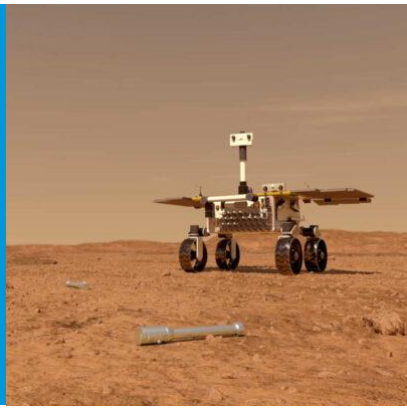
Belgium

esero

Fiche pour l'enseignant

Le rover *Fetchbot*

Phase 4 / Contrôle du rover avec Python



Une fois construit un rover muni d'une caméra (fiche « Phase 3 / Construction du rover FetchBot ») et entraîné un modèle de reconnaissance d'images avec la Teachable Machine, cette activité montre comment le faire se déplacer en le programmant de façon simple avec Python.

esa

belspo

La Scientothèque
L'ÉGALITÉ DES CHANCES PAR LES SCIENCES

CONTRÔLE DU FETCHBOT AVEC PYTHON

EN BREF

RÉSUMÉ DE L'ACTIVITÉ

Une fois que les élèves ont construit un rover muni d'une caméra et entraîné un modèle de reconnaissance d'images avec la Teachable Machine (voir fiche « Construction du rover FetchBot"), cette activité montre comment le faire se déplacer en le programmant de façon simple avec Python.

NOTIONS ABORDÉES

Robotique, programmation avec Python, condition, boucle.

TRANCHE D'ÂGE PRÉCONISÉE

14 à 18 ans

DURÉE

2 heures 30

DISPOSITIF PÉDAGOGIQUE

Par groupe de 2

PREREQUIS

1. Connaissances de bases de Python (voir Reconnaissance d'images, Phase 2 / avec Python)

2. Avoir pris en main le Raspberry Pi, et savoir s'y connecter avec VNC viewer (voir Raspberry Pi: Prise en main et préparation)

3. Avoir construit le FetchBot (voir Construction du rover, Phase 1)

MATÉRIEL

- un rover FetchBot
- un laptop/tablette par groupe de 2, avec connexion à Internet

PHASES DE L'ACTIVITÉ

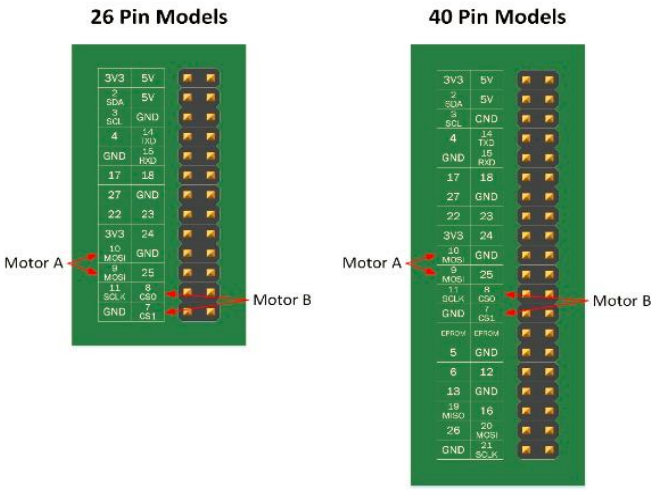
Phase	Description	Durée
1	Broches de contrôle	15 min
2	Faire avancer le robot	30 min
3	Faire tourner le robot	45 min
4	Utiliser des fonctions pour contrôler le robot	1 h

DESCRIPTION DÉTAILLÉE

PHASE 1: BROCHES DE CONTRÔLE

La carte contrôleur de moteur EduKit prend la sortie de certaines broches GPIO (General Purpose Input/Output) du Raspberry Pi et fait tourner les moteurs en avant ou en arrière selon que ces broches sont activées ou désactivées. La carte contrôleur est un type de contrôleur appelé "H Bridge". Wikipédia propose un bon article sur le pont en H pour en savoir plus.

La carte contrôleur de moteur EduKit utilise deux broches pour contrôler le moteur gauche et deux broches pour contrôler le moteur droit. Les broches GPIO 9 et 10 contrôlent le moteur A (le moteur de droite), et les broches 7 et 8 contrôlent le moteur B (le moteur de gauche).



Pour chaque moteur, une broche est utilisée pour faire tourner le moteur vers l'avant. L'autre broche GPIO fait tourner le moteur en arrière. Lorsque les deux broches sont désactivées, ou lorsque les deux broches sont activées, le moteur ne tourne pas.

Pour le FetchBot, les jeunes vont utiliser la broche GPIO 10 pour faire tourner le moteur de droite en avant, et la broche GPIO 9 pour le faire tourner en arrière.

