

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Programmation des LEGO EV3

Laboratoire AMPÈRE
Département FIMI - INSA de Lyon
P2I 6 : Mécatronique
Module Automatique

Dernière mise à jour le 23 février 2024

Objectifs du module “Automatique”

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Introduction

Les bases

Programmation avancée

Notion d'automatique

Conclusion

Plan

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Objectifs du module “Automatique”

Introduction

Les bases

Programmation avancée

Notion d'automatique

Conclusion

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Objectifs du module “Automatique”

- ▶ S'initier à l'automatique
- ▶ Découvrir la programmation Python adaptée au LEGO EV3
- ▶ Prendre en main le robot EV3 et s'initier à la mécatronique
- ▶ Récupérer les informations des capteurs et commander les actionneurs
- ▶ Mettre en œuvre un programme multi-tâches
- ▶ Analyser les performances du robot

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Déroulement (5 séances)

- ▶ Séance 1 (4h) : initiation à Python pour EV3
- ▶ Séance 2 (4h) : initiation à l'automatique
- ▶ Séance 3 (4h) : Mini projets et défis (1/3)
- ▶ Séance 4 (4h) : Mini projets et défis (2/3)
- ▶ Séance 5 (4h) : Mini projets et défis (3/3)

Plan

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Objectifs du module “Automatique”

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation avancée

Notion d'automatique

Conclusion

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Matériel disponible

Contenu

- ▶ 1 brique EV3
 - ▶ 6 boutons
 - ▶ 1 écran LCD
 - ▶ 1 haut parleur
 - ▶ 1 module Bluetooth
 - ▶ 1 module WiFi (en option)
 - ▶ 1 prise USB
 - ▶ 1 carte microSD
 - ▶ 1 connexion mini USB vers PC
 - ▶ 1 batterie
- ▶ 4 types de capteurs
- ▶ 2 types de moteur
- ▶ des câbles de connexion
- ▶ 1 câble USB
- ▶ 1 chargeur

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Vue d'ensemble



Pièces LEGO

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Organisation

- ▶ Plus de 1000 pièces LEGO sont disponibles
- ▶ Les capteurs et les actionneurs sont dans des boîtes séparées
- ▶ Les briques EV3 et la connectique aussi

Consigne

Pensez aux autres et rangez correctement !

Richard MOREAU

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion



Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Systèmes embarqués

Quelques exemples pour la robotique mobile :

Arduino Uno



Raspberry Pi 3



Lego EV3



Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Systèmes embarqués

Comparaisons :

	Arduino Uno	Raspberry Pi 3	Lego EV3
CPU	ATmega328P 20 MHz	ARM Cortex-A53 1.2 GHz	ARM9 300 Mhz
OS	-	Linux	Linux
Prix	24€	36€	200€

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Que choisir ?

► Arduino :

- + Système open source : Grande communauté, beaucoup d'exemples.
- + Beaucoup d'entrées/sorties
- + Multitude de modules
- +/- Pas d'OS
 - Performances

Idéal pour expérimenter et réaliser des premiers prototypes

Que choisir ?

Objectifs

Introduction

What is in the box?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

- ▶ Raspberry :
 - + Performances
 - + OS Linux : Clavier, Écran, Serveur
 - + Communauté active
 - Peu d'entrées/sorties

Un mini ordinateur pour quelques euros.

Que choisir ?

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

► LEGO :

- + Construire un robot en quelques clips
- + OS Linux : Server
 - Capteurs LEGO seulement
 - Prix

Idéal pour apprendre.

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Petit historique du LEGO Mindstorm

Avant l'EV3, il y avait



RCX



NXT

Hardware

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

	RCX	NXT	EV3
Date	1998	2006	2013
CPU	H8/300 16MHz	ARM7 48MHz	ARM9 300MHz
Mémoire	32 KB RAM 16 KB ROM	64 KB RAM 256 KB Flash	64 MB RAM 16 MB Flash
LCD	Segmented	100x64	178x128
Entrée	3	4	4
Sortie	3	3	4

Spécificité EV3 :

- ▶ slot microSD pour installer un OS
- ▶ OS Linux embarqué

Software

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Possibilité de programmer :

- ▶ en C++
- ▶ en C
- ▶ en Java
- ▶ en Python
- ▶ en Micropython
- ▶ en Matlab
- ▶ en programmation graphique (Mindstorm, LabView, Matlab/Simulink)
- ▶ avec Arduino

