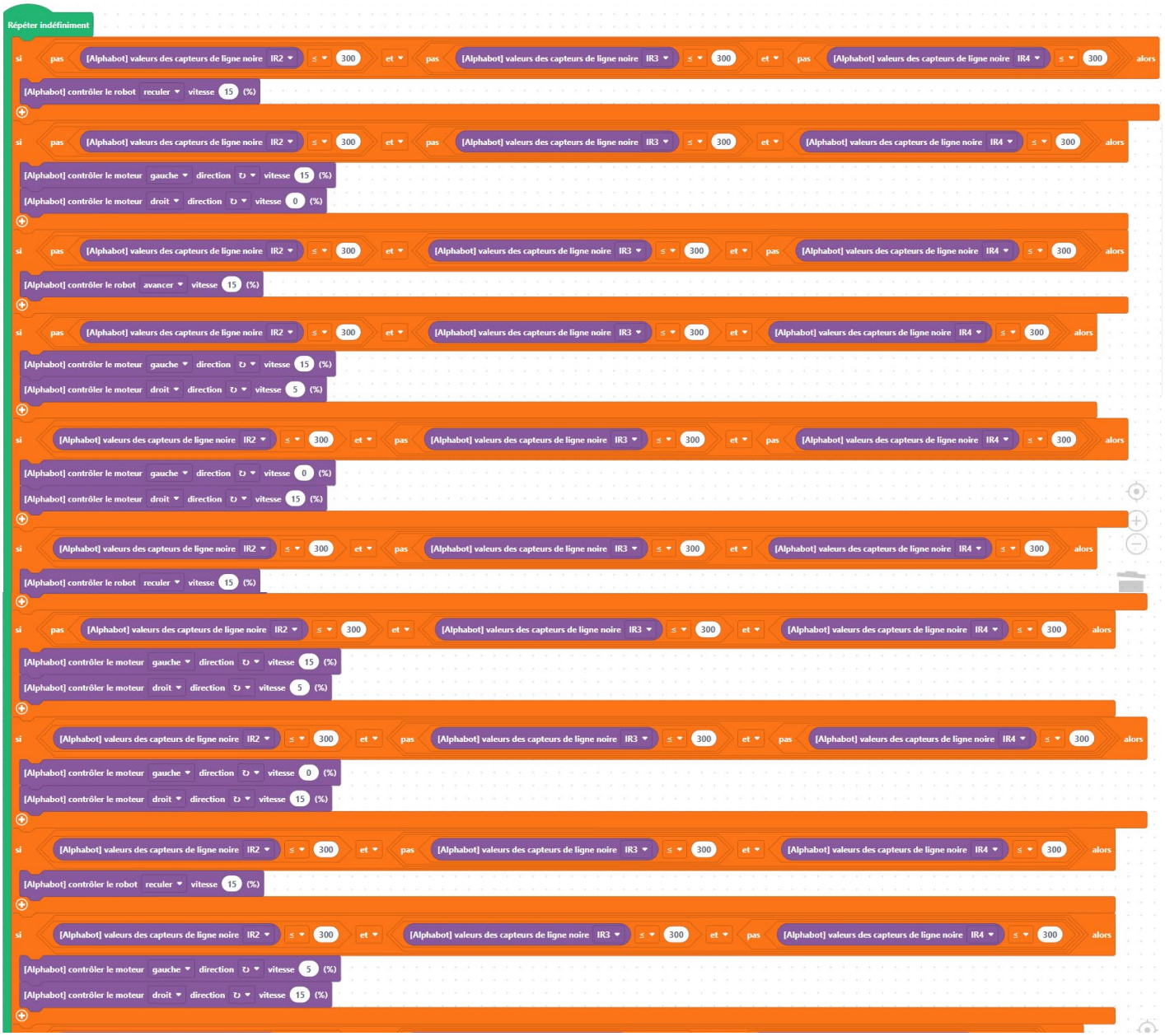
• Mission 6-2 : Suivre une ligne avec 3 capteurs

Essayons maintenant d'améliorer notre programme en utilisant trois capteurs de ligne : à gauche, au centre et à droite du robot. On utilisera le code suivant 0 pour blanc et 1 pour la ligne noire :

Capteur gauche	Capteur centre	Capteur droite	Moteur gauche	Moteur droite
0	0	0	-15%	-15%
0	0	1	15%	0
0	1	0	15%	15%
0	1	1	15%	5%
1	0	0	0	15%
1	0	1	-15%	-15%
1	1	0	5%	15%
1	1	1	-15%	-15%

```
while True:
  if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveBackward(15)
  if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=15)
    alphabot.setMotors(right=0)
  if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveForward(15)
  if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=15)
    alphabot.setMotors(right=5)
  if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=0)
    alphabot.setMotors(right=15)
  if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveBackward(15)
  if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=5)
    alphabot.setMotors(right=15)
  if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveBackward(15)
  oled.fill(0)
  oled.show()
```


Début
PAS IR 2 et Pas IR 3 et PAS IR4
reculer vitesse 15 %
PAS IR 2 et Pas IR 3 et IR4
Moteur gauche sens horaire vitesse 15%
moteur droit sens horaire vitesse 0%
PAS IR 2 et IR 3 et PAS IR4
avancer vitesse 15 %
PAS IR 2 et IR 3 et IR4
moteur gauche sens horaire vitesse 15%
moteur droite sens horaire vitesse 5%
IR 2 et Pas IR 3 et PAS IR4
moteur gauche sens horaire vitesse 0%
moteur droite sens horaire vitesse 15%
IR 2 et Pas IR 3 et IR4
reculer vitesse 15 %
IR 2 et IR 3 et PAS IR4
moteur gauche sens horaire vitesse 5%
moteur droite sens horaire vitesse 15%
IR 2 et IR 3 et IR4
reculer vitesse 15 %


Répéter indéfiniment
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot reculer vitesse 15 (%)
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 15 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 0 (%)
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot avancer vitesse 15 (%)
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 15 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 5 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 0 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 15 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot reculer vitesse 15 (%)
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 15 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 5 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 0 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 15 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot reculer vitesse 15 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 5 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 15 (%)