Le moteur de gauche utilise la broche GPIO 8 pour le faire tourner en avant et la broche GPIO 7 pour le faire tourner en arrière.

PHASE 2: FAIRE AVANCER LE ROBOT

Une fois le rover construit, les jeunes peuvent s'y connecter avec VNC viewer depuis un ordinateur ou une tablette.

Demandez ensuite aux jeunes d'ouvrir l'éditeur Python Thonny, et de créer un nouveau fichier en cliquant sur l'icône "+". Ils doivent ensuite copier le code ci-dessous:

Remarque: Tous les textes après les '#" sont des commentaires, et il n'est pas obligé de les recopier. Ils aident cependant à comprendre ce que fait le code.

```
# Robot forward
import time  # Import the Time library
import gpiozero # Import the GPIO Zero Library

right_motor = gpiozero.Motor(9, 10)
left_motor = gpiozero.Motor(7, 8)

# Forward: Turn both motors on with a speed of 0.5
```

```
left_motor.forward(0.5)
right_motor.forward(0.5)

# Wait for 1 seconds
time.sleep(1)

# Turn the motors off
left_motor.stop()
right_motor.stop()
```

Il faut enfin sauvegarder le fichier sous le nom “1-rover-forward.py”.

Les jeunes sont maintenant prêts à exécuter le code.

Remarque : Avant d’exécuter le code, ne pas oublier d’allumer la batterie. Il est également plus prudent de placer le robot sur un support de manière à ce que les moteurs ne touchent pas le sol et ainsi éviter que le robot ne tombe du bureau !

Pour lancer le code, il suffit de cliquer sur le bouton “Run” de l’éditeur Thonny. Si le code ne s’exécute pas correctement, il se peut qu’il y ait une erreur dans le code qui a été tapé.

Les deux moteurs devraient tourner en avant en même temps pendant une demie seconde. Si un seul moteur tourne, il faut vérifier le câblage. Si l’un des moteurs tourne en arrière, il suffit d’intervertir les fils rouge et noir de ce moteur dans le bornier.

PHASE 3: FAIRE TOURNER LE ROBOT

Pour faire tourner le robot, il suffit de ne faire avancer qu’un moteur:

- Pour faire tourner le robot à gauche, stopper le moteur gauche et faire tourner le moteur droit:

```
# Turn left

left_motor.stop()

right_motor.forward(0.5)
```

- Pour faire tourner le robot à droite, stopper le moteur droit et faire tourner le moteur gauche

```
# Turn right

right_motor.stop()

left_motor.forward(0.5)
```

Dans l'éditeur Thonny, les jeunes doivent ensuite créer un nouveau fichier "2-rover-forward-turn.py" et copier le code ci-dessous. Il permet de faire avancer le FetchBot pendant une seconde, puis de le faire tourner à droite pendant une seconde, puis de le faire tourner à gauche pendant une seconde.

```
# Robot forward, left and right

import time  # Import the Time library

import gpiozero # Import the GPIO Zero Library


right_motor = gpiozero.Motor(9, 10)

left_motor = gpiozero.Motor(7, 8)


# Forward: Turn both motors on with a speed of 0.5

left_motor.forward(0.5)

right_motor.forward(0.5)


# Wait for 1 seconds

time.sleep(1)
```

```
# Turn the motors off
left_motor.stop()
right_motor.stop()

# Turn right: Only turn on left motor with a speed of 0.5
right_motor.stop()
left_motor.forward(0.5)

# Wait for 1 seconds
time.sleep(1)

# Turn the motors off
left_motor.stop()
right_motor.stop()

# Turn right: Only turn on right motor with a speed of 0.5
right_motor.forward(0.5)
left_motor.stop()

# Wait for 1 seconds
time.sleep(1)

# Turn the motors off
```

```
left_motor.stop()

right_motor.stop()
```

PHASE 4: UTILISER DE FONCTIONS POUR CONTRÔLER LE ROBOT

Le code ci-dessus est un peu difficile à lire. Il est possible d'utiliser des fonctions pour faire avancer ou tourner le robot, et rendre le code plus clair.