Solution retenue

Visual Studio Code + EV3 dev Python

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Configurer Visual Studio Code (1/3)

1. Lancer Visual Studio Code
2. Installer les extensions :
 - 2.1 EV3DEV-browser
 - 2.2 python
3. Choisir comme espace de travail un dossier de votre *HOME INSA*.
par exemple H:/Documents/P2I6/EV3/
4. Dans le terminal de VS Code :

pip install python-ev3dev2
5. Fermer et relancer VS Code

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Configurer Visual Studio Code (2/3)

6. Aperçu général de Visual Studio Code

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Hello.py - EV3Python - Visual Studio Code
- File Menu:** Fichier, Edition, Sélection, Affichage, Atteindre, Exécuter, Terminal, Aide
- Editor:** Hello.py (content:

```
1 #!/usr/bin/env python3
2
3 import time
4 print('Hello World')
5 time.sleep(5)
```

)
- Explorer:** EV3PYTHON (Hello.py)
- Terminal:** ev3dev (Output: Starting: brickrun --directory="/home/robot/EV3Python" "/home/robot/EV3Python/Hello.py"

Completed successfully.)
- Device Browser:** EV3DEV DEVICE BROWSER (ev3dev (Status: Battery: 7.74V, /home/robot, EV3Python))
- Bottom Status Bar:** L 5, col 14 Espaces : 4 UTF-8 LF Python 3.8.10 64-bit

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

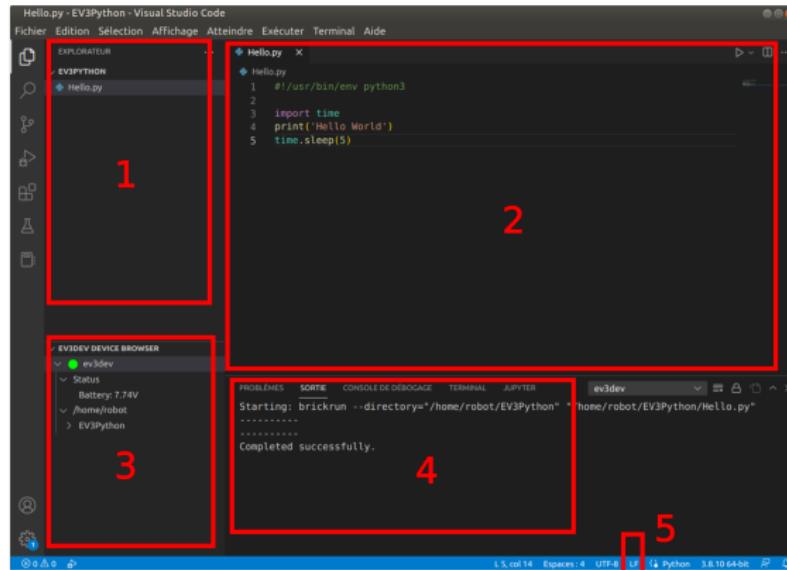
Programmation

Automatique

Conclusion

Configurer Visual Studio Code (3/3)

1. Explorateur de fichiers locaux
2. Éditeur de code
3. Explorateur de fichiers distants
4. Boîte de dialogue (débogueur, terminal, ...)
5. Vérification que la fin de la ligne est bien configuré (LF)



Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Tester votre configuration (1/3)

Démarche

1. Créer un nouveau fichier HelloWorld.py
2. La première ligne de votre code est forcément :
`#!/usr/bin/env python3`
3. Ensuite vous pouvez écrire votre code Python normalement. Le programme HelloWorld s'écrit donc :

```
#!/usr/bin/env python3
from time import sleep # Pour utiliser la méthode sleep
print('Hello World')
sleep(5) # Pour laisser le message affiché 5 secondes
```

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Tester votre configuration (2/3)

Démarche

4. Pour connecter la brique LEGO, sur Linux, tout fonctionne tout seul, il n'y a rien à faire ;)
5. Pour Windows, il faut
 - 5.1 aller dans l'extension EV3DEV Device Browser puis cliquer sur **Click here to connect to a device**
 - 5.2 cliquer sur **I don't see my device**
 - 5.3 lui donner un nom ('EV3USB' par exemple) et lui rentrer l'adresse IP qui apparaît sur le LEGO.
 - 5.4 dans le menu de Brickman -> Wireless & Network -> All Network Connections -> Wired -> Connect

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Tester votre configuration (3/3)

Démarche

Une fois la brique connectée, pour lancer votre code, il faut :

6. appuyer sur la touche F5
7. choisir l'environnement ev3dev
8. choisir Download and run current file in output pane

Résultat

Vous devriez voir le message ‘‘Hello World’’ s'afficher sur la brique Lego pendant 5 secondes.

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

Si vous n'avez pas pris votre loupe ...

Problème d'affichage

Le texte '**Hello World**' qui apparaît est difficilement lisible à l'écran car il est trop petit.

2 solutions

- Il faut modifier la police d'affichage. Pour cela, il suffit d'ajouter :

```
import os
os.system('setfont Lat15-TerminusBold14')
```

- Il est possible d'afficher le texte sur le terminal de Visual Studio Code. Pour cela il suffit de préciser que l'affichage doit se faire dans le fichier **strerr** plutôt que celui par défaut :

```
from sys import stderr
print("Hello World",file=stderr)
```

Objectifs

Introduction

What is in the box ?

Hardware

Software

Les bases

Programmation

Automatique

Conclusion

En cas de problème

Un problème ? ... Une solution

1. Vérifier que la brique est allumée.
2. Vérifier que le câble USB est branché sur la brique EV3 **ET** sur le PC.
3. Lire les éventuels messages d'erreur.
4. Recommencer.
5. Ne pas paniquer et appeler le prof.

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Plan

Objectifs du module “Automatique”

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation avancée

Notion d'automatique

Conclusion

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

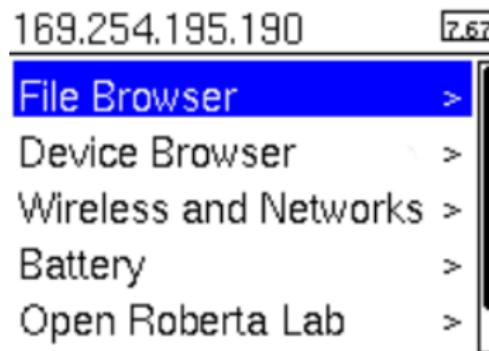
Automatique

Conclusion

L'affichage

L'écran LCD affiche :

- ▶ son adresse IP (169.254.195.190)
- ▶ l'état de la batterie (7.67)
- ▶ un menu
- ▶ la taille est de 178x128 pixels monochromes



Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Rappel

Alternative à l'affichage sur l'écran LCD

- ▶ L'affichage peut se faire sur le terminal de VSCode
- ▶ Il faut que le LEGO soit connecté
- ▶ Il faut préciser dans l'affichage que la sortie se fera dans le fichier stderr
- ▶ Cela peut être utile pour afficher la valeur d'un capteur par exemple

```
# !/usr/bin/env python3
from sys import stderr
from time import sleep

while True:
    # print to EV3 LCD screen
    print("Hello World")
    # print to VS Code output panel
    print("Hello World", file =stderr)
    sleep(0.5)
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Pour gérer le délai d'exécution

La méthode `sleep`

Il est possible de faire une pause avant de passer à l'instruction suivante en faisant appel à :

`time.sleep(second)`

où `second` représente le nombre de secondes à attendre.

Attention il faut penser à importer la librairie `time` en rajoutant en début de programme :

`from time import sleep`

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Les LEDs

Description

- ▶ Il y a 2 LEDs sous les boutons (**LEFT** et **RIGHT**).
- ▶ 7 couleurs sont pré définies :
RED, GREEN, YELLOW, ORANGE, AMBER, BLACK

Utilisation des LEDs des LEGO

- ▶ Il faut créer une instance de la classe **Leds**
- ▶ On peut ensuite utiliser la méthode
set_color('uneLed', 'uneCouleur') où :
 - ▶ '**uneLed**' est à choisir parmi '**LEFT**' et '**RIGHT**'
 - ▶ '**unecouleur**' est à choisir parmi les couleurs citées ci-dessus
- ▶ La méthode **all_off()** permet de tout éteindre

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Les LEDs

Exemple

```
#!/usr/bin/env python3
from ev3dev2.led import Leds
from time import sleep
leds = Leds()
leds.all_off() # LEDs éteintes
sleep(1)
# Les 2 deviennent AMBER
leds.set_color('LEFT', 'AMBER')
leds.set_color('RIGHT', 'AMBER')
sleep(4)
# La gauche devient verte et la droite rouge
leds.set_color('LEFT', 'GREEN')
leds.set_color('RIGHT', 'RED')
sleep(4)
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

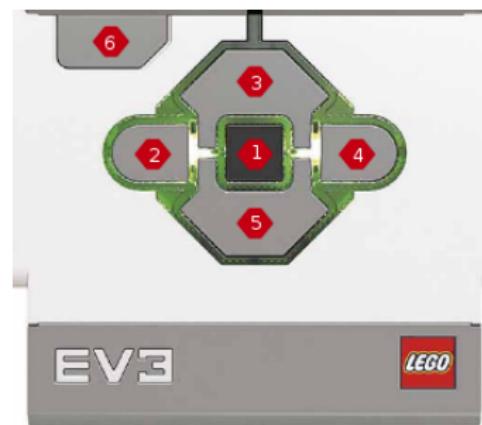
Automatique

Conclusion

Les boutons

6 boutons sont disponibles :

1. **ENTER**
2. **LEFT**
3. **UP**
4. **RIGHT**
5. **DOWN**
6. **BACKSPACE**



Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Les boutons

Pour récupérer l'identité des boutons, on crée une instance de la classe **Button()** :

```
btn = Button()
```

Lors de l'appui sur un bouton, les variables suivantes deviennent **True** :

- ▶ **btn.left**
- ▶ **btn.right**
- ▶ **btn.up**
- ▶ **btn.down**
- ▶ **btn.enter**
- ▶ **btn.backspace**

Particularité

Attention le bouton **backspace** quitte automatiquement le programme.

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Les boutons

Pour attendre l'appui sur n'importe quel bouton :

btn.any()

Particularités

A chaque appui sur un bouton, des caractères spéciaux apparaissent (par exemple '^ [[D' apparaît quand on appuie sur LEFT) lorsque l'affichage se fait sur le LCD de la brique.

Ce n'est pas le cas lorsque l'affichage se fait sur le terminal de VSCode

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Exemple

Réaction suite à l'appui sur les boutons

```
#!/usr/bin/env python3
from ev3dev2.button import Button
from sys import stderr

btn = Button()
print('GO !!!')
while not (btn.enter and btn.up):
    if btn.left :
        print('LEFT has been pressed',file=stderr)
    if btn.right :
        print('RIGHT has been pressed',file=stderr)
    if btn.up :
        print('UP has been pressed',file=stderr)
    if btn.down :
        print('DOWN has been pressed',file=stderr)
    if btn.enter :
        print('ENTER has been pressed',file=stderr)
print('FIN')
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Les capteurs

Capteur de contact



Capteur ultrason



Capteur de couleur



Capteur d'angle



Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Capteurs

Fonctionnement générique

- ▶ Un capteur est une instance d'une **classe** provenant de la librairie **ev3dev2.sensor**.
- ▶ Le constructeur prend un argument en entrée qui indique sur quel port est branché le capteur : **INPUT_X** où $X \in [1, 4]$
- ▶ Il faut indiquer un mode de fonctionnement en modifiant l'attribut **mode**
- ▶ Il est possible d'envoyer une commande particulière au capteur en modifiant l'attribut **command**
- ▶ La valeur est obtenue *via* la méthode **value()**

Exemple

```
#!/usr/bin/env python3
from ev3dev2.sensor import Sensor, INPUT_1
mon_capteur = Sensor(INPUT_1)
mon_capteur.mode='UN_MODE'
mon_capteur.commande='MA_COMMANDE'
valeur_capteur = mon_capteur.value()
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Capteur de contact

Fonctionnement

1 seul mode : **TOUCH**

- ▶ Appuyé (renvoie 1)
- ▶ Relâché (renvoie 0)



Exemple

```
#!/usr/bin/env python3
```

```
from ev3dev2.sensor import Sensor, INPUT_1
mon_capteur = Sensor(INPUT_1)
mon_capteur.mode='TOUCH'
while (True):
    print(mon_capteur.value())
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Capteur de distance

Fonctionnement

7 modes de fonctionnement dont :

- ▶ **US-DIST-CM** : Mesure en centimètres
- ▶ **US-DC-CM** : Mesure en millimètres
- ▶ **US-LISTEN** : pour “écouter” d’autres capteurs US