• Mission 6-3 : Suivre une ligne et éviter les obstacles

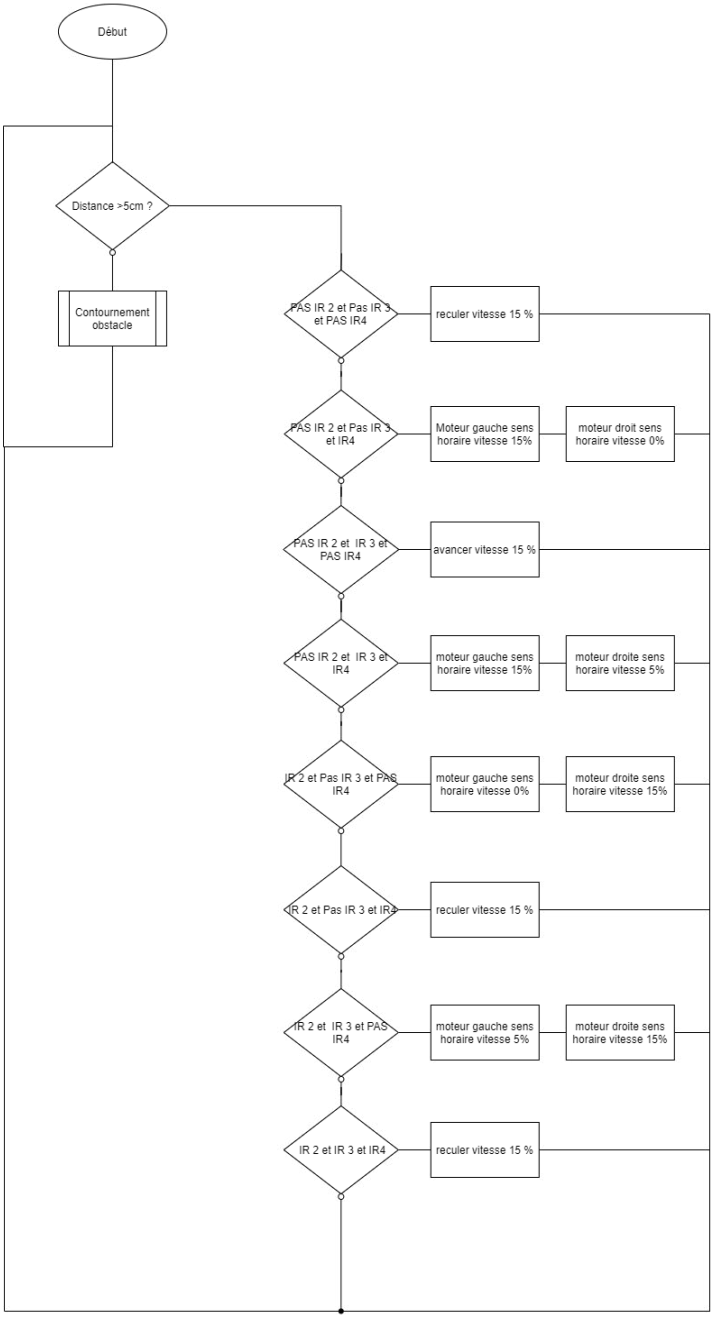
Pour cette mission, nous allons imaginer que le robot détecte un objet alors qu'il suit sa ligne. Dans ce cas de figure, il devra s'arrêter, le contourner puis retrouver la ligne pour continuer derrière lui.

Dans notre cas, nous limiterons la taille de l'objet gênant à 10 cm de diamètre.
Nous allons utiliser les fonctions développées pour la mission 5. On garde les fonctions "Avancer 10 cm" et "Tourner à droite" ou "Tourner à gauche" de 90°. On rajoute simplement une fonction de contournement d'obstacles et on améliore notre initialisation avec une pause.

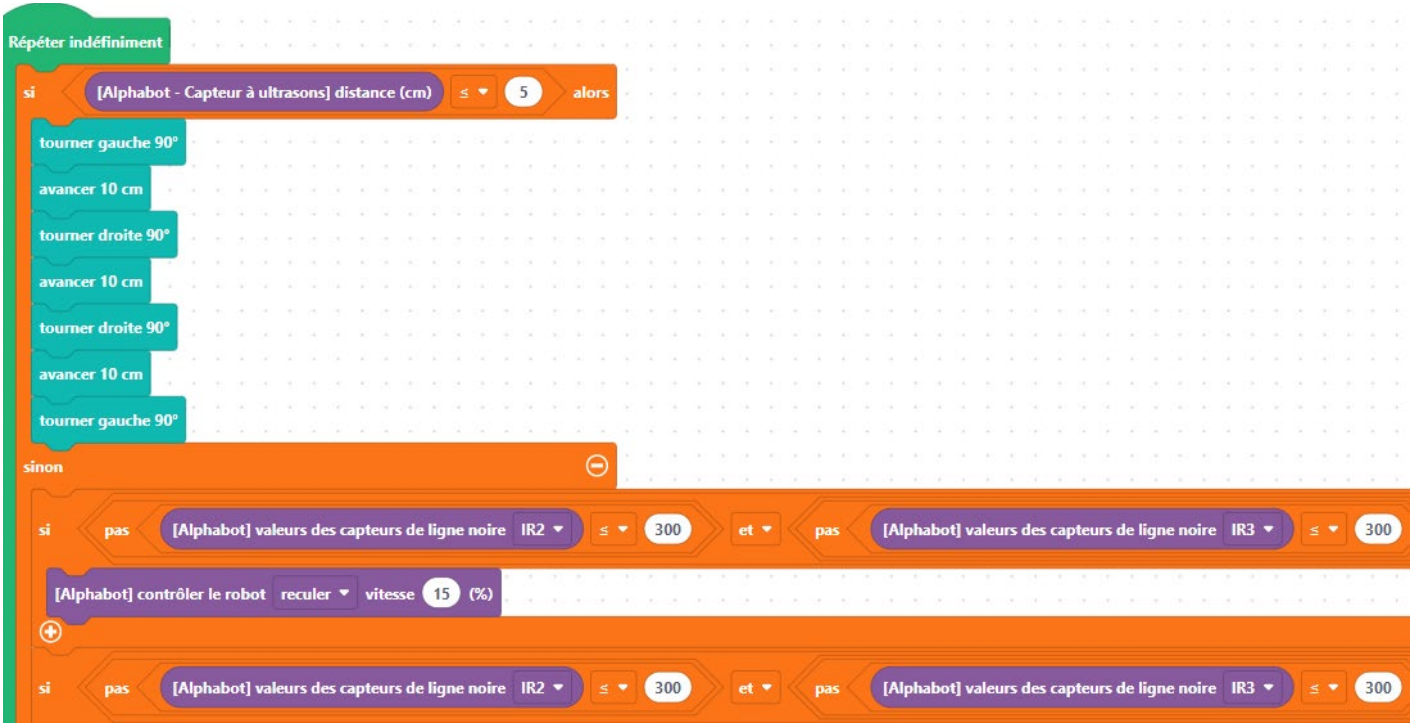
Essayons de mettre cela en place.
Pour cela nous aurons besoins des blocs suivants :

Les algorithmes et le programme deviennent plus conséquents mais on retrouve de nombreuses lignes de code déjà écrites avec très peu de changements, juste des compléments. Seules les parties importantes des algorithmes et des codes ont été reproduites dans leur intégralité.

```
while True:
  if alphabot.readUltrasonicDistance() <= 5:
    tourner_gauche_90_C2_B0()
    avancer_10_cm()
    tourner_droite_90_C2_B0()
    avancer_10_cm()
    tourner_droite_90_C2_B0()
    avancer_10_cm()
    tourner_gauche_90_C2_B0()
  else:
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=15)
      alphabot.setMotors(right=0)
    if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
```


Début
Distance >5cm ?
Contournement obstacle
PAS IR 2 et Pas IR 3 et PAS IR4
reculer vitesse 15 %
PAS IR 2 et Pas IR 3 et IR4
Moteur gauche sens horaire vitesse 15%
moteur droit sens horaire vitesse 0%
PAS IR 2 et IR 3 et PAS IR4
avancer vitesse 15 %
PAS IR 2 et IR 3 et IR4
moteur gauche sens horaire vitesse 15%
moteur droite sens horaire vitesse 5%
IR 2 et Pas IR 3 et PAS IR4
moteur gauche sens horaire vitesse 0%
moteur droite sens horaire vitesse 15%
IR 2 et Pas IR 3 et IR4
reculer vitesse 15 %
IR 2 et IR 3 et PAS IR4
moteur gauche sens horaire vitesse 5%
moteur droite sens horaire vitesse 15%
IR 2 et IR 3 et IR4
reculer vitesse 15 %

• 58

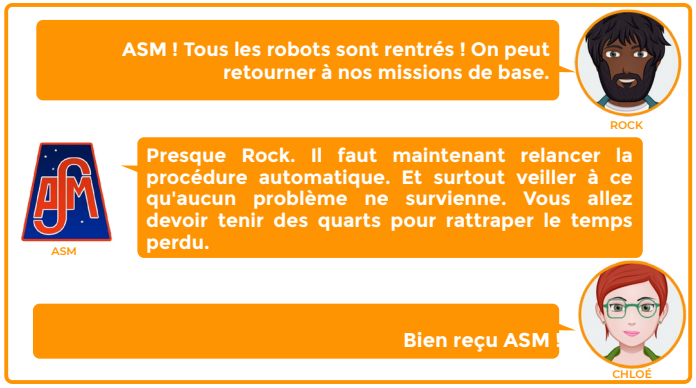

Répéter indéfiniment
si
[Alphabot - Capteur à ultrasons] distance (cm)
≤
5
alors
tourner gauche 90°
avancer 10 cm
tourner droite 90°
avancer 10 cm
tourner droite 90°
avancer 10 cm
tourner gauche 90°
sinon
si
pas
[Alphabot] valeurs des capteurs de ligne noire
IR2
≤
300
et
pas
[Alphabot] valeurs des capteurs de ligne noire
IR3
≤
300
[Alphabot] contrôler le robot
reculer
vitesse
15
(%)
si
pas
[Alphabot] valeurs des capteurs de ligne noire
IR2
≤
300
et
pas
[Alphabot] valeurs des capteurs de ligne noire
IR3
≤
300

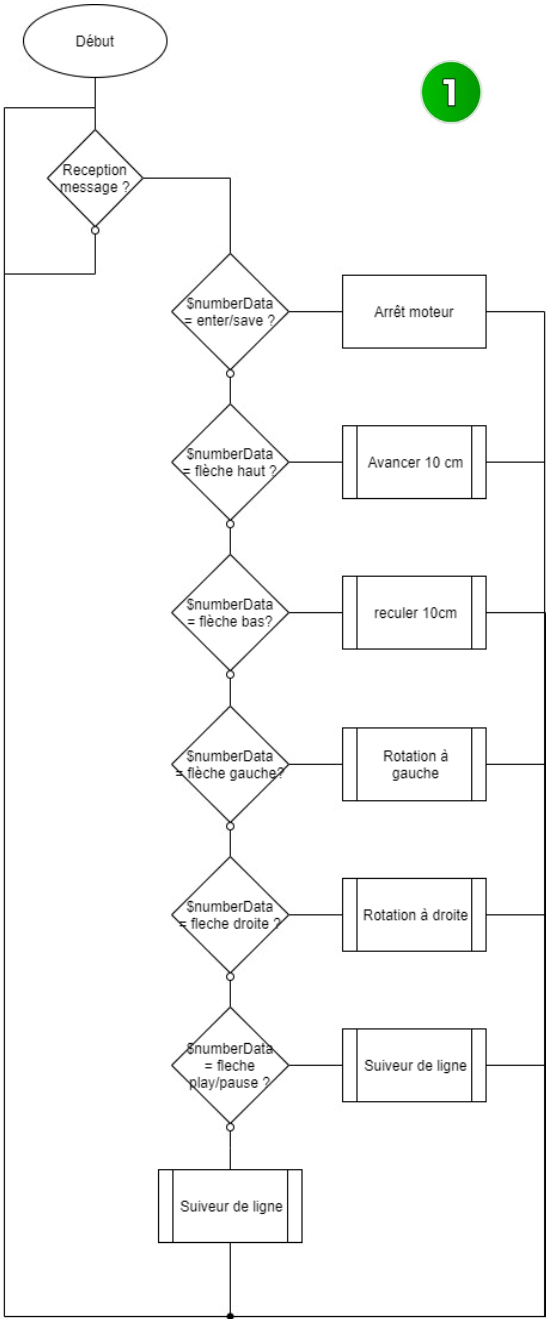
Mission • 7 ⏱ à l'appréciation de l'encadrant
Enfin la routine !

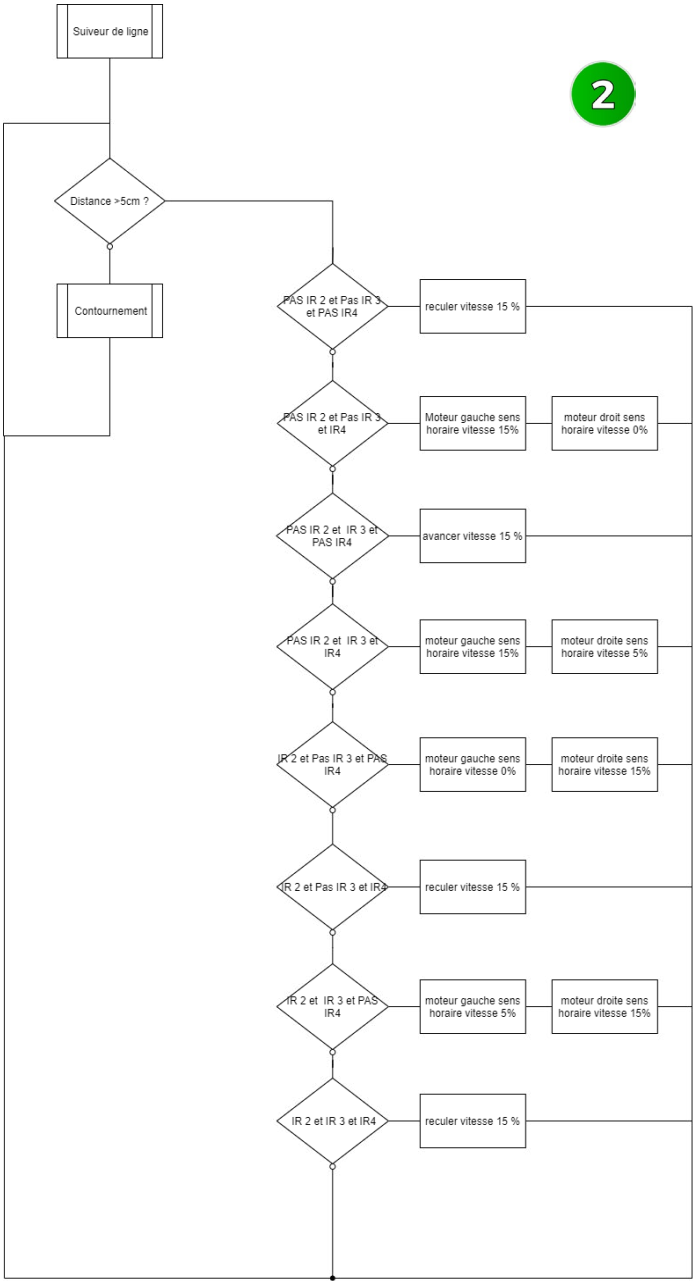
Mise en situation complexe : Le robot suit la boucle en permanence mais une chute de température brutale survient. Cet évènement le contraint à arrêter toute activité pour se mettre à l'abri. Pour cela on passe en commande manuelle et on planifie un retour au centre Ares I ou aux coordonnées indiquées. Les déplacements seront sécurisés en utilisant les capteurs à ultrason ou le capteur ToF pour vérifier que chaque mouvement est valide. Le pilotage à distance se fera avec la télécommande. Il faudra prévoir des déplacements de plusieurs natures.

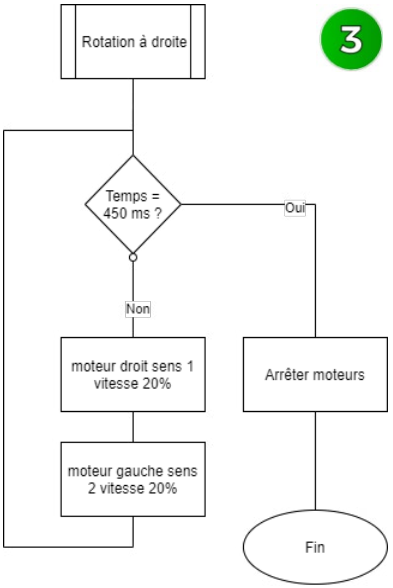
Attention, les boutons de la télécommande sont en nombre limité, il sera aussi nécessaire de laisser un bouton libre pour simuler un problème et ne pas oublier d'en réserver un autre pour relancer le robot en suiveur de ligne.

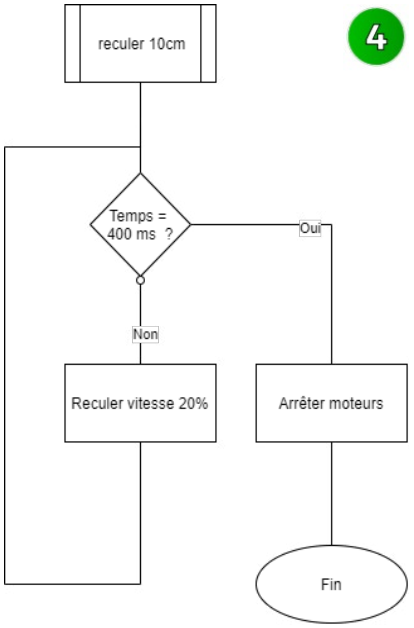
Le code est présenté ici dans son intégralité. Bien qu'il ne soit pas trop complexe, il devient conséquent en taille !

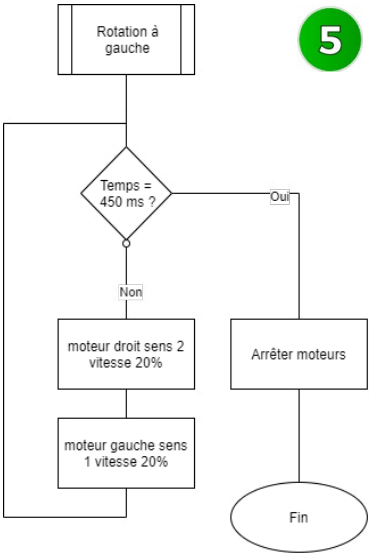


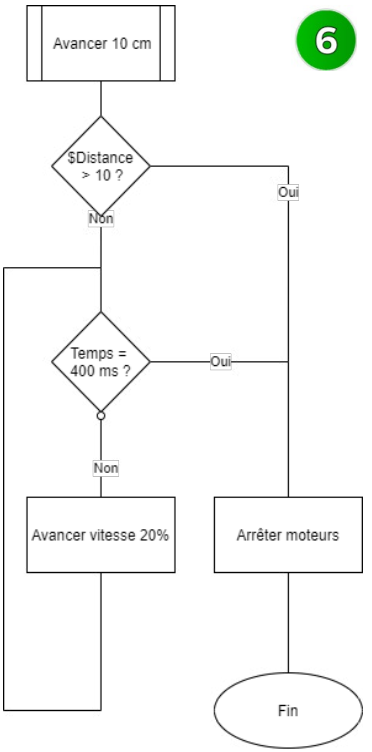


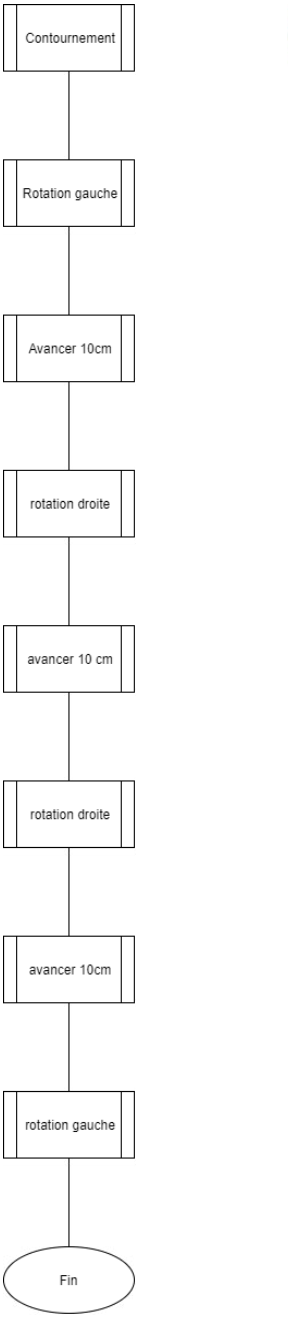

Suiveur de ligne
2
Distance >5cm ?
Contournement
PAS IR 2 et Pas IR 3 et PAS IR4
reculer vitesse 15 %
PAS IR 2 et Pas IR 3 et IR4
Moteur gauche sens horaire vitesse 15%
moteur droit sens horaire vitesse 0%
PAS IR 2 et IR 3 et PAS IR4
avancer vitesse 15 %
PAS IR 2 et IR 3 et IR4
moteur gauche sens horaire vitesse 15%
moteur droite sens horaire vitesse 5%
IR 2 et Pas IR 3 et PAS IR4
moteur gauche sens horaire vitesse 0%
moteur droite sens horaire vitesse 15%
IR 2 et Pas IR 3 et IR4
reculer vitesse 15 %
IR 2 et IR 3 et PAS IR4
moteur gauche sens horaire vitesse 5%
moteur droite sens horaire vitesse 15%
IR 2 et IR 3 et IR4
reculer vitesse 15 %

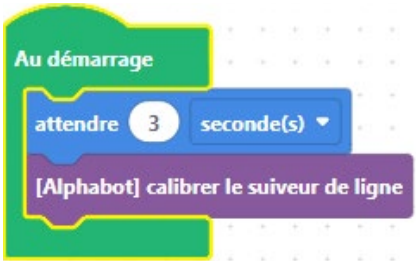

Rotation à droite
3
Temps = 450 ms ?
Oui
Non
moteur droit sens 1 vitesse 20%
moteur gauche sens 2 vitesse 20%
Arrêter moteurs
Fin


reculer 10cm
4
Temps = 400 ms ?
Oui
Non
Reculer vitesse 20%
Arrêter moteurs
Fin


Rotation à gauche
5
Temps = 450 ms ?
Oui
Non
moteur droit sens 2 vitesse 20%
moteur gauche sens 1 vitesse 20%
Arrêter moteurs
Fin


Avancer 10 cm
6
$Distance > 10 ?
Oui
Non
Temps = 400 ms ?
Oui
Non
Avancer vitesse 20%
Arrêter moteurs
Fin


Contournement
7
Rotation gauche
Avancer 10cm
rotation droite
avancer 10 cm
rotation droite
avancer 10cm
rotation gauche
Fin

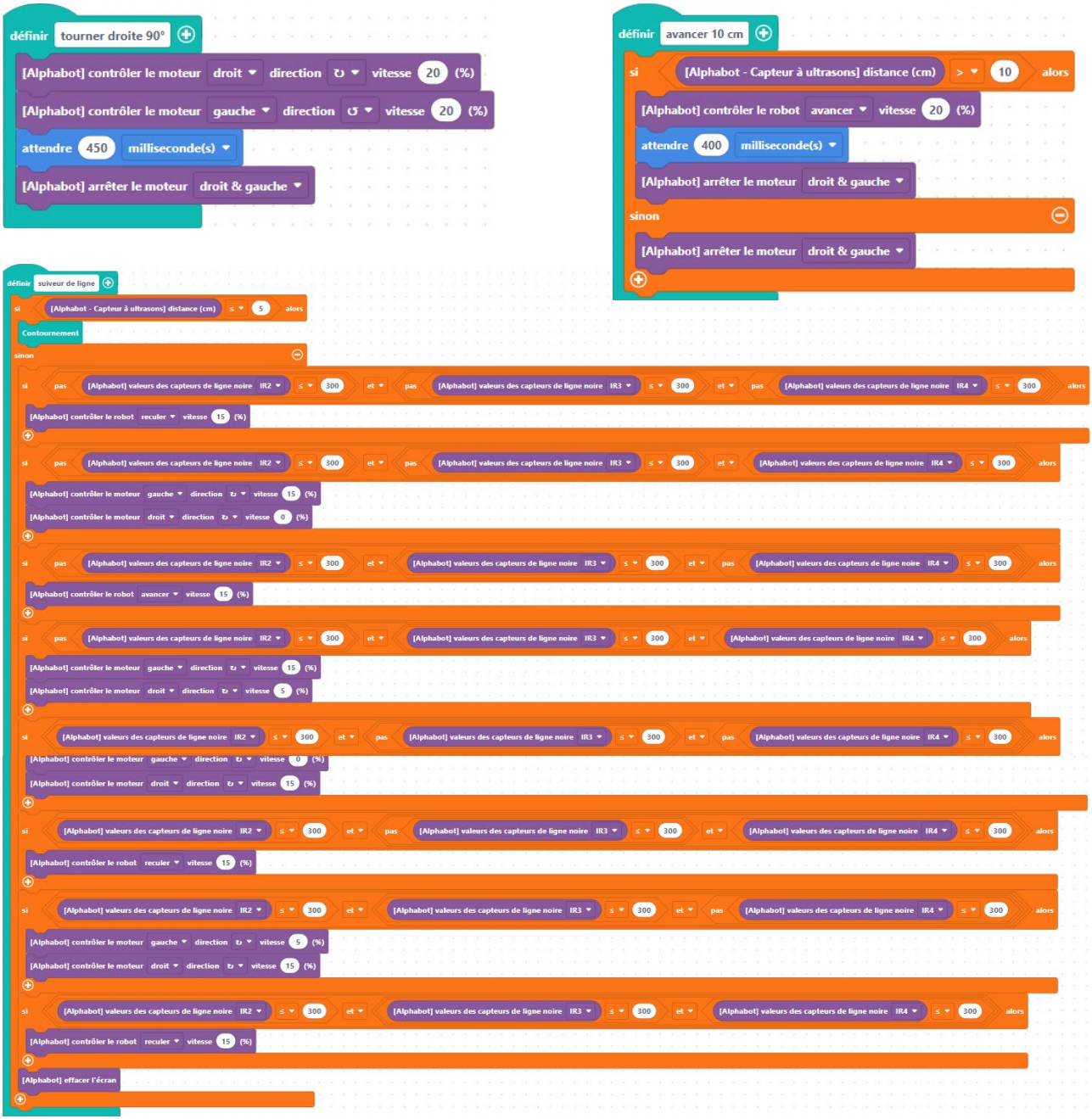

Au démarrage
attendre 3 seconde(s)
[Alphabot] calibrer le suiveur de ligne


définir Contournement
tourner gauche 90°
avancer 10 cm
tourner droite 90°
avancer 10 cm
tourner droite 90°
avancer 10 cm
tourner gauche 90°


définir reculer 10
[Alphabot] contrôler le robot reculer vitesse 20 (%)
attendre 400 milliseconde(s)
[Alphabot] arrêter le moteur droit & gauche


Répéter indéfiniment
[Alphabot] si la commande enter/save est reçue par télécommande NEC alors
[Alphabot] arrêter le moteur droit & gauche
sinon si haut est reçue alors
avancer 10 cm
sinon si bas est reçue alors
reculer 10
sinon si gauche est reçue alors
tourner gauche 90°
sinon si droit est reçue alors
tourner droite 90°
sinon si play/pause est reçue alors
suiveur de ligne
sinon
suiveur de ligne


définir tourner gauche 90°
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 20 (%)
[Alphabot] contrôler le moteur droit direction ↺ vitesse 20 (%)
attendre 450 milliseconde(s)
[Alphabot] arrêter le moteur droit & gauche


définir tourner droite 90°
[Alphabot] contrôler le moteur droit direction ↻ vitesse 20 (%)
[Alphabot] contrôler le moteur gauche direction ↺ vitesse 20 (%)
attendre 450 milliseconde(s)
[Alphabot] arrêter le moteur droit & gauche
définir avancer 10 cm
si [Alphabot - Capteur à ultrasons] distance (cm) > 10 alors
[Alphabot] contrôler le robot avancer vitesse 20 (%)
attendre 400 milliseconde(s)
[Alphabot] arrêter le moteur droit & gauche
sinon
[Alphabot] arrêter le moteur droit & gauche
définir suiveur de ligne
si [Alphabot - Capteur à ultrasons] distance (cm) ≤ 5 alors
Contournement
sinon
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot reculer vitesse 15 (%)
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 15 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 0 (%)
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot avancer vitesse 15 (%)
si pas [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 15 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 5 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 0 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 15 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot reculer vitesse 15 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et pas [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le moteur gauche direction ↻ vitesse 5 (%)
[Alphabot] contrôler le moteur droit direction ↻ vitesse 15 (%)
si [Alphabot] valeurs des capteurs de ligne noire IR2 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR3 ≤ 300 et [Alphabot] valeurs des capteurs de ligne noire IR4 ≤ 300 alors
[Alphabot] contrôler le robot reculer vitesse 15 (%)
[Alphabot] effacer l'écran

```
import machine
from stm32_alphabot_v2 import AlphaBot_v2
import utime
from stm32_nec import NEC_8, NEC_16
import gc
from stm32_ssd1306 import SSD1306, SSD1306_I2C

alphabot = AlphaBot_v2()
ir_current_remote_code = None
oled = SSD1306_I2C(128, 64, machine.I2C(1))

def remoteNEC_basicBlack_getButton(hexCode):
  if hexCode == 0x0c: return "1"
  elif hexCode == 0x18: return "2"
  elif hexCode == 0x5e: return "3"
  elif hexCode == 0x08: return "4"
  elif hexCode == 0x1c: return "5"
  elif hexCode == 0x5a: return "6"
  elif hexCode == 0x42: return "7"
  elif hexCode == 0x52: return "8"
  elif hexCode == 0x4a: return "9"
  elif hexCode == 0x16: return "0"
  elif hexCode == 0x40: return "up"
  elif hexCode == 0x19: return "down"
  elif hexCode == 0x07: return "left"
  elif hexCode == 0x09: return "right"
  elif hexCode == 0x15: return "enter_save"
  elif hexCode == 0x0d: return "back"
  elif hexCode == 0x45: return "volMinus"
  elif hexCode == 0x47: return "volPlus"
  elif hexCode == 0x46: return "play_pause"
  elif hexCode == 0x44: return "setup"
  elif hexCode == 0x43: return "stop_mode"
```

```
  elif hexCode == 0x44: return "setup"
  elif hexCode == 0x43: return "stop_mode"
  else: return "NEC remote code error"

def remoteNEC_callback(data, addr, ctrl):
  global ir_current_remote_code
  if data < 0:  # NEC protocol sends repeat codes.
    print('Repeat code.')
  else:
    print('Data {:02x} Addr {:04x} Ctrl {:02x}'.format(data, addr, ctrl))
    ir_current_remote_code = remoteNEC_basicBlack_getButton(data)

classes = (NEC_8, NEC_16)
ir_remote = classes[0](alphabot.pin_IR, remoteNEC_callback)
alphabot.calibrateLineFinder()

def suiveur_de_ligne():
  if alphabot.readUltrasonicDistance() <= 5:
    Contournement()
  else:
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=15)
      alphabot.setMotors(right=0)
    if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveForward(15)
    if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=15)
      alphabot.setMotors(right=5)
    if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=0)
      alphabot.setMotors(right=15)
    if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=5)
      alphabot.setMotors(right=15)
    if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    oled.fill(0)
    oled.show()

def Contournement():
  tourner_gauche_90_C2_B0()
  avancer_10_cm()
  tourner_droite_90_C2_B0()
  avancer_10_cm()
  tourner_droite_90_C2_B0()
  avancer_10_cm()
  tourner_gauche_90_C2_B0()
```

```
84 def tourner_gauche_90_C2_B0():
85   alphabot.setMotors(left=20)
86   alphabot.setMotors(right=-20)
87   utime.sleep_ms(450)
88   alphabot.stop()
89 
90 def tourner_droite_90_C2_B0():
91   alphabot.setMotors(right=20)
92   alphabot.setMotors(left=-20)
93   utime.sleep_ms(450)
94   alphabot.stop()
95 
96 def avancer_10_cm():
97   if alphabot.readUltrasonicDistance() > 10:
98     alphabot.moveForward(20)
99     utime.sleep_ms(400)
100    alphabot.stop()
101  else:
102    alphabot.stop()
103 
104 def reculer_10():
105  alphabot.moveBackward(20)
106  utime.sleep_ms(400)
107  alphabot.stop()
108 
109 utime.sleep(3)
110 
111 while True:
112  utime.sleep_ms(150)
113  gc.collect()
114  if ir_current_remote_code == "enter_save":
115    alphabot.stop()
116  elif ir_current_remote_code == "up":
117    avancer_10_cm()
121    tourner_gauche_90_C2_B0()
122  elif ir_current_remote_code == "right":
123    tourner_droite_90_C2_B0()
124  elif ir_current_remote_code == "play_pause":
125    suiveur_de_ligne()
126  else:
127    suiveur_de_ligne()
```

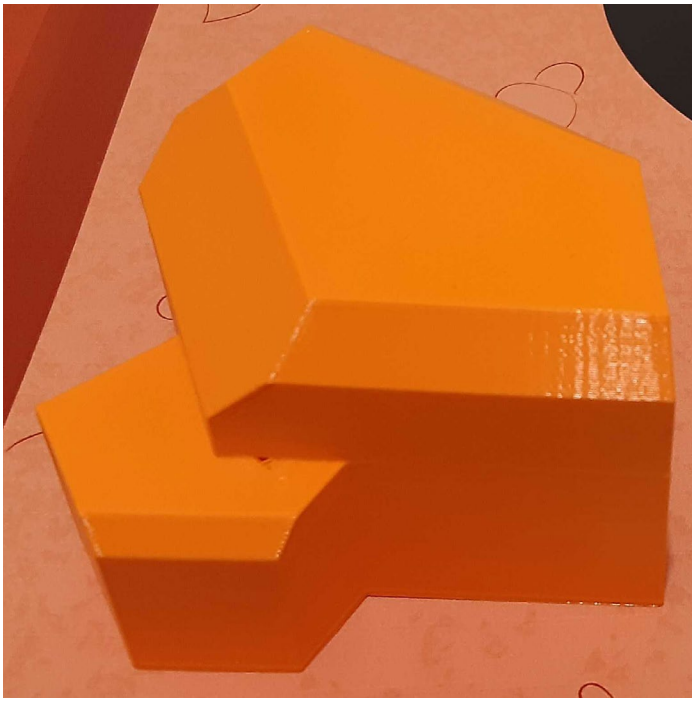
Pour aller plus loin

Afin de compléter la programmation de ce livret il est possible de rajouter les activités suivantes en cours d'année ou sur d'autres niveaux en parallèle.

Réalisation d'objets en 3D :

- Mini cratère qui s'adapte au parcours
- Rocher
- Base Martienne
- Panneau solaire

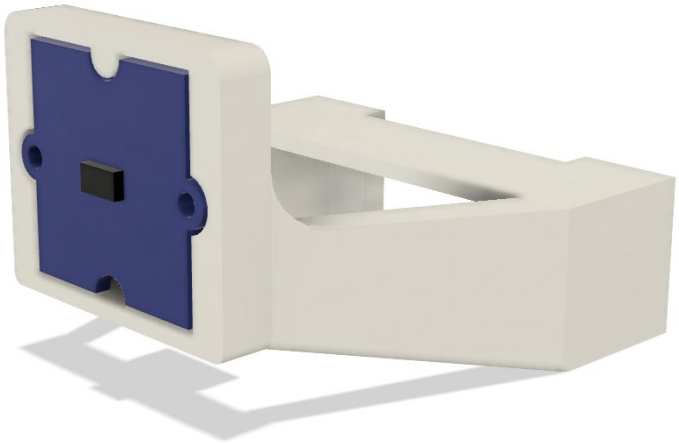
Un projet plus conséquent consisterait à concevoir une base pour accueillir le robot. On pourrait partir de l'idée d'un garage avec une porte automatique, ce qui permettrait d'utiliser le capteur à ultrasons afin de détecter la présence du robot et de commander l'ouverture ou la fermeture du sas d'entrée de la base. Ce projet mettrait en œuvre de la programmation, mais aussi de la conception et de la fabrication avec machine outils, imprimante 3D ou découpe laser.



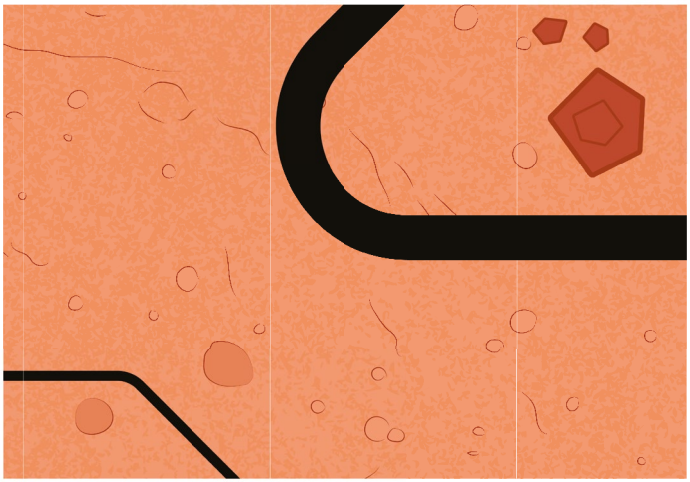
Un exemple de rocher martien imprimé en 3D, qui constitue un obstable idéal pour le capteur de distance !

Pour aller plus loin

Si vous souhaitez remplacer le capteur à ultrasons par le capteur "Time of Flight", voici un support à imprimer en 3D. Retrouvez-le sur le site vittascience.com/learn.

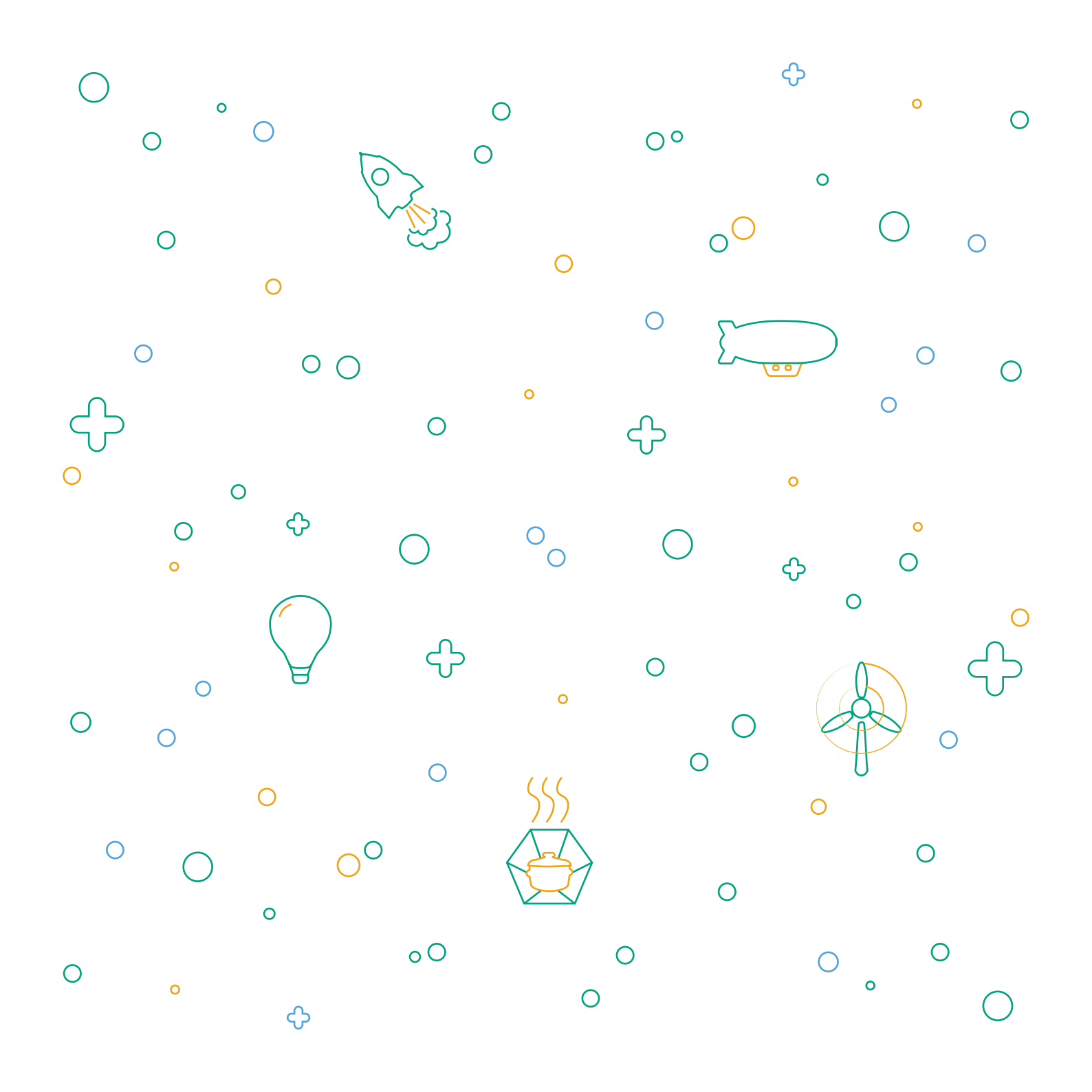


Pour fermer la piste du robot martien, imprimez cette feuille au format A4 et placez-la sur la piste. Retrouvez-la sur le site vittascience.com/learn



Notes

Notes



**Voici le guide de construction du robot martien.
Il détaille chaque étape et chaque atelier nécessaire
à la réalisation de l'expérience. Ce livret n'est pas exhaustif,
l'imagination et les ressources disponibles sur le site
sont là pour aider à approfondir l'expérience !**

Ce livret est susceptible d'évoluer, nous vous invitons à consulter la version disponible en ligne qui est mise à jour régulièrement sur : www.vittascience.com/learn
(chercher "Robot martien" pour trouver le guide d'utilisation)

vitta science | life.augmented

STEM your way
Innovation depends on you

IPCEI on Microelectronics