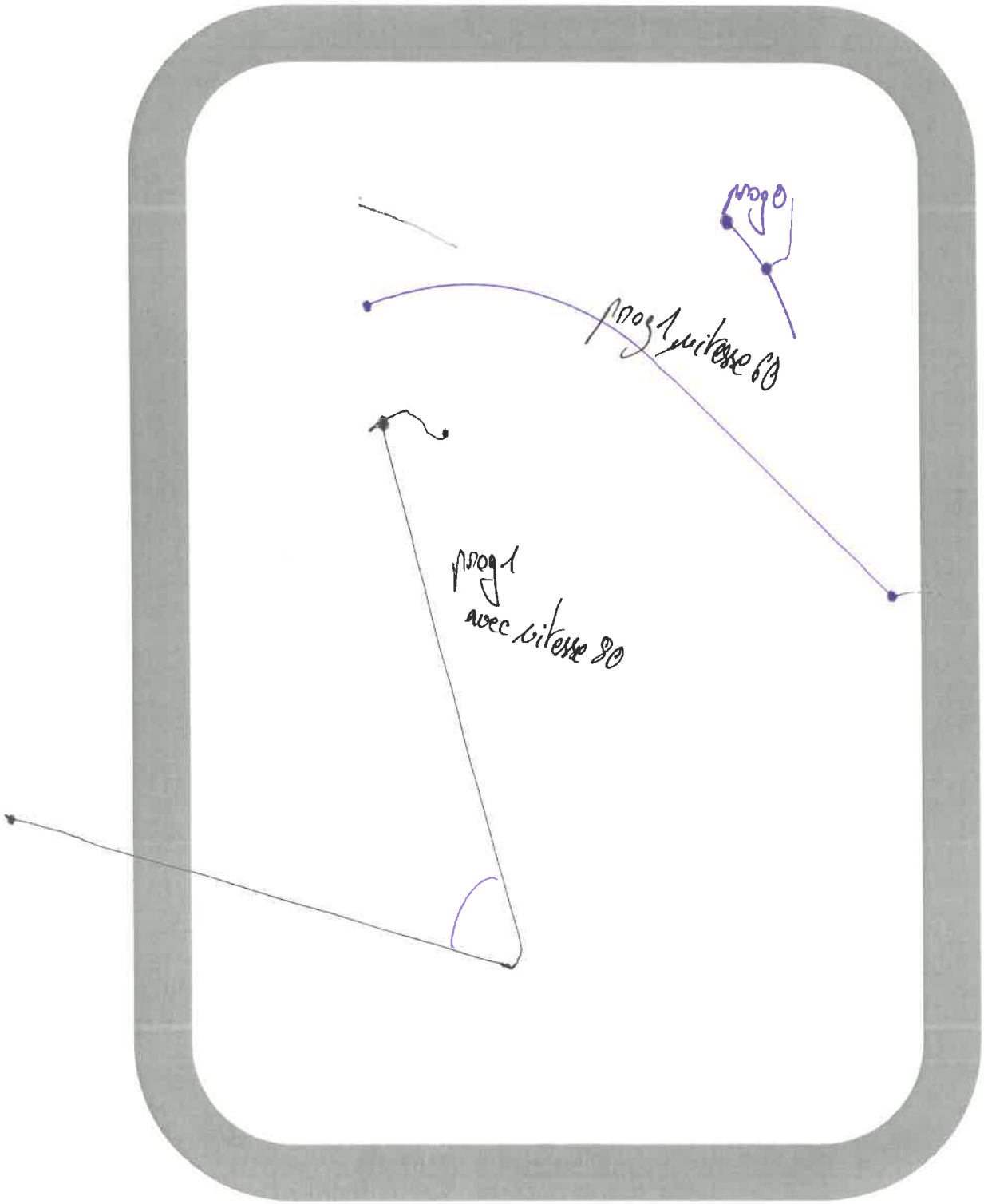
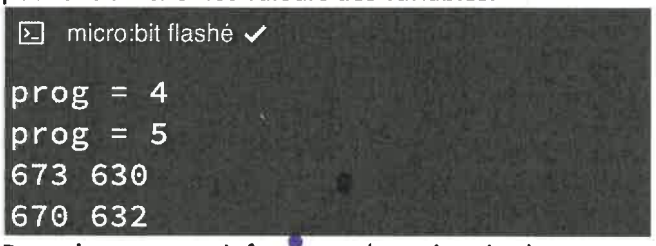
Trace

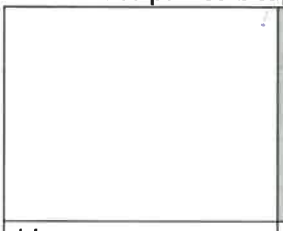
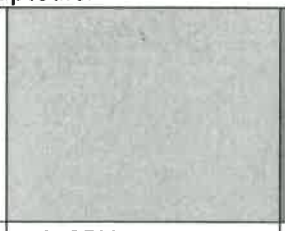


**Attention** : ce n'est pas le terminal sous le simulateur, mais celui sous votre programme!


Ceci permet de communiquer entre robot et ordinateur. Avec l'instruction `print()` nous pouvons afficher les valeurs des variables.




Posez les capteurs infrarouges (sous le robot) exactement sur les 4 cases et mesurez les valeurs lues par les 2 capteurs.

|  |  |  |  |
|---|---|---|---|
| blanc | gris 25% | gris 50% | noir |
| ~~892~~ ~~848~~ 917 560 | 222 299 | 160 242 | 71 97 |

## Prog 6 : s'arrêter au bord de la table

Quand nous appuyons le bouton B, le robot commence à avancer avec une vitesse de 60.
Si tout fonctionne bien, le robot doit s'arrêter au bord de la table.
Que fait le bout de code suivant ?

| Code | Explication |
|---|---|
| `if left < 100 or right < 100:`<br>`  robot.move(0, 0)` | Si les capteurs droit et gauche détectent une valeur de capteur moins que 100, le robot s'arrête |

Alignez une feuille avec le bord de la table et faites 2-3 traces avec le robot.

## Prog 7 : suivi de ligne

Le suivi de ligne est une tache classique en robotique. Avec les deux capteurs, nous allons faire une mesure différentielle (soustraction).

| Code | Explication |
|---|---|
| `d = left - right` | le robot calcule la différence entre à droite et la gauche et cela doit valoir 0 |
| `d = d // 10` | Cela rend l'ajustement du robot moins brusque (division entière par 10) |
| `robot.move(10 - d, 10 + d)` | le robot ajuste en fonction de la vitesse des 2 moteurs. |

Mettez un stylo dans le robot, et démontrez le suivi de ligne avec la piste sur la page suivante.

| | |
|---|---|
| | le sens de l'horloge. |
| `robot.move(0, 60, 4000)` | le robot tourne avec une vitesse de 60 avec la roue droite pendant 4000 ms |

Mesurez les angles de l'arc

| | |
|---|---|
| Angle du premier arc | |
| Angle du deuxième arc | |

## Prog 3 : dessiner un triangle

**Programmez le robot pour dessiner un triangle d'un côté d'environ 10 cm.**
**Ajoutez un commentaire à chaque ligne de code.**

```
for i in range(3):
    robot.move(60, 60, 500)
    robot.move(60, -60, 1000)
```

## Prog 4 : dessiner un hexagone

**Programmez le robot pour dessiner un hexagone d'un côté d'environ 5 cm.**
**Ajoutez un commentaire à chaque ligne de code.**

```
for i in range(6):
    robot.move(60, 60, 250)
    robot.move(60, -60, 500)
```

## Prog 5 : les 2 capteurs de lumière

**Le robot possède 2 capteurs de lumière qui lui permettent de détecter le bord de la table ou une ligne noire.**

```
if prog == 5:
    if button_b.is_pressed():
        left = pin1.read_analog()
        right = pin2.read_analog()
        print(left, right)
        sleep(200)
```

Que font ces lignes de code ?

| Code | Comportement du robot |
|---|---|
| `left = pin1.read_analog()` | le capteur gauche détecte la valeur de la couleur |
| `right = pin2.read_analog()` | le capteur droit détecte la valeur de la couleur |
| `print(left, right)` | affiche les valeurs détectées par les capteurs |

Pour lire ces valeurs, nous allons afficher le terminal série (console). Pour ceci le micro:bit doit être connecté avec un câble à l'ordinateur.

- prog2 – dessiner un S
- prog3 – dessiner un triangle
- prog4 – dessiner un hexagone

## Prog 0 : dessiner un trait

Ce programme fait avancer et reculer le robot. Que font ces deux lignes de code exactement ?

| Code | Comportement du robot |
|---|---|
| `robot.move(60, 60, 1000)` | le robot avance avec les deux roues à la vitesse 60, pendant 1000 ms |
| `robot.move(-60, -60, 500)` | le robot recule avec les 2 roues avec la vitesse 60, pendant 1000 ms |

Utilisez un stylo pour laisser une trace. Faites les mesures suivantes.

| | |
|---|---|
| Longueur de la ligne quand le robot avance | 3cm |
| Longueur de la ligne quand le robot recule | 4,5 cm |
| Calculez la vitesse du robot quand il avance (formule : vitesse = distance/ temps). Quelle est l'unité ? | 0,03 m/s |
| Vitesse du robot quand il recule | 0,09 m/s |

## Prog 1 : dessiner un V

Que font ces 3 lignes de code? Que fait le robot ? Que dessine-t-il ? Il tourne de quel angle ? dans quel sens ?

| Code | Comportement du robot |
|---|---|
| `robot.move(60, 60, 1000)` | le robot avance avec les deux roues à la vitesse 60, pendant 1000 ms |
| `robot.move(60, -60, 1000)` | le robot tourne avec une roue à une vitesse de 60 et l'autre à -60 pendant 1000 ms |
| `robot.move(60, 60, 1000)` | le robot avance avec les 2 roues à la vitesse 60, pendant 1000 ms |

Mesurez les angles

| | |
|---|---|
| Angle entre les deux lignes du V | 60° |
| Angle de pivotement du robot | 60° |

## Prog 2 : dessiner un S

Que font ces 3 lignes de code? Que fait le robot ? Que dessine-t-il ? Il tourne de quel angle ? dans quel sens ?

| Code | Comportement du robot |
|---|---|
| `robot.move(60, 0, 2000)` | le robot dessine un arc de cercle d'environs 120° dans |

Pour utiliser ces fonctions, vous devez importer le module **KitronikMOVEMotor**.
Vous pouvez trouver l'original sur GitHub:
https://github.com/KitronikLtd/micropython-microbit-kitronik-MOVE-motor/tree/master

## La méthode move()

Pour simplifier l'utilisation, nous avons ajouté une méthode supplémentaire **`move(speed1, speed2, duration=0`**. Cette méthode permet de contrôler les deux moteurs simultanément avec une vitesse allant de -255 à +255, pendant une durée spécifiée. Si la durée est 0, le robot ne s'arrête pas, et continue à bouger.

```
def move(self, speed1, speed2, duration=0):
    if speed1 == 0:
        self.motorOff('l')
    elif speed1 > 0:
        self.motorOn('l', 'f', speed1)
    else:
        self.motorOn('l', 'r', -speed1)

    if speed2 == 0:
        self.motorOff('r')
    elif speed2 > 0:
        self.motorOn('r', 'f', speed2)
    else:
        self.motorOn('r', 'r', -speed2)

    if duration > 0:
        sleep(duration)
        self.move(0, 0)
```

Expliquez la signification des 3 paramètres de la méthode **`move()`**

| | |
|---|---|
| speed1 | vitesse du moteur gauche, allant de -255 à 255 |
| speed2 | vitesse du moteur droite, allant de -255 à 255 |
| duration | la durée du mouvement |

Comme nous avons modifié le module, nous adaptons également sa version dans la première ligne.

```
# microbit-module: KitronikMOVEMotor@1.1.1
```

## Chargement des fichiers

- Ouvrez le navigateur Googe Chrome
- Allez sur la page de l'éditeur: **`python.microbit.org`**
- Chargez le programme principal : **`main.py`**
- Ajoutez le module **`KitronikMOVEMotor.py`**

## Documentation des traces

Prenez des feuilles A4, mettez votre **nom**, et la **date** du jour en haut de chaque page. Désignez vos traces par **prog0** à **prog4**. Mesurez et annotez les **distances** (en cm) et les **angles** (en degrées) à coté des traces. Ajoutez des **explications** et commentaires sur la feuille :

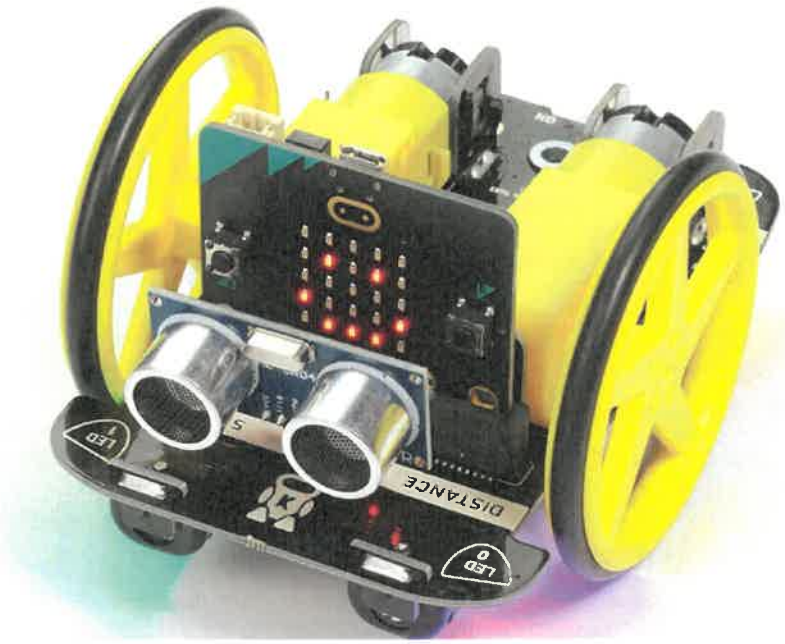
- prog0 – dessiner un trait
- prog1 – dessiner un V

# TP robot

| | | |
|---|---|---|
| Nom : Delorme | Classe : 3H4 | No. matériel: no4 |
| Prénom : Elouan | Date : 02/08/2025 | |

## Description du robot

Le robot **kitronik :MOVE motor for BBC micro:bit** est un véhicule avec 2 moteurs, un **buzzer**, 4 **LEDs**, un **capteur optique** de suivi de ligne et en **capteur ultrason** pour mesurer la distance, et 2 connecteurs pour des **servomoteurs** qui pourraient servir à contrôler un bras ou une pince.




Le micro:bit communique avec le robot en utilisant les broches suivantes (données).
En lisant la description ci-dessus, remplissez les éléments manquants du tableau.

| pins | élément | fonction |
|---|---|---|
| 0 | buzzers | faire un son, par exemple sirène, klaxon |
| 1, 2 | détecteur optique | détecter le bord de table, suivre une ligne noir |
| 8 | neopixels | allumer des LED coloriées, blanc (phares), orange (clignoteurs), rouge (freinage), |
| 13, 14 | capteurs ultrason | mesurer une distance, pour éviter un obstacle |
| 15, 16 | servomoteur 1, 2 | Bouger des pièces par exemple |
| 19, 20 | moteurs (I2C) | permettre au robot d'avancer |