Exploration avec les robots

Livret enseignant

Ce livret offre des suggestions pédagogiques, que chaque enseignant est libre d'adapter selon sa propre pratique et dans le respect de sa liberté pédagogique. A noter, les prérequis indiqués à la fin du premier temps sont à prendre en compte avant votre venue au Campus Numeria.

Premier temps du parcours – Activités préparatoires en classe

Consigne introductive aux élèves

Dans notre séquence Exploration avec les robots, vous allez piloter un robot explorateur muni d'une caméra pour interagir avec le terrain.

Nous utiliserons une plateforme de programmation en ligne, dotée d'une simulation.

1. Objectifs de la séance

Dans cette séance, vous découvrirez le matériel et le système de programmation qui sera mis à votre disposition lors de votre venue au Campus Numeria (Futuroscope).

2. Le matériel nécessaire

Le robot Maqueen Plus V2 par DFRobot sera le robot explorateur. Il sera simulé sur la plateforme Vittascience.

Deux cartes micro:bit seront nécessaires pour effectuer de la télétransmission.

Un servomoteur permettra de simuler le système de pince du robot.

3. Plan synthétique de la séance

Introduction : Présentation de la mission et découverte du matériel (30 min)

- Objectifs :
  - Rappeler des points clés : algorithme et programmation.
  - Présenter l'outil de programmation qui sera utilisé lors du parcours (Vittascience.fr).
- Matériel : présentation des supports (robot, livret élève, livret ressources) à retrouver sur la page web du Campus Numéria.
- Organisation pédagogique : classe entière.

Opération soutenue par l'État dans le cadre des dispositifs « Innovation dans la forme scolaire » et « Compétences et métiers d'avenir » de France 2030, opérés par la Caisse des dépôts

Activité 1 : Introduction à l'algorithmie (40 minutes)

- Objectif : introduire les notions d'algorithme et d'algorigramme.
- Matériel : livret ressources et livret élève fournis par l'enseignant.
- Organisation pédagogique : individuel.

Activité 2 : Communication à distance entre cartes (60 à 90 minutes)

- Objectifs :
  - Prendre en main la plateforme Vittascience.fr.
  - Apprendre à utiliser les instructions de communication.
  - Réaliser un programme simple à partir d'un algorigramme.
- Matériel :
  - Livret ressources et livret élève.
  - Exercice de programmation pratique sur le livret ressources.
  - 2 cartes micro:bit par groupe et un ordinateur.
- Organisation pédagogique : travail en groupe de 2 à 4 élèves.

Activité 3 : Contrôle du robot à grande distance (40 minutes)

- Objectifs :
  - Utiliser un programme de contrôle à grande distance du robot.
  - Simuler un programme sur Vittascience.fr.
  - Exercice d'application pour comprendre l'impact de la distance.
  - Echanger et argumenter.
- Organisation pédagogique :
  - Livret ressources et livret élève.
  - Exercices pratiques sur le livret ressources.
  - Travail en groupes de 2 élèves par ordinateur.
  - Entraide entre les élèves.
  - Echanges argumentés en classe entière.

Activité 4 : Contrôler un servomoteur (30 minutes)

- Objectifs :
  - Réaliser un programme simple à partir d'un algorigramme.
  - Simuler un programme sur Vittascience.fr
- Organisation pédagogique :
  - Travail en groupes de 2 élèves par ordinateur.
  - Entraide entre les élèves.

Conclusion (10 minutes)

- Objectifs :
  - Revenir sur les objectifs de la séquence et les mettre en lien avec le deuxième temps.
  - Faire le bilan des apprentissages.
- Organisation pédagogique : discussion collective.

Prérequis pour la sortie au Campus Numeria

- Savoir utiliser la plateforme de Vittascience
- Savoir ajouter/modifier les blocs de commandes pour piloter le robot Maqueen Plus V2
- Apporter les livrets élèves

Introduction

Mise en situation

En 1996, la NASA a envoyé un robot explorateur sur Mars. Dénommé Sojourner, il est le premier véhicule à roues à se déplacer sur une autre planète que la Terre.

Nous allons chercher et comprendre quels sont les défis à la réalisation d'une telle avancée et l'importance de la robotique dans ce contexte.

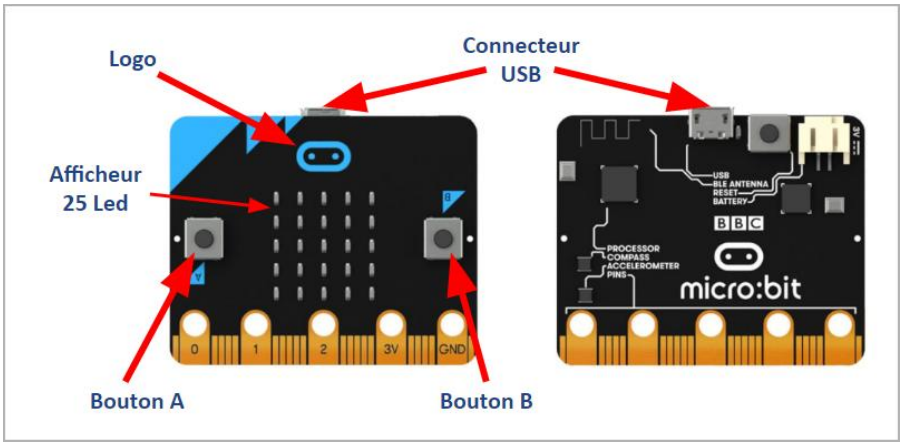
Le saviez-vous ?
Pour nommer ses robots, la NASA lance un concours auprès de la jeunesse américaine.

Alors que Valérie Ambroise n'a que 12 ans, elle remporte le concours et donne le nom de Sojourner.

Sojourner signifie voyageur ou étranger, mais Valérie a choisi ce nom en hommage à Sojourner Truth, une afro-américaine militante pour le droit des femmes, quelques que soient leurs origines, et pour l'abolitionnisme de l'esclavage au XIX siècle.

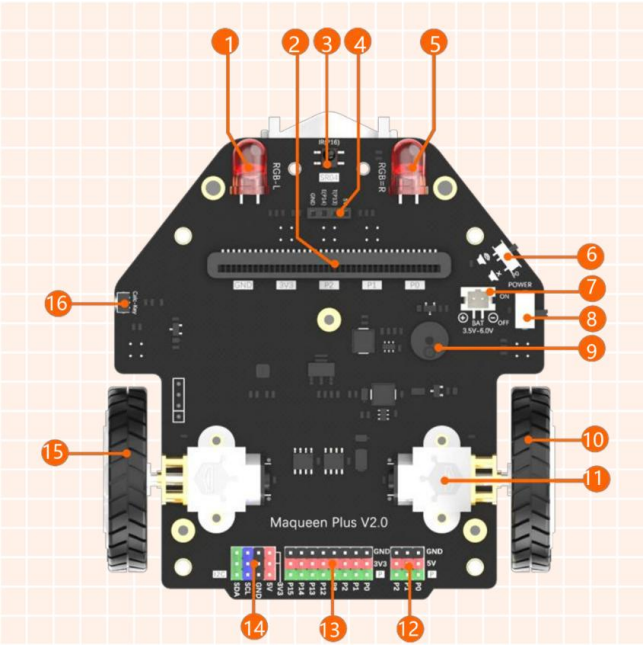
Présentation du matériel

Notre robot est pilotable grâce à une carte électronique munie d'un microprocesseur et de son programme qui permet le pilotage du moteur de chaque roue. Nous allons utiliser une carte micro:bit de Microbit.org.



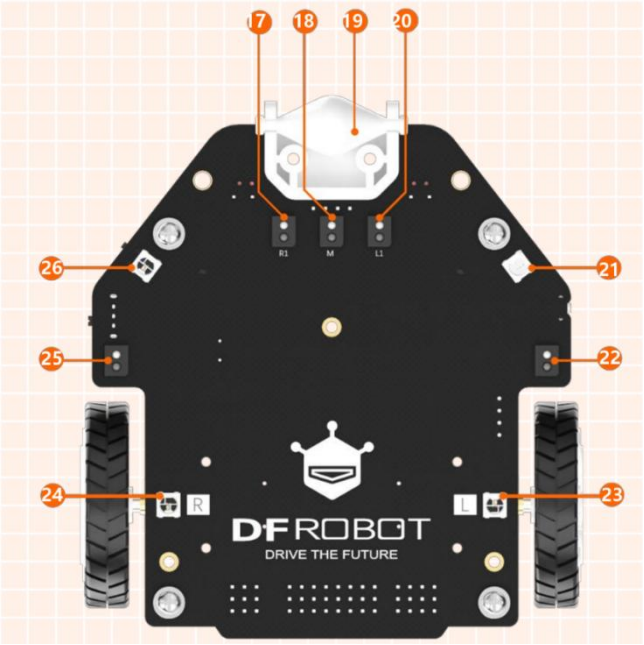
Découverte du robot

Vue du dessus



1 – LED Gauche
2 – Port micro:bit
3 – Récepteur infrarouge
4 – Port capteur à ultrason
5 – LED Droite
6 – Interrupteur du buzzer
7 – Port d'alimentation
8 – Interrupteur d'alimentation
9 – Buzzer
10 – Roue droite
11 – Moteur
12 – Ports pour servomoteur
13 – Ports d'entrée/sortie
14 – Port I2C (permet la connexion de périphériques)
15 – Roue gauche
16 – Bouton de calibration des capteurs de ligne

Vue du dessous



17 – Capteur de ligne « droit »
18 – Capteur de ligne « milieu »
19 – Roue d'appui
20 – Capteur de ligne « gauche »
21 – LED RGB 0
22 – Capteur de ligne « arrière gauche »
23 – LED RGB 1
24 – LED RGB 2
25 – Capteur de ligne « arrière droit »
26 – LED RGB 3

Sur le robot, on va trouver des capteurs qui permettent de recevoir des informations, et des actionneurs qui réalisent l'action.

Capteur : composant qui renvoie une information sur une grandeur physique.

Actionneur : composant qui réalise une commande envoyée par le microcontrôleur.

Consigne : entourer les noms des capteurs du robot Maqueen Plus V2, et souligner les noms des actionneurs dans le livret élève.

Note à l'enseignant

Ouverture possible NSI, STI2D : déterminer le type de composant présent. Les élèves devront chercher (chez eux) les spécificités techniques du composant servant à mesurer le champ magnétique. (LSM303AGR Ultracompact high-performance eCompass module: ultralow-power 3-axis accelerometer and 3-axis magnetometer)

Activité 1 – Introduction à l'algorithmie

Un **algorithme** est une suite d'instructions ou d'étapes à suivre pour résoudre un problème ou faire fonctionner quelque chose.

Exemple :

Algorithme de déplacement automatique d'un robot qui peut rencontrer un obstacle

Si le robot ne détecte pas d'obstacle à moins de 2 cm, celui-ci doit continuer d'avancer, sinon le robot et son programme s'arrêtent.

Si on traduit la phrase précédente en langage algorithmique, cela donne :

Obstacle à moins de 2 cm ? → Oui : le robot s'arrête + fin du programme → Non : le robot avance

Un **algorigramme** (ou organigramme algorithmique) est une représentation schématique et codifiée d'un algorithme.

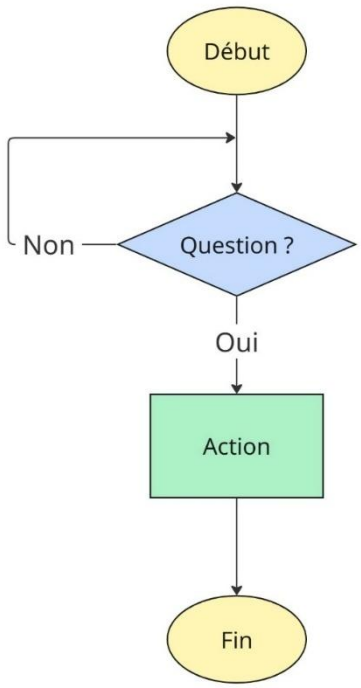
Exemple :

Un **algorigramme** débute toujours par un « **début** », représenté dans une forme arrondie.

Le **losange** représente une question dont la réponse possible est uniquement « **oui** » ou « **non** ». Il fait référence aux **capteurs** du robot.

Le **rectangle** symbolise l'action à réaliser. Il correspond à l'**actionneur**.

Un **algorigramme** se termine le plus souvent par une « **fin** », représentée dans une forme arrondie, sauf dans le cas d'une boucle infinie.

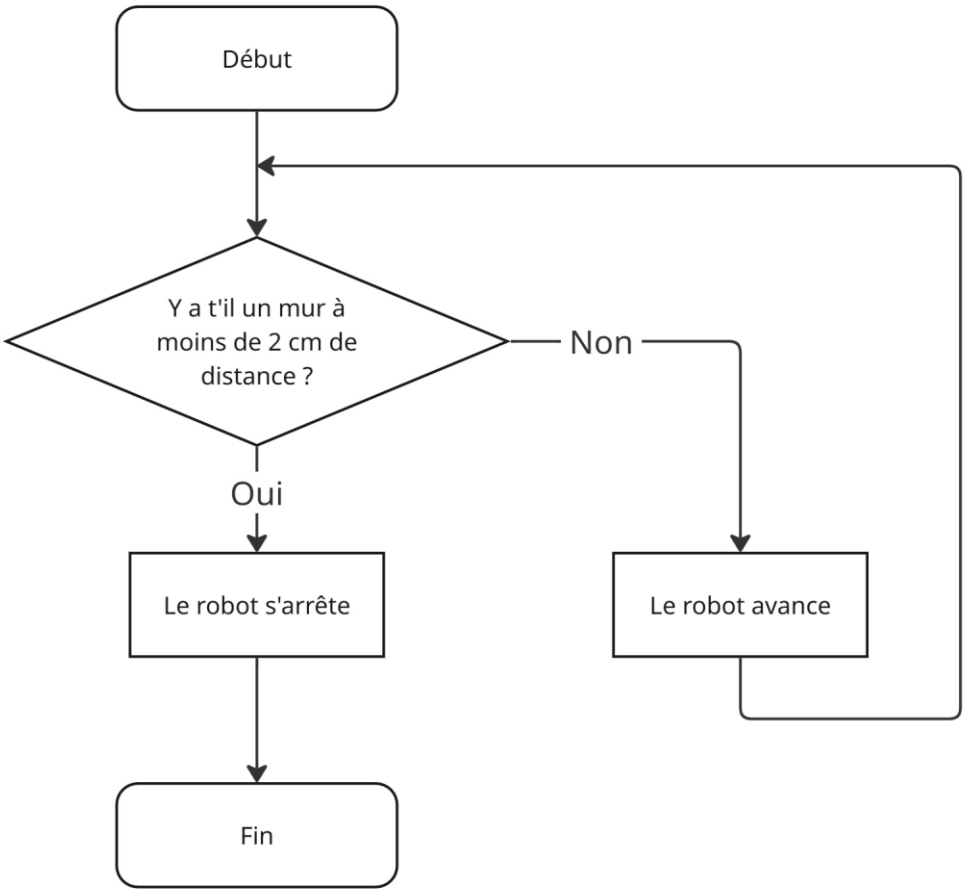


Note à l'enseignant

Dans ce parcours tous les algorigrammes ont été conçus avec une alternative pour chaque question avec « oui » verticalement et « non » horizontalement.

Dans un souci de clarté l'algorigramme est lu verticalement vers le bas.

Algorigramme de déplacement automatique d'un robot qui peut rencontrer un obstacle à moins de 2 cm.



A toi de jouer !

Voici la description du comportement d'un robot :

Si le robot ne détecte pas d'obstacle à moins de 2 cm, celui-ci doit continuer d'avancer sinon le robot pivote à gauche puis avance de nouveau.

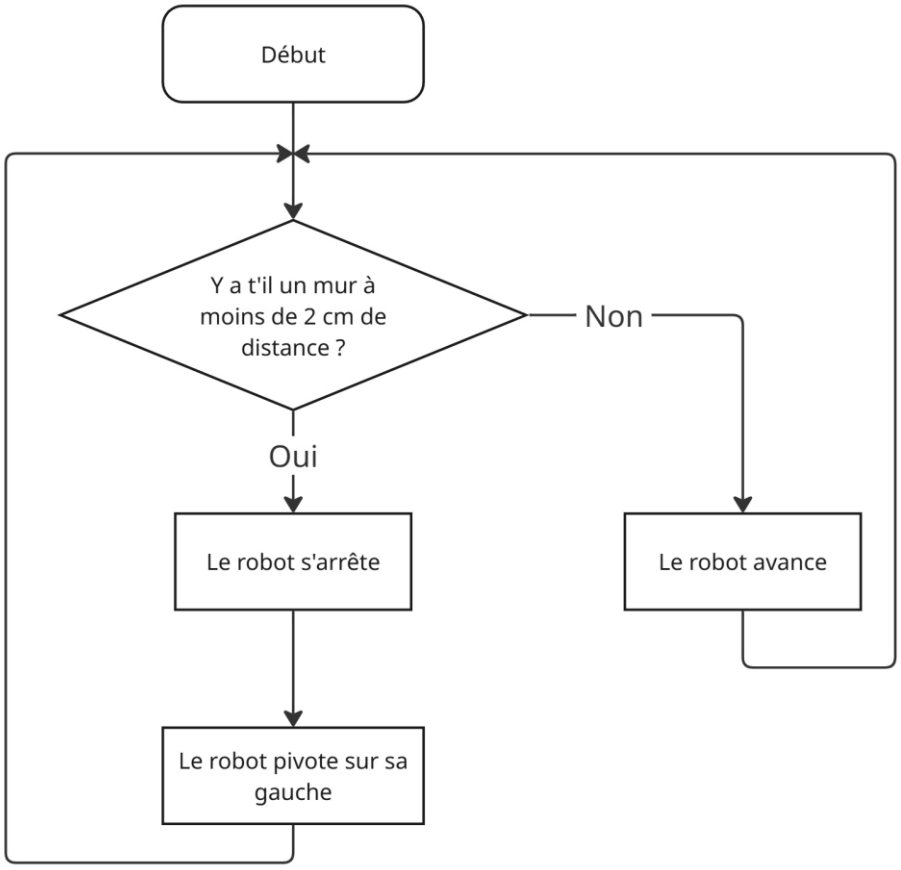
Consignes :

- Traduire cette phrase en langage algorithmique dans le livret élève.
- Construire l'algorigramme associé dans le livret élève.

Note à l'enseignant

Une activité similaire est proposée lors du troisième temps pour évaluer la compréhension de la notion d'algorithme.

Correction possible :

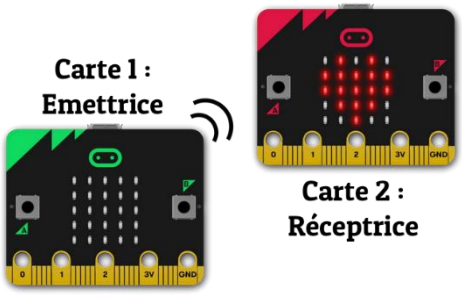


Activité 2 – Communication à distance entre cartes

Pour pouvoir échanger des données entre la Terre et le robot sur Mars, il faut réussir à transmettre des informations à distance. Celles-ci peuvent être des commandes de déplacements, des images, du son…

Le but de l'activité est de programmer une carte « émettrice » et une carte « réceptrice » tout en respectant un cahier des charges.

Cette programmation se fera en Python.



Note à l'enseignant

L'interface de programmation que nous utiliserons sera celle de Vittascience : https://fr.vittascience.com/microbit/

En fonction du niveau des élèves et de la discipline, vous pourrez laisser les élèves écrire le code, le compléter ou le donner entièrement.

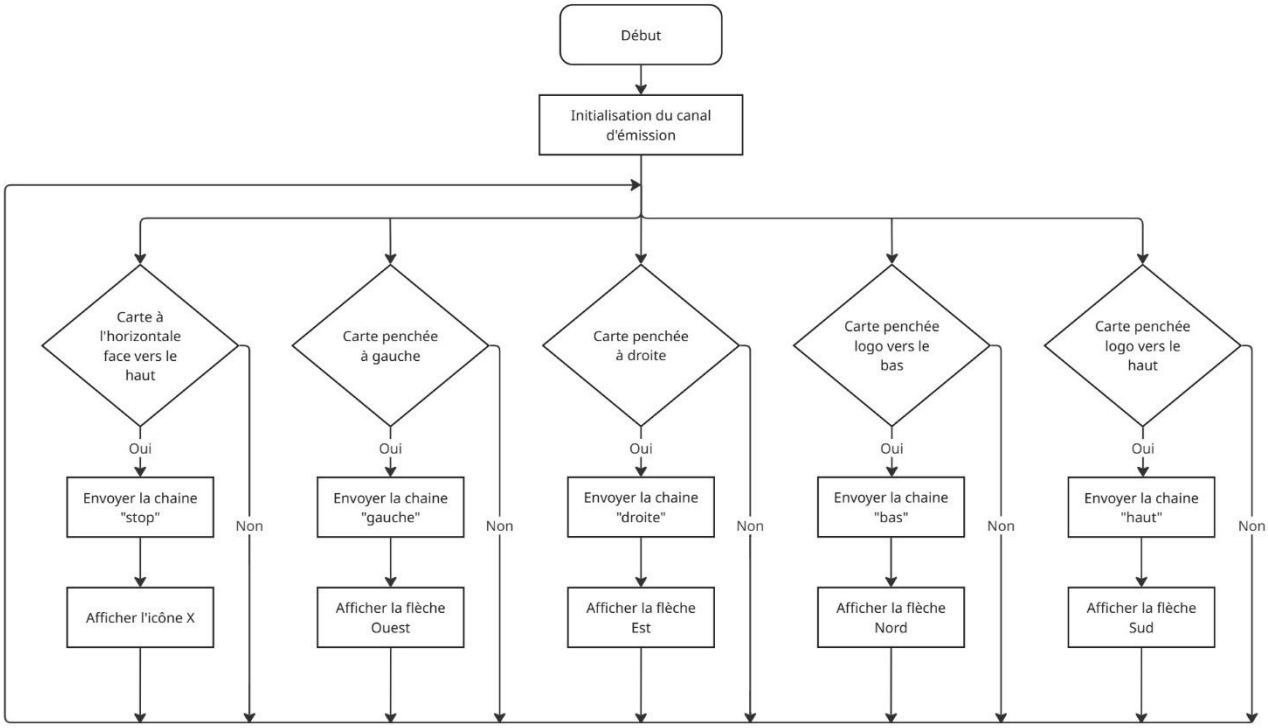
Attention, si en général la boucle infinie est interdite en programmation celle-ci peut être judicieuse pour un système embarqué. On commencera donc par importer les bibliothèques et une boucle while TRUE :

Document 1 – Code utile

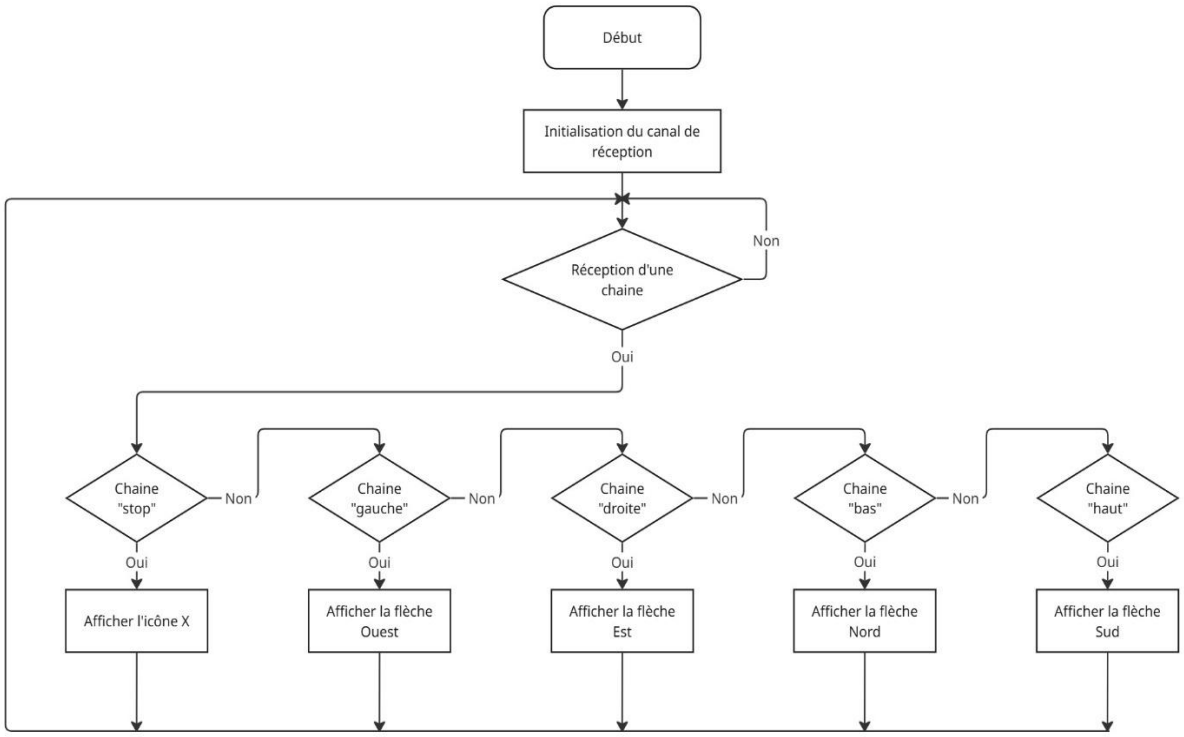
Code	Description
from microbit import *	Importe la bibliothèque microbit. Doit obligatoirement être en tête de programme.
import radio	Importe la bibliothèque radio. Doit obligatoirement être en tête de programme.
radio.config(group=1)	Définit le canal de communication et ses paramètres. A ajuster dans le cadre de plusieurs groupes d'élèves (group=xx par émetteur/récepteur).
radio.on()	Active le module radio.
radio.send()	Envoie un message sur le canal.
radio.receive()	Tente de recevoir un message et le stocke
display.show(Image.NO)	Permet d'afficher une croix sur la matrice de LED de la microbit.
display.show(Image.ARROW_N)	Permet l'affichage d'une flèche spécifique : • N : Nord • S : Sud • W : Ouest • E : Est
accelerometer.current_gesture()	Renvoie une chaîne de caractères indiquant le geste effectué : • shake : secouer • up : logo vers le haut • down : logo vers le bas • left : bascule de la carte vers la gauche • right : bascule de la carte vers la droite • face up : carte à l'horizontale face vers le haut

Plus d'informations sur la librairie micro:bit ici.

Document 2 - Algorigramme de la carte « Emettrice »



Document 3 - Algorigramme de la carte « Réceptrice »



Consignes :

Chaque groupe aura un canal dédié définit par l'enseignant pour éviter les conflits de communication.

A partir des documents précédents, vous allez programmer les deux cartes.

Pour cela, ouvrir le mode multi en cliquant sur le lien : Mode Multi

⚠️ Nous vous conseillons d'utiliser un navigateur qui accepte la communication avec les cartes micro:bit.

- Dans la partie gauche, il faut programmer la carte « Emettrice ».
- Dans la partie droite, il faut programmer la carte « Réceptrice ».
- Connecter la carte « émettrice » à l'ordinateur à l'aide du câble USB
- Cliquer sur 

dans la partie gauche.

- Sélectionner "BBC micro:bit CMSIS-DAP" puis cliquer sur "se connecter".
- Déconnecter la carte.
- Connecter la carte « réceptrice » à l'ordinateur à l'aide du câble USB
- Cliquer sur 

dans la partie droite.
- Sélectionner "BBC micro:bit CMSIS-DAP" puis cliquer sur "se connecter".
- Alimenter les deux cartes.

Observer le comportement.

Note à l'enseignant

Pensez à préciser à chaque groupe quel canal lui est dédié.

Il pourrait être nécessaire de préciser que le bouton < / > permet de voir uniquement le « mode bloc ».

Note à l'enseignant

Correction du programme de la carte émettrice

```
from microbit import * # Importe toutes les fonctionnalités de la bibliothèque micro:bit
import radio          # Importe le module radio pour la communication

radio.on() # Active le module radio

radio.config(group = 1) # Configure la radio pour utiliser le canal 1

while True: # Démarre une boucle infinie qui s'exécute continuellement
  gesture = accelerometer.current_gesture() # Récupère le geste actuel et le stocke dans une
variable

  if gesture == 'face up': # Si le micro:bit est posé à plat, face vers le haut
    display.show(Image.NO) # Affiche une croix
    radio.send('stop')     # Envoie le message 'stop'

  elif gesture == 'left':  # Sinon, si le micro:bit est penché à gauche
    display.show(Image.ARROW_W) # Affiche une flèche vers la gauche de la carte
    radio.send('gauche')   # Envoie le message 'gauche'

  elif gesture == 'right': # Sinon, si le micro:bit est penché à droite
    display.show(Image.ARROW_E) # Affiche une flèche vers la droite de la carte
    radio.send('droite')   # Envoie le message 'droite'

  elif gesture == 'up': # Sinon, si le micro:bit est penché vers le haut (logo vers le haut)
    display.show(Image.ARROW_S) # Affiche une flèche vers le bas de la carte
    radio.send('haut')     # Envoie le message 'haut'

  elif gesture == 'down': # Sinon, si le micro:bit est penché vers le bas (logo vers le bas)
    display.show(Image.ARROW_N) # Affiche une flèche vers le haut de la carte
    radio.send('bas')      # Envoie le message ‘bas'
```

Correction du programme de la carte réceptrice

```
from microbit import * # Importe toutes les fonctionnalités de la bibliothèque micro:bit
import radio          # Importe le module radio pour la communication

radio.on() # Active le module radio

radio.config(group = 1) # Configure les paramètres radio détaillés

while True: # Démarre une boucle infinie qui s'exécute continuellement
  stringData = radio.receive() # Tente de recevoir un message et le stocke

  if stringData: # Vérifie si un message a bien été reçu

    if stringData == 'stop': # Si le message est 'stop'
      display.show(Image.NO) # Affiche une croix

    elif stringData == 'gauche': # Sinon, si le message est 'gauche'
      display.show(Image.ARROW_W) # Affiche une flèche vers la gauche de la carte

    elif stringData == 'droite': # Sinon, si le message est 'droite'
      display.show(Image.ARROW_E) # Affiche une flèche vers la droite de la carte

    elif stringData == 'haut': # Sinon, si le message est 'haut'
      display.show(Image.ARROW_S) # Affiche une flèche vers le bas de la carte

    elif stringData == 'bas': # Sinon, si le message est 'bas'
      display.show(Image.ARROW_N) # Affiche une flèche vers le haut de la carte
```

Activité 3 – Contrôle du robot à grande distance

L'objectif de l'activité est de simuler le comportement d'un robot contrôlé à grande distance.
Il faudra garder un regard critique sur la situation.

Consignes :

Ouvrir le mode multi en cliquant sur le lien : Mode Multi
Dans la partie gauche, il faut ouvrir le fichier "télécommande Mars" :

- Cliquer sur  ou puis "ouvrir projet"
- Puis taper "télécommande Mars" dans la barre de recherche et dérouler les "projets partagés"

Dans la partie droite, il faut ouvrir le fichier "Robot Commandé" :

- Cliquer sur  ou puis "ouvrir projet"
- Puis taper "Robot Commandé" dans la barre de recherche et dérouler les "projets partagés"

Ensuite, simuler le comportement des deux cartes en **cliquant** sur les deux boutons  .

⚠️ Un rafraichissement de la page peut être nécessaire.

Questions : (répondre dans le livret élève)

1. Quelle difficulté rencontrez-vous lors de la simulation ?
2. Pourquoi a-t-on ajouté un délai de 3s avant l'envoi de chaque commande ?
3. Cette durée vous semble-t-elle cohérente avec la situation de contrôle d'un robot sur Mars à partir de la Terre ?

Vérifiez votre hypothèse par le calcul.

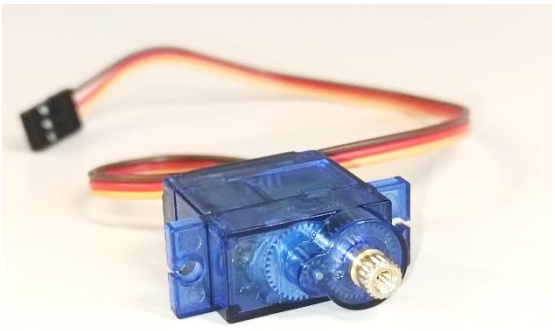
Application : (répondre dans le livret élève)

La distance entre Mars et la Terre selon son orbite est de 55,7 à 401 millions de kilomètres.

Les ondes radio se déplacent à une vitesse proche de celle de la lumière, qui est de 300 000 kilomètres par seconde dans le vide.

1. Quelle est la durée minimale pour qu'un signal radio émis à la surface de la Terre arrive jusqu'à la surface de Mars ? Exprimer le résultat dans une unité adéquate.
2. Quelle est la durée maximale pour qu'un signal radio émis à la surface de la Terre arrive jusqu'à la surface de Mars ? Exprimer le résultat dans une unité adéquate.

Activité 4 – Contrôler un servomoteur



Un robot d'exploration peut être équipé de caméras, de pinces ou d'autres détecteurs qui sont placés sur des bras commandés.

Ces bras sont commandés par des servomoteurs.

Document 1 – Qu'est-ce qu'un servomoteur ?

Un servomoteur est un système motorisé capable d'atteindre des positions prédéterminées extrêmement précises, puis de les maintenir.
Celui que nous utiliserons est un servomoteur à moteur rotatif, la position de celui-ci est déterminée une valeur d'angle en degré comprise entre 0° et 180°.
Il permettra d'activer une pince à l'avant de notre robot.

Quelques usages des servomoteurs :

- Barrière automatique
- Automatisation des processus dans les lignes de production
- Modélisme miniature, comme les petits robots ou avions télécommandés
- Instruments médicaux
- Contrôle de vannes
- Positionnement de miroirs ou d'antennes
- Danse avec les robots

Note à l'enseignant

Attention, le programme python a besoin d'une bibliothèque externe maqueenplusv2.py, en fonction du niveau des élèves et de la discipline, il sera intéressant de l'ouvrir et de détailler les fonctions intégrées)

SNT et NSI 1ère : import de bibliothèque :

```
import maqueenplusv2      # Importe la bibliothèque spécifique au robot MaqueenPlusV2
```

On appelle ensuite les différentes fonctions avec maqueenplusv2.nom_de_la_fonction. (On pourrait aussi simplifier le code en changeant partout et en mettant import maqueenplusv2 as mv2)

Par exemple on commence toujours par :

```
maqueenplusv2.initRobot() # Initialise les composants et capteurs du robot Maqueen
```

Voici la liste des autres fonctions indispensables :

Fonctions Moteurs

- **maqueenplusv2.drive(vitesse)** : Fait avancer les deux moteurs à la même vitesse (0-255).
- **maqueenplusv2.drive(vitesse_gauche, vitesse_droite)** : Fait avancer chaque moteur à une vitesse différente (0-255).
- **maqueenplusv2.backup(vitesse)** : Fait reculer le robot à la vitesse donnée (0-255).
- **maqueenplusv2.spin_left(vitesse)** : Fait pivoter le robot vers la gauche sur lui-même.
- **maqueenplusv2.spin_right(vitesse)** : Fait pivoter le robot vers la droite sur lui-même.
- **maqueenplusv2.stop()** : Arrête complètement les deux moteurs.
- **maqueenplusv2.motors(vitesse_g, direction_g, vitesse_d, direction_d)** : Commande complète pour les moteurs, où la direction peut être maqueenplusv2.FORWARD ou maqueenplusv2.BACKWARD.

Fonctions Capteurs de Ligne

Le robot possède 5 capteurs de ligne, nommés de gauche à droite : L2, L1, M, R1, R2.

- **maqueenplusv2.read_line_sensor(capteur)** : Lit la valeur analogique d'un capteur. Par exemple, maqueenplusv2.read_line_sensor (maqueenplusv2.M) pour le capteur du milieu. La fonction retourne une valeur élevée (proche de 240) si le capteur est sur une ligne noire, et une valeur basse (proche de 70) sur une surface blanche.
- **maqueenplusv2.sensor_on_line(capteur)** : Renvoie True (Vrai) si le capteur spécifié détecte une ligne noire, et False (Faux) sinon. C'est une manière plus simple de savoir si le robot est sur la ligne.

Fonctions Phares (LEDs)

- **maqueenplusv2.headlights(selecteur, etat)** : Allume ou éteint les phares avant.
  - Le sélecteur peut être maqueenplusv2.LEFT, maqueenplusv2.RIGHT, ou maqueenplusv2.BOTH.
  - L'état peut être maqueenplusv2.ON (allumé) ou maqueenplusv2.OFF (éteint).

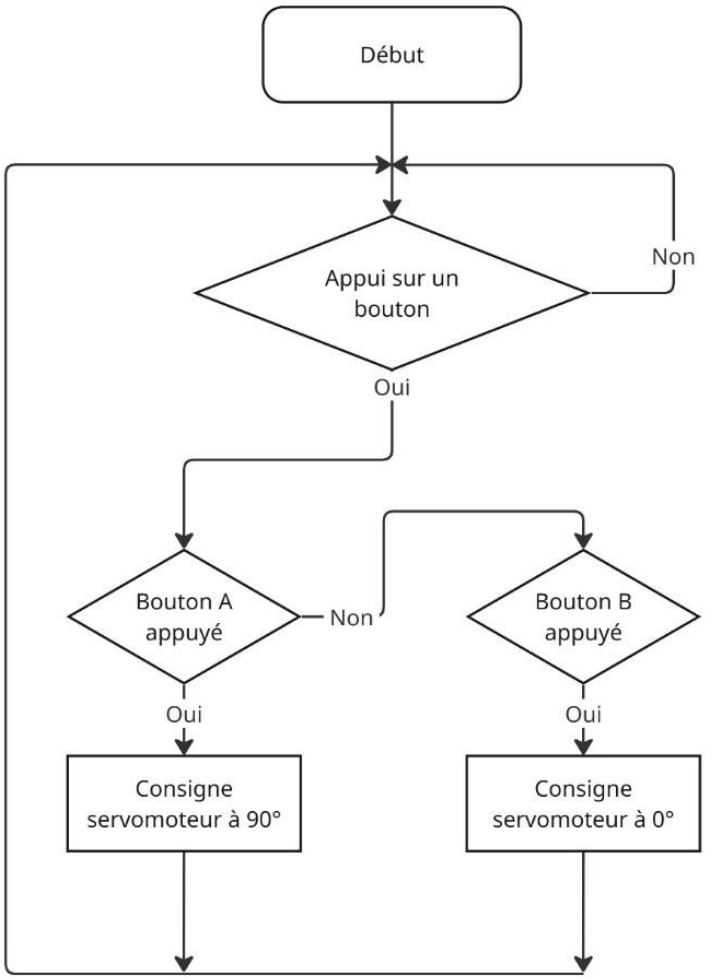
Fonctions Servo-moteurs

- **maqueenplusv2.set_servo_angle(pin, angle)** : Positionne un servo-moteur à un angle précis. Il faut préciser la broche sur laquelle le servo est branché (par exemple pin1) et l'angle désiré (entre 0 et 180 degrés).

Document 2 – Code utile

Code	Description
import maqueenplusv2	Importe la bibliothèque spécifique au robot MaqueenPlusV2
button_a.is_pressed()	Effectue une commande si le bouton A est appuyé.
maqueenplusv2.set_servo_angle(pin1, 90)	Positionne le servo sur la broche P1 à 90 degrés, pinces fermées. L'angle est compris entre 0 et 90 dégrés.

Document 3 – Algorigramme de contrôle du servomoteur



Consignes :

A partir des documents précédents, vous allez programmer la carte qui sera associée à un servomoteur.
Pour cela, ouvrir la page de programmation de Vittascience en cliquant sur le lien : Programmer

Puis cliquer sur **Maqueen Plus**.

Programmer votre carte et vérifier son comportement grâce à la simulation .

Note à l'enseignant

Correction du programme

```
from microbit import *      # Importe toutes les fonctionnalités de la bibliothèque micro:bit
import maqueenplusv2      # Importe la bibliothèque spécifique au robot MaqueenPlusV2
# Le servo-moteur doit être branché sur l'un des ports P0, P1 ou P2 du Maqueen.
# P0 est utilisé pour le son sur le robot MaqueenPlusV2.
# Cet exemple utilise la broche P1 (pin1).
while True: # Crée une boucle infinie qui s'exécute continuellement
  if button_a.is_pressed(): # Si le bouton A est pressé
    maqueenplusv2.set_servo_angle(pin1, 90) # Positionne le servo sur la broche P1 à 90 degrés , pinces fermées
  elif button_b.is_pressed(): # Sinon, si le bouton B est pressé
    maqueenplusv2.set_servo_angle(pin1, 0)  #Positionne le servo sur la broche P1 à 0 degré, pinces ouvertes
```

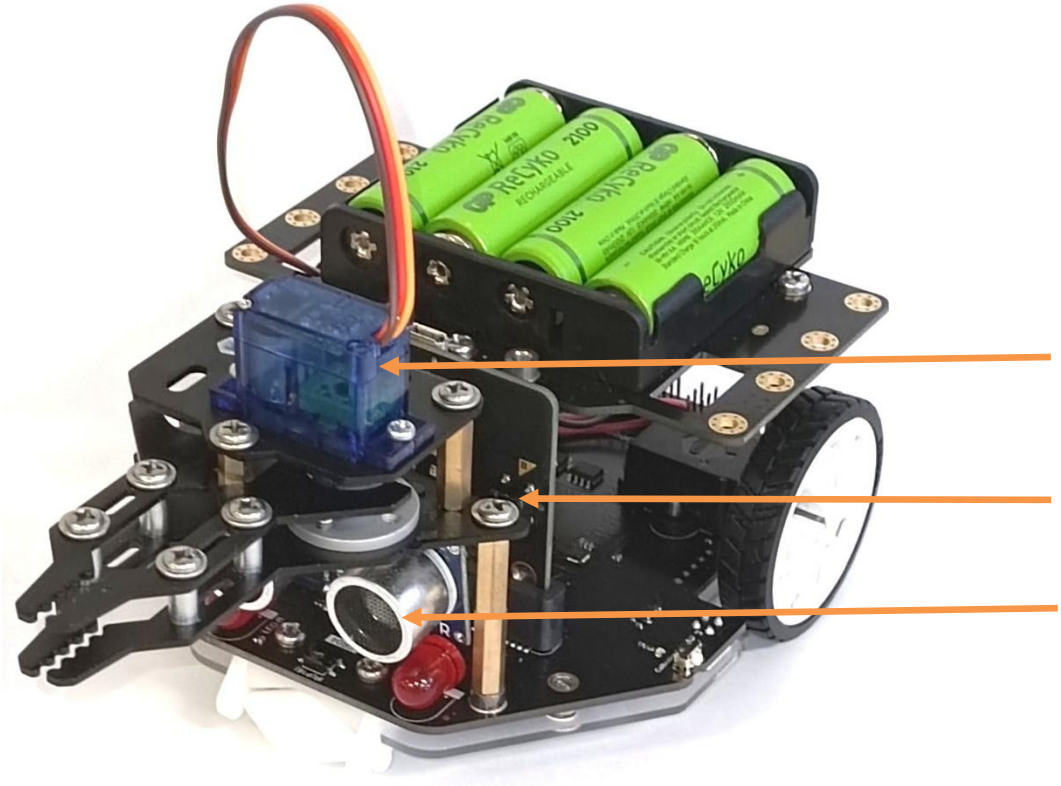
Conclusion

Consigne : (répondre dans le livret élève)

A partir des notions abordées dans les activités 1 à 4, quelles difficultés pourrait-on rencontrer à la réalisation d'une telle mission ?

Annoter la présentation suivante et reconnaitre les capteurs et actionneurs ajoutés sur le robot Maqueen Plus V2 (aide).

Présentation du robot utilisé au Campus Numeria (Futuroscope)



Note à l'enseignant

Dans l'ordre d'apparition : Servomoteur, Carte micro:bit (détecteur de champ magnétique, température, luminosité…), Module ultrason HC SR-04 (mesure de distance).

Deuxième temps du parcours – Activités au Campus Numeria

Activité 1 – Pilotage à grande distance

Lors du premier temps du parcours vous avez simulé une télétransmission de commandes de déplacement à grande distance.

Vous allez tester en réalité cette situation pour mieux appréhender les difficultés.

Consignes :

Ouvrir le mode multi en cliquant sur le lien : Mode Multi

Dans la partie gauche, il faut ouvrir le fichier “télécommande Mars” :

- Cliquer sur  ou puis “ouvrir projet”
- Puis taper “télécommande Mars” dans la barre de recherche et dérouler les “projets partagés”

Dans la partie droite, il faut ouvrir le fichier “Robot Commandé” :

- Cliquer sur  ou puis “ouvrir projet”
- Puis taper “Robot Commandé” dans la barre de recherche et dérouler les “projets partagés”

- Modifier le canal de communication (celui-ci correspond au numéro indiqué sous le robot)

- Connecter la carte « télécommande » à l’ordinateur à l’aide du câble USB
- Cliquer sur  dans la partie gauche.

- Sélectionner “BBC micro:bit CMSIS-DAP” puis cliquer sur “se connecter”.
- Déconnecter la carte et l’alimenter

- Connecter la carte « robot » à l’ordinateur à l’aide du câble USB
- Cliquer sur  dans la partie droite.

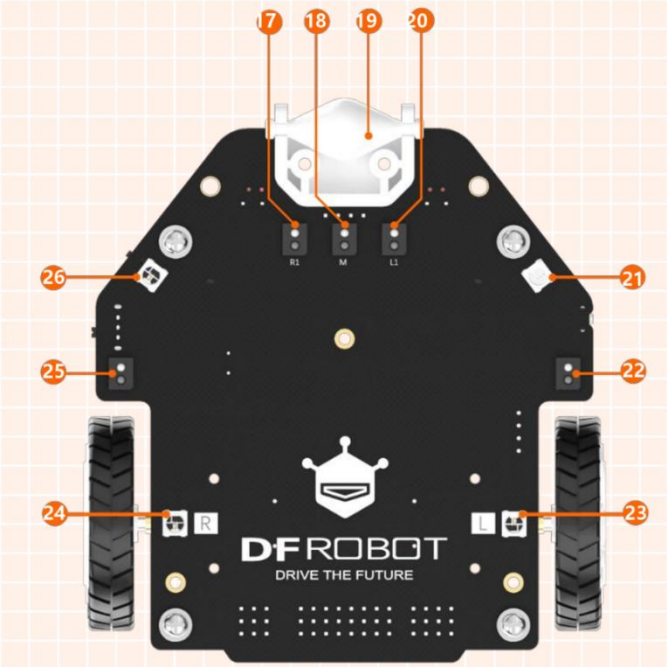
- Sélectionner “BBC micro:bit CMSIS-DAP” puis cliquer sur “se connecter”.
- Déconnecter la carte et l’insérer dans le robot.
- Alimenter le robot

- Déplacer les robots dans l’arène pour former un carré.

Activité 2 - Pilotage automatique par suiveur de ligne

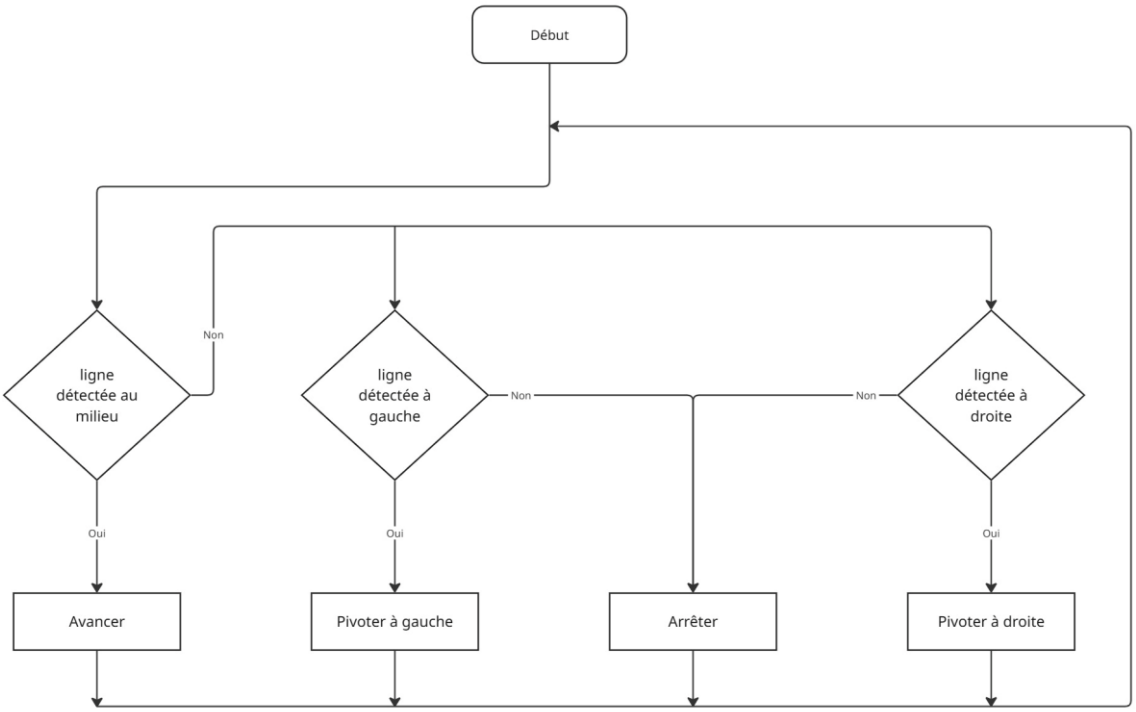
Pour répondre aux difficultés rencontrées, nous allons utiliser un système de guidage autonome pour suivre un parcours défini à l'avance.
Sur Mars cela pourrait se faire par signaux radio.

Document 1 – Présentation du dispositif de suivi de ligne



17 – Capteur de ligne « droit » R1
18 – Capteur de ligne « milieu » M
19 – Roue d'appui
20 – Capteur de ligne « gauche » L1
21 – LED RGB 0
22 – Capteur de ligne « arrière gauche »
23 – LED RGB 1
24 – LED RGB 2
25 – Capteur de ligne « arrière droit »
26 – LED RGB 3

Document 2 - Algorigramme



Consignes :

A partir des documents précédents, vous allez programmer la carte qui sera associée au robot pour que celui-ci se déplace automatiquement sur la ligne noire de la piste.

Pour cela, ouvrir la page de programmation de Vittascience en cliquant sur le lien : Programmer

Puis cliquer sur **Maqueen Plus**.

Ouverture du fichier “Suivi de ligne” :

- Cliquer sur  ou puis “ouvrir projet”
- Puis taper “Suivi de ligne” dans la barre de recherche et dérouler les “projets partagés”
- Compléter le code en remplaçant les « ???? ».
- Téléverser pour tester.

Vous pouvez faire autant d’essais que nécessaire pour vérifier votre programme et régler la vitesse.

Remarque :

- Un bouton nommé « Calibration Key » sur le dessus du robot peut être utilisé pour définir la couleur de la ligne.
- Seuls les capteurs de ligne à l’avant seront utilisés.
- Une vitesse trop importante rend le comportement moins réactif.

Note à l'enseignant

Version attendue avec les instructions données dans le livret.

Programme python

```
from microbit import *      # Importe toutes les fonctionnalités de la bibliothèque micro:bit
import maqueenplusv2      # Importe la bibliothèque spécifique au robot MaqueenPlusV2

""" MaqueenPlusV2 - Suiveur de ligne simple """

maqueenplusv2.initRobot() # Initialise les composants du robot

VITESSE = 40 # Définit une vitesse de base pour faciliter les modifications

while True: # Boucle principale qui s'exécute en continu

  # La logique est une suite de choix exclusifs, on utilise donc if/elif/else.
  # Le robot ne prendra qu'une seule décision à chaque tour de boucle.

  if maqueenplusv2.sensor_on_line(maqueenplusv2.M): # 1. Cas idéal : le capteur du milieu (M) voit la ligne
    maqueenplusv2.drive(VITESSE)                # On avance tout droit

  elif maqueenplusv2.sensor_on_line(maqueenplusv2.L1): # 2. Sinon, si le capteur gauche (L1) voit la ligne, le robot a trop dévié à droite
    maqueenplusv2.spin_left(VITESSE)              # On pivote à gauche pour revenir sur la ligne

  elif maqueenplusv2.sensor_on_line(maqueenplusv2.R1): # 3. Sinon, si le capteur droit (R1) voit la ligne, le robot a trop dévié à gauche
    maqueenplusv2.spin_right(VITESSE)              # On pivote à droite pour revenir sur la ligne

  else: # 4. Cas par défaut : si aucun des capteurs centraux ne voit la ligne
    maqueenplusv2.stop() # Le robot est probablement perdu, on s'arrête par sécurité
```

Activité 3 – Fonctions

Mission 1 : Tri manuel

Lors de l'exploration, un intérêt particulier est porté sur la constitution de la matière (présence d'eau, roche calcaire…).

Nous allons coder une carte pour comprendre le fonctionnement de base.

Deux types d'objets existent :

- Ceux qui ont un intérêt (qui contiennent un aimant) ;
- Ceux qui ne suscite aucun intérêt (qui ne contiennent pas d'aimant).

Document 1 – Code spécifique

On pourra tester la pince avant de commencer à programmer

```
from microbit import *      # Importe les fonctionnalités de base
import maqueenplusv2      # Importe la bibliothèque du robot
# --- Configuration Matérielle ---
# SERVO de la pince sur le port P1 ou SP2 (PAS P0)
SERVO_PIN = pin1 #ici branché sur pin1 sur le robot
while True:
    for i in range(0,40,10): # de l'angle 0 à 90 par pas de 10
        maqueenplusv2.set_servo_angle(SERVO_PIN, i) # <-- ICI ON OUVRE LA PINCE AU DÉPART
        sleep(1000)
        display.scroll(i) # on affiche l'angle
```

Puis ensuite on peut mesurer le champ magnétique ambiant avec

```
get_magnetic_strength()
```

Pour mesurer la distance, on utilise le capteur ultra-son dont on propose ici une définition de fonction

```
def read_distance_cm():
    """ Mesure et retourne la distance en cm du capteur à ultrasons. """
    ULTRASONIC_TRIG.write_digital(0); sleep_us(2)
    ULTRASONIC_TRIG.write_digital(1); sleep_us(10)
    ULTRASONIC_TRIG.write_digital(0)
    duration = time_pulse_us(ULTRASONIC_ECHO, 1, 30000)
    return duration *0.0343/2
```

Donc on utilisera les opérateurs booléens, ici AND, pour vérifier que deux conditions sont vraies

Remarque :

Le temps mesuré est la durée aller-retour, donc la distance totale parcourue par le son est :

Distance totale = vitesse du son × temps

La vitesse du son dans l'air est d'environ 343 m/s, soit 0.0343 cm/µs.

Donc :

Distance totale (aller-retour) = 0.0343 (cm/µs) × durée (µs)

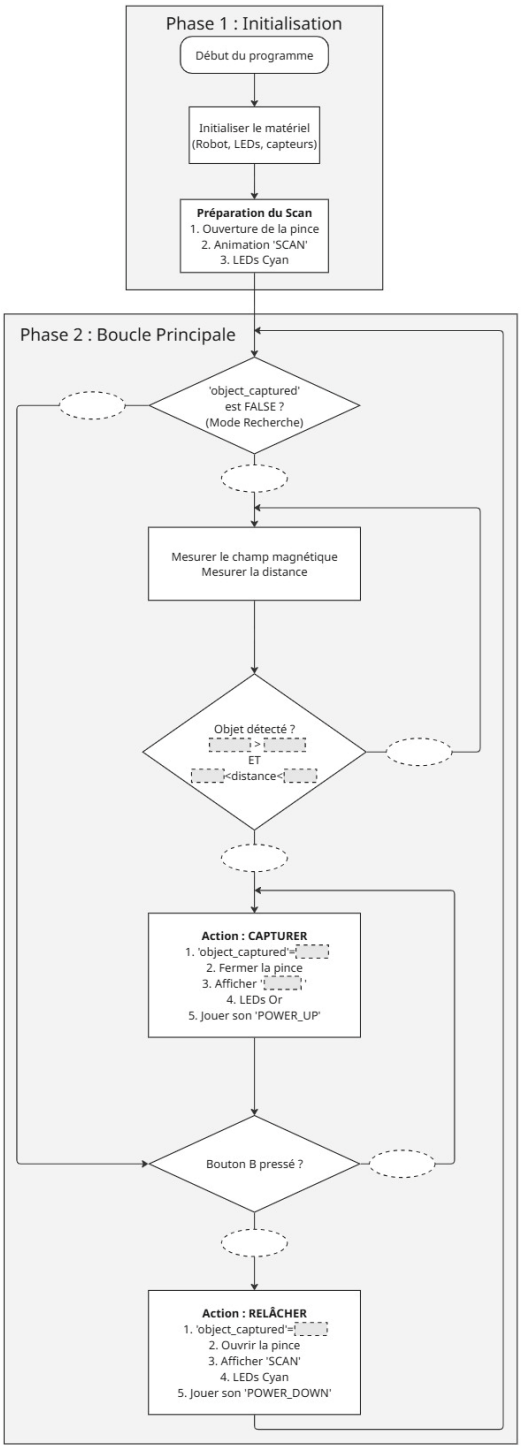
Mais on veut la distance allée simple (du capteur à l'objet), donc on divise par 2.

Consignes :

Ouverture du fichier “Tri par le robot” :

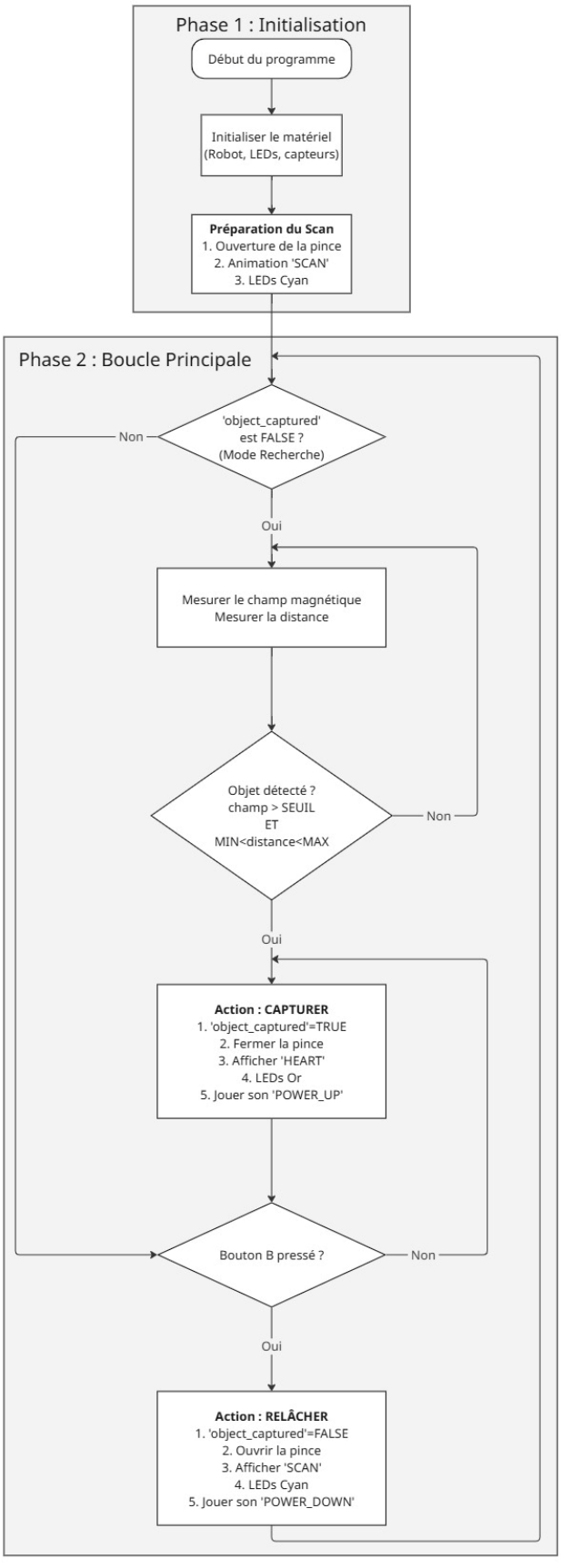
- Cliquer sur  ou puis “ouvrir projet”
- Puis taper “Tri par le robot” dans la barre de recherche et dérouler les “projets partagés”
- Téléverser pour tester.

Dans le livret élève, compléter les zones en pointillés de l’algorigramme suivant à partir du code et de vos observations.



Note à l'enseignant

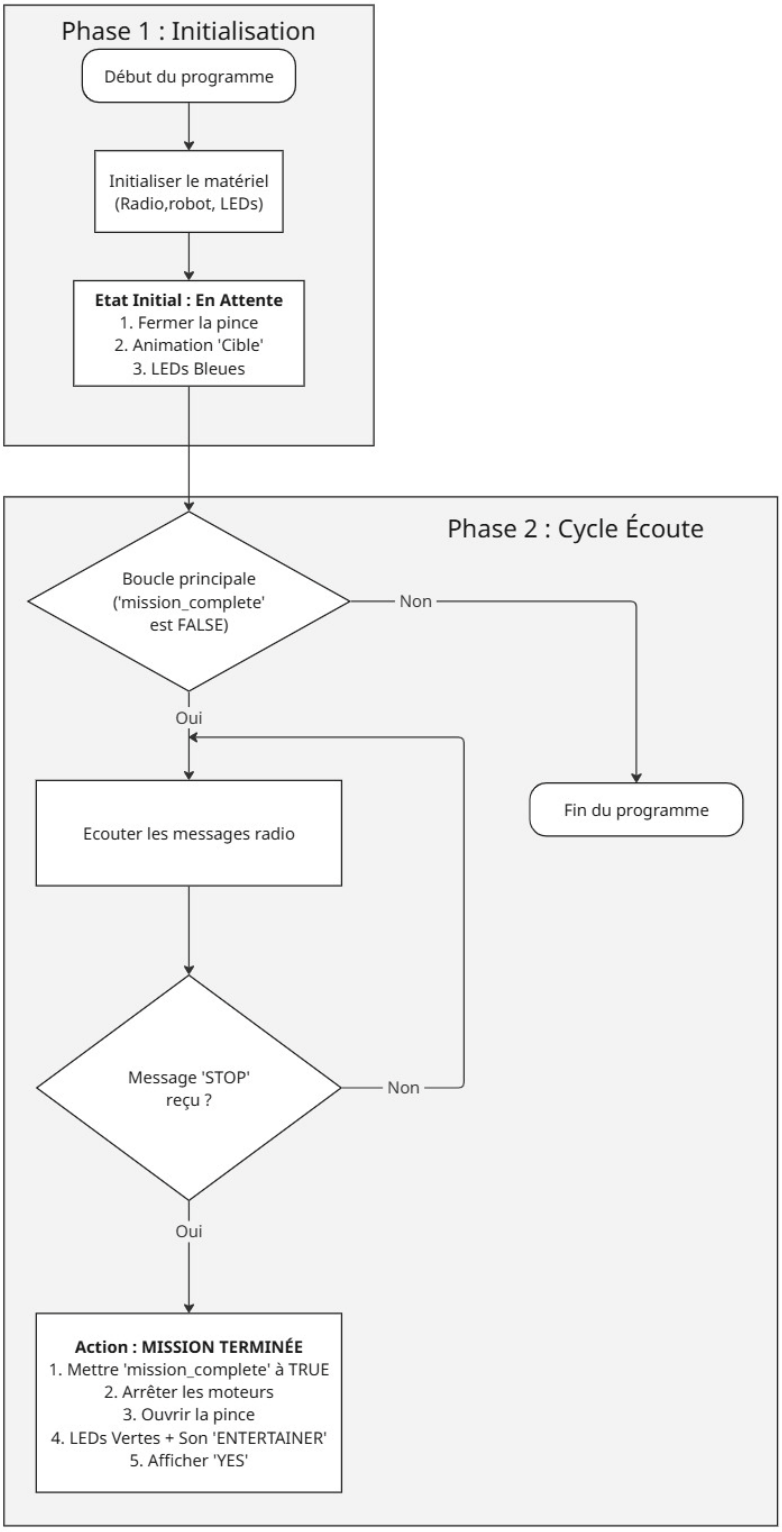
Correction de l'algorigramme



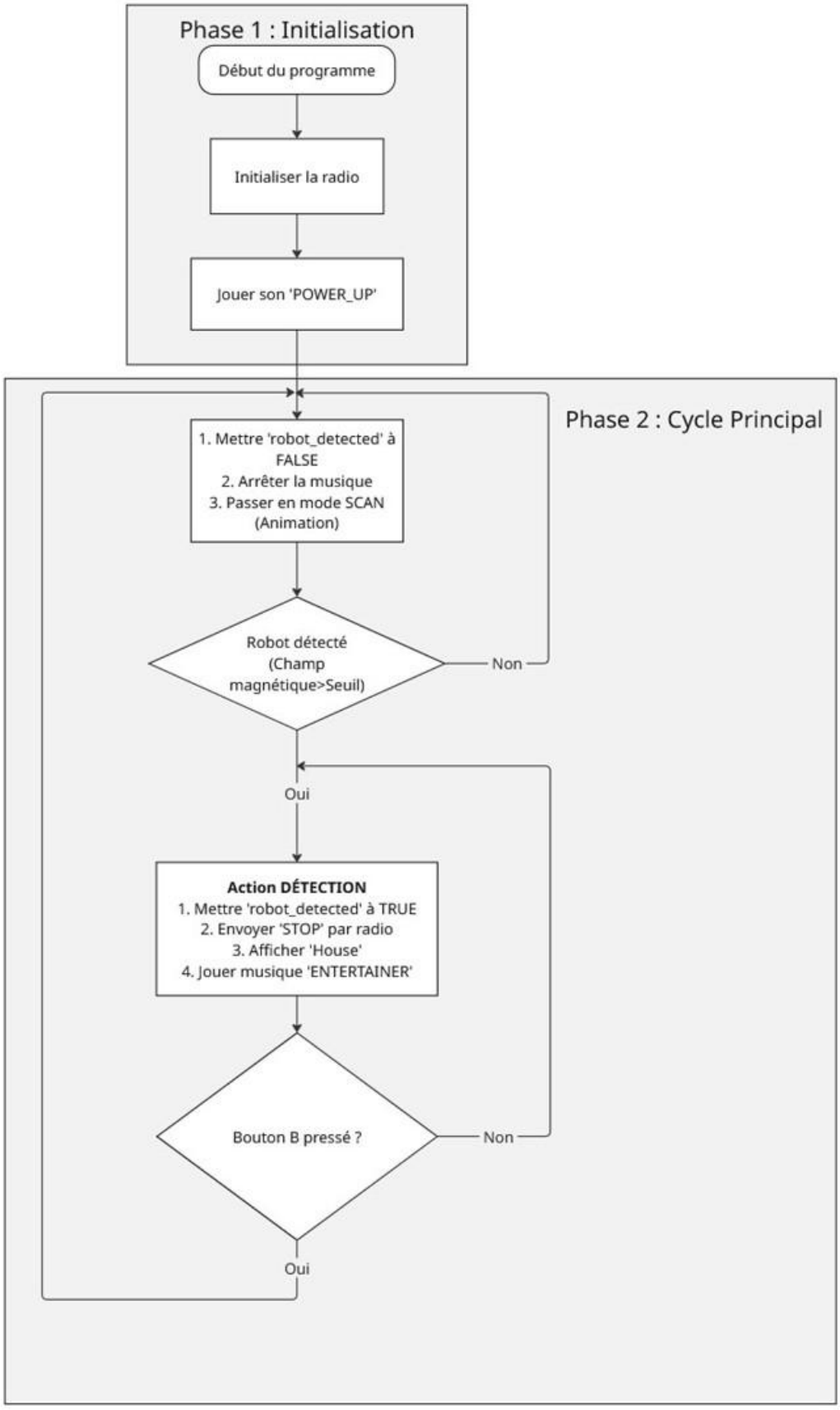
Mission 2 : Préparation du bagage pour le retour

Lorsque le robot retourne à la base en transportant un objet contenant un aimant, celui-ci continue son chemin jusqu'à ce que la base le prévienne de sa proximité.

Document 1 – Algorigramme « robot »



Document 2 – Algorigramme « base »



Consignes :

Ouvrir le mode multi en cliquant sur le lien : Mode Multi

Dans la partie gauche, il faut ouvrir le fichier “base Mars Python” :

- Cliquer sur  ou puis “ouvrir projet”
- Puis taper “base Mars Python” dans la barre de recherche et dérouler les “projets partagés”

Dans la partie droite, il faut ouvrir le fichier “robot Mars Python” :

- Cliquer sur  ou puis “ouvrir projet”
- Puis taper “robot Mars Python” dans la barre de recherche et dérouler les “projets partagés”

Note à l’enseignant

Correction de la carte « base »

```
from microbit import *
import radio
import music

"""
PROJET : BASE DE DÉPÔT INTELLIGENTE
- Se calibre au démarrage pour connaître le champ magnétique ambiant.
- Détecte l'approche d'un aimant (le robot qui arrive).
- Envoie un signal radio unique pour ordonner au robot de s'arrêter.
- Fournit un feedback visuel et sonore riche.
"""

# ============================================================================
# --- 1. INITIALISATION ET CONFIGURATION ---
# ============================================================================
radio.on()
radio.config(channel=7, power=6) # Configuration du canal radio

# --- Constantes ---
MAGNETIC_THRESHOLD = 100000 # Sensibilité de la détection (plus élevée car l'aimant est plus loin)

# --- Variables d'état ---
baseline_strength = 0
calibrated = False
robot_detected = False # Pour n'envoyer le signal qu'une seule fois
```

Modifier le canal de communication (celui-ci correspond au numéro indiqué sous le robot)

- Compléter les codes en remplaçant les « ???? ».
- Téléverser pour tester.

Note à l'enseignant

Correction de la carte « base » (suite)

```
#
================================================================
=============
# --- 2. BOUCLE PRINCIPALE DE SURVEILLANCE ---
#
================================================================
=============
display.show(Image.ALL_CLOCKS, delay=100, loop=True, wait=False) # Animation de scan

while True:
    # --- PHASE DE SCAN (uniquement si le robot n'a pas encore été détecté) ---
    if not robot_detected:
        difference = abs(compass.get_field_strength() - baseline_strength)

        if difference > MAGNETIC_THRESHOLD:
            # --- ROBOT DÉTECTÉ ! ---
            robot_detected = True # On change l'état pour ne plus scanner

            # Action 1: Envoyer le signal
            radio.send('STOP')

            # Action 2: Feedback visuel et sonore
            display.show(Image.HOUSE) # Affiche une maison : "Bienvenue à la base"
            music.play(music.ENTERTAINER, wait=False) # Joue une longue mélodie de bienvenue

    # --- PHASE DE RÉINITIALISATION (pour un nouveau cycle) ---
    if button_b.is_pressed() and robot_detected:
        robot_detected = False # On réinitialise l'état
        music.stop() # Arrête la musique en cours
        display.show(Image.ALL_CLOCKS, delay=100, loop=True, wait=False) # Relance le scan

    sleep(100)
```

Note à l'enseignant

Correction de la carte « robot »

```
from microbit import *
import radio
import music
import maqueenplusv2
import neopixel

"""
PROJET : ROBOT EN RETOUR DE MISSION
- Démarre avec la pince fermée, en simulant un déplacement.
- Écoute les ordres de la base via radio.
- À la réception du signal 'STOP', il s'arrête, dépose l'objet et confirme la fin de mission.
"""
#
============================================================
============
# --- 1. INITIALISATION ET CONFIGURATION ---
#
============================================================
============
radio.on()
radio.config(channel=7, power=6) # Configuration du canal radio

# --- Configuration Matérielle ---
SERVO_PIN = pin1
CLAW_OPEN = 0
CLAW_CLOSED = 40

# --- Initialisation des composants ---
maqueenplusv2.initRobot()
np = neopixel.NeoPixel(pin15, 4)

# --- Variables d'état ---
mission_complete =False

# --- Fonction Utilitaire ---
def set_led_color(r, g, b):
    for i in range(4): np[i] = (r, g, b)
    np.show()

#
============================================================
============
# --- 2. ÉTAT INITIAL : EN MISSION ---
#
============================================================
============
maqueenplusv2.set_servo_angle(SERVO_PIN, CLAW_CLOSED) # Pince fermée sur l'objet

display.show(Image.ARROW_N) # Affiche une flèche : "J'avance"
set_led_color(0, 0, 80) # LEDs Bleues : "En mission de transport"
```

Note à l'enseignant

Correction de la carte « robot » (suite)

```
#
================================================================
=============
# --- 3. BOUCLE PRINCIPALE D'ÉCOUTE ---
#
================================================================
=============
while not mission_complete:
    message = radio.receive() # Écoute les messages

    if message == 'STOP':
        # --- ORDRE REÇU : FIN DE MISSION ! ---
        mission_complete = True # On change l'état pour arrêter la boucle

        # Action 1: Arrêt et dépôt
        maqueenplusv2.stop()
        maqueenplusv2.set_servo_angle(SERVO_PIN, CLAW_OPEN) # Ouvre la pince

        # Action 2: Feedback visuel et sonore
        display.show(Image.YES) # Affiche "Validé"
        set_led_color(0, 80, 0) # LEDs Vertes : "Mission accomplie"
        music.play(music.ENTERTAINER) # Joue une mélodie de fin
```

Activité 4 – Tout en un

Il est temps d'explorer Mars !
Les différentes parties abordées ont été reprises pour réaliser un programme complexe.

Consignes :
Ouvrir le mode multi en cliquant sur le lien : Mode Multi

Dans la partie gauche, il faut ouvrir le fichier « base mission Mars Python»

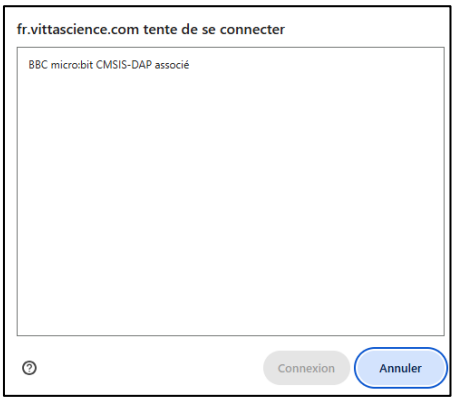
- Cliquez sur  ou puis "ouvrir projet"
- Puis taper "base mission Mars Python" dans la barre de recherche et dérouler les "projets partagés"

Dans la partie droite, il faut ouvrir le fichier « Mission Mars Python»

- Cliquez sur  ou puis "ouvrir projet"
- Puis taper "Mission Mars Python" dans la barre de recherche et dérouler les "projets partagés"

- Modifier le canal de communication dans les deux programmes (celui-ci correspond au numéro indiqué sous le robot)
- Connecter la carte « robot » à l'ordinateur à l'aide du câble USB
- Cliquer sur 

dans la partie gauche.



- Sélectionner "BBC micro:bit CMSIS-DAP" puis cliquer sur "se connecter".
- Déconnecter la carte et l'insérer dans le robot.
- Connecter la carte « base » à l'ordinateur à l'aide du câble USB
- Cliquer sur 

dans la partie droite.
- Sélectionner "BBC micro:bit CMSIS-DAP" puis cliquer sur "se connecter".
- Vous pouvez commencer les tests sur le plateau en disposant les différents objets sur le chemin.
  Attention les objets doivent être assez espacés les uns des autres.

Vous pouvez modifier les valeurs des variables dans la section « Au démarrage » pour adapter le comportement du robot.

<u>Observations et regard critique :</u>

- Quelles améliorations pourraient être apportées à votre robot ?
- Qu'est-ce qui fonctionne bien ? ne fonctionne pas ?
- Est-ce qu'un système 100% automatisé est envisageable ? …

Troisième temps du parcours – Le retour sur Terre

Retour sur votre expérience

Chaque groupe présente à la classe les résultats obtenus pendant le deuxième temps du parcours.

Faire émerger qu'il y a une différence entre réalité et simulation, que le comportement n'est pas toujours conforme au programme.

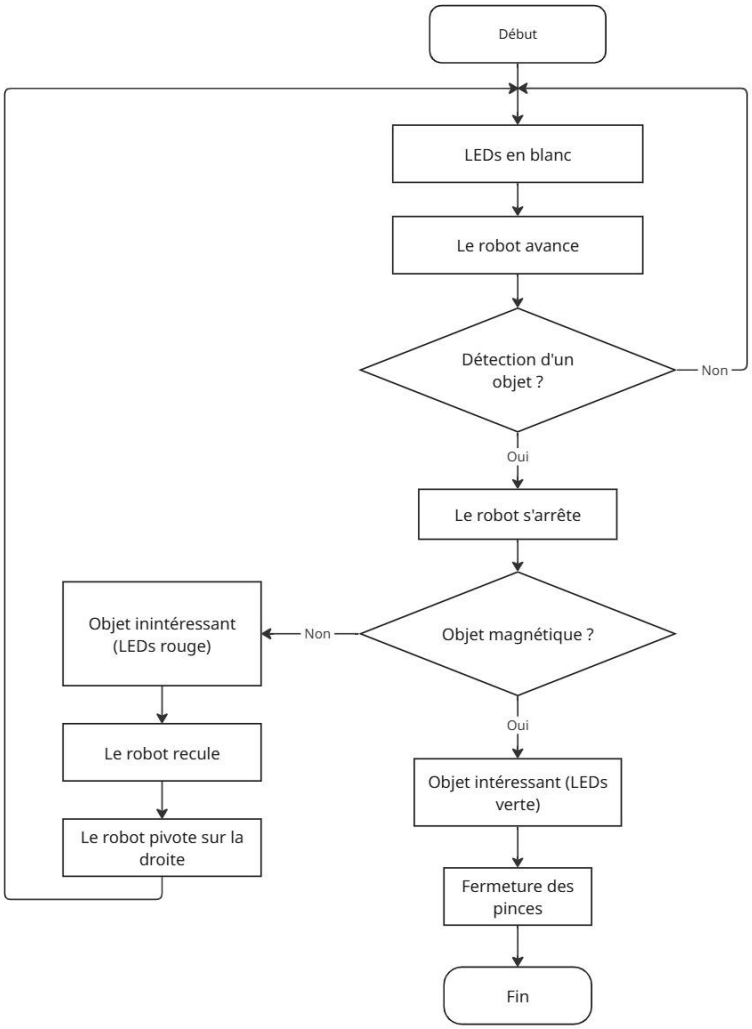
A vous de jouer !

Regarder cette vidéo.
A partir de celle-ci vous allez devoir (sur le livret élève) :

- Traduire son comportement en phrases courantes dans le livret élève
- Faire de même en langage algorithmique dans le livret élève
- Construire l'algorigramme associé dans le livret élève

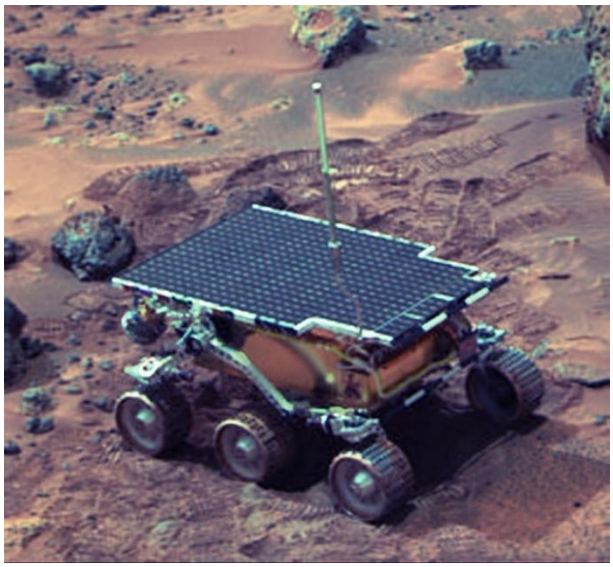
Note à l'enseignant

Correction de l'algorigramme



Etude de Sojourner et de l'exploration du sol de Mars

Le robot Sojourner est arrivée le **4 juillet 1997**.



Crédits : NASA/JPL

Durant 95 jours terrestres, ce rover a pu réaliser 16 analyses (roches et sol) et des expérimentations de mécanique des sols (abrasion des roues…)

Mais d'autres robots mobiles ont aussi exploré Mars :

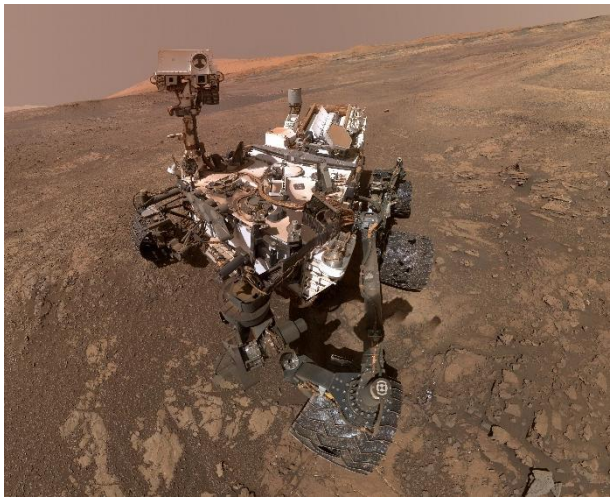
- Spirit (NASA), 4 janvier 2004. Bloqué en 2009, sa mission s'achève en 2010.

Crédits : NASA/JPL/Cornell University

- Opportunity (NASA), 25 janvier 2004. Fin de mission en 2018 pour cause de tempête de poussière.

Ces deux derniers robots sont jumeaux et ont eu une durée de fonctionnement bien supérieure à celle prévue qui devait être de 90 jours. Cela a permis une exploration bien plus vaste du sol.

- Curiosity (NASA), 6 août 2012 et toujours actif.



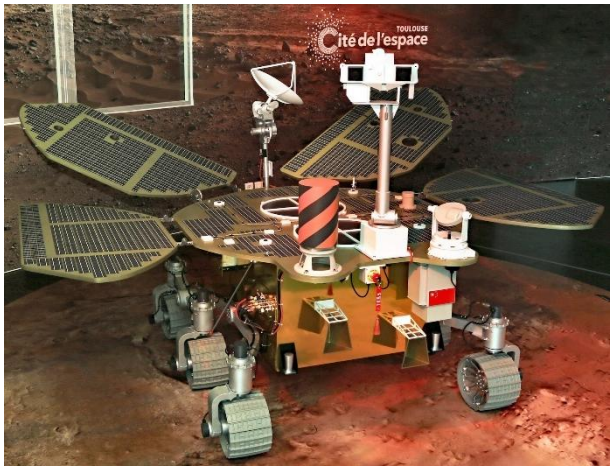
Crédits : NASA/JPL-Caltech/MSSS

Curiosity n'est plus un simple robot explorateur martien (MER) mais une mini laboratoire alimenté par un générateur nucléaire contrairement aux précédents qui étaient alimentés par panneaux solaires.

- Perseverance (NASA), 18 février 2021 et toujours actif.

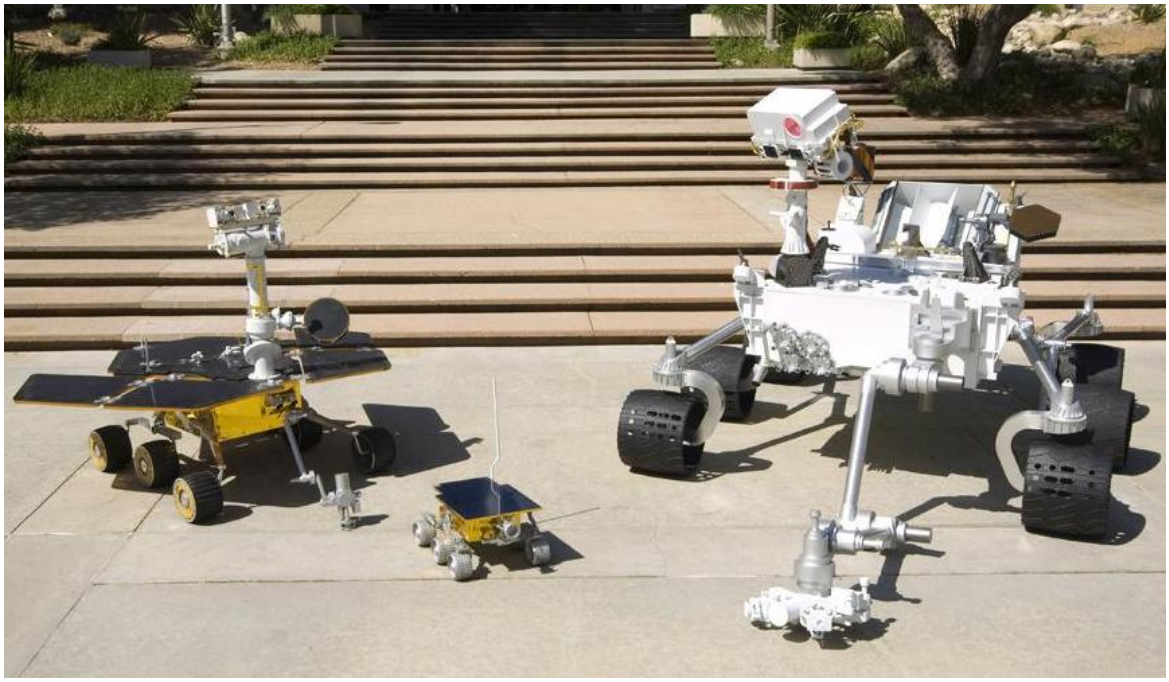
Perseverance est un robot similaire à Curiosity. Il effectue sa mission dans une autre région de la surface martienne.

- Zhurong (CNSA), 14 mai 2021, en sommeil depuis mai 2022.



Crédits : Laurent Garcia/Cité de l'espace

Zhurong est le premier rover martien chinois (il est 25 % plus petit que Curiosity). La Chine devient le deuxième pays à réussir un atterrissage martien et à établir une communication. À la suite d'une tempête de poussière pendant son hibernation, celui-ci n'a jamais pu être réveillé.



Crédits : NASA/JPL-Caltech

Des maquettes grandeur nature de trois générations de rovers martiens de la NASA illustrent l'augmentation de leur taille, du rover Sojourner (projet Mars Pathfinder) au centre, aux rovers jumeaux Spirit et Opportunity à gauche, jusqu'au rover Curiosity.

Embarquant des outils de mesure scientifique plus perfectionnés et une alimentation bien supérieure, ces robots peuvent explorer de plus vastes étendues et ainsi collecter des données plus riches.

Pour comparaison, Sojourner n'a parcouru qu'une centaine de mètres, alors que Perseverance avait parcouru plus de 25 km en mars 2024.

Contrôler un bras

Toute évolution des robots martien entraine de nouveaux défis et une précision de plus en plus grande.
Le bras de prélèvement de Perseverance nécessite d'utiliser des servomoteurs.

Si le vous souhaitez, nous proposons un parcours avec un bras analogue totalement contrôlable à distance Les mouvements que vous programmez depuis votre classe sont envoyés au bras situé au Campus Numeria (Futuroscope). Les images sont retransmises en direct sur votre écran.

Concours robotique

Si vous avez apprécié ce parcours et que vous désirez aller plus loin, il existe des concours robotiques accessibles à tous. Pourquoi ne pas participer à l'un d'entre eux avec votre propre robot ?