La fonction pour avancer sera appelée "forward", et prendra comme paramètres la vitesse (speed) et la durée pendant laquelle faire avancer le robot (duration). Par défaut, on mettra les valeurs de 0.5 pour la vitesse, et de 1 seconde pour la durée.

```
def forward(speed=0.5, duration=1):

    # Forward: Turn both motors on with the speed "speed"

    left_motor.forward(speed)

    right_motor.forward(speed)


    # Wait for "duration" seconds

    time.sleep(duration)


    # Turn the motors off

    left_motor.stop()

    right_motor.stop()
```

Pour aller en arrière, créons la fonction backward:

```
def backward(speed=0.5, duration=1):

    # Backward: Turn both motors on for backward rotation with the speed "speed"
    left_motor.backward(speed)
    right_motor.backward(speed)

    # Wait for "duration" seconds
    time.sleep(duration)

    # Turn the motors off
    left_motor.stop()
    right_motor.stop()
```

De la même manière, on peut créer deux fonctions pour tourner à droite et tourner à gauche. Appelons-les ‘turn_right’ et ‘turn_left’:

```
def turn_right(speed=0.5, duration=1):

    # right: Turn on right motor backward and left motor forward with a speed of 0.5
    right_motor.backward(speed)
    left_motor.forward(speed)
```

```
    # Wait for "duration" seconds
    time.sleep(duration)

    # Turn the motors off
    left_motor.stop()
    right_motor.stop()

def turn_left(speed=0.5, duration=1):

    # left: Turn on right motor foward and left motor backward with a speed of 0.5
    right_motor.forward(speed)
    left_motor.backward(speed)

    # Wait for "duration" seconds
    time.sleep(duration)

    # Turn the motors off
    left_motor.stop()
    right_motor.stop()
```

Remarque

Pour les fonctions permettant de tourner, le code a été modifié pour que les deux moteurs tournent de façon inverse, ce qui permettra au robot de tourner plus efficacement.

Dans l'éditeur Thonny, les jeunes doivent créer un nouveau fichier "motor_control.py" et copier le code ci-dessous où les fonctions sont utilisées pour contrôler le FetchBot. Avec ces fonctions, il est possible d'écrire de façon plus claire la suite de déplacements que l'on veut faire faire au rover, par exemple, avancer pendant deux secondes à la vitesse 1, puis tourner à gauche pendant 3 secondes à la vitesse 0.5, puis avancer pendant 1 seconde à la vitesse 0.5.

```
# Robot forward, left and right

import time  # Import the Time library

import gpiozero # Import the GPIO Zero Library


right_motor = gpiozero.Motor(9, 10)

left_motor = gpiozero.Motor(7, 8)


def forward(speed=0.5, duration=1):


    # Forward: Turn both motors on with the speed "speed"

    left_motor.forward(speed)

    right_motor.forward(speed)


    # Wait for "duration" seconds

    time.sleep(duration)


    # Turn the motors off
```

```
    left_motor.stop()
    right_motor.stop()

def backward(speed=0.5, duration=1):
    # Backward: Turn both motors on for backward rotation with the speed "speed"
    left_motor.backward(speed)
    right_motor.backward(speed)

    # Wait for "duration" seconds
    time.sleep(duration)

    # Turn the motors off
    left_motor.stop()
    right_motor.stop()

def turn_right(speed=0.5, duration=1):

    # right: Turn on right motor backward and left motor forward with a speed of 0.5
    right_motor.backward(speed)
    left_motor.forward(speed)

    # Wait for "duration" seconds
    time.sleep(duration)
```

```
    # Turn the motors off
    left_motor.stop()
    right_motor.stop()

def turn_left(speed=0.5, duration=1):
    # left: Turn on right motor foward and left motor backward with a speed of 0.5
    right_motor.forward(speed)
    left_motor.backward(speed)

    # Wait for "duration" seconds
    time.sleep(duration)

    # Turn the motors off
    left_motor.stop()
    right_motor.stop()
```

L'étape suivante est de créer un nouveau fichier '3-test-rover.py' pour importer ce code avec l'instruction import motor_control. Il est possible d'appeler les fonctions pour contrôler le FetchBot en rajoutant motor_control avant le nom de la fonction. Le code est beaucoup plus clair:

```
# Import functions for controlling robot
import motor_control
```

```
# Move forward

motor_control.forward(speed=1, duration=3)

# Turn left

motor_control.turn_left(speed=0.5, duration=2)

# Move forward

motor_control.forward(speed=0.5, duration=1)
```

La dernière étape est de copier ce code dans le nouveau fichier '3-rover-control.py' et de le tester.

POUR ALLER PLUS LOIN

Les jeunes savent maintenant comment faire avancer ou tourner le FetchBot à différentes vitesses, et pour différentes durées. Il est possible d'expérimenter différents parcours ou de demander aux jeunes d'écrire le code qui permettrait au robot de faire un itinéraire en forme de carré.

RÉFÉRENCES ET LIENS UTILES

- Documentation CamJam EduKit pour le contrôle des moteurs - Introduction
- Documentation CamJam EduKit pour le contrôle des moteurs - Faire tourner le robot
- Documentation de la librairie GPIO Zero

REMERCIEMENTS

Cette section reprend des éléments:

- du tutoriel CamJam EdutKit
- du tutoriel de prise en main de la caméra Raspberry de la Raspberry Pi Foundation.

LICENCE

Nous sommes intéressés par vos retours et suggestions. Vous pouvez nous contacter à contact@lascientotheque.be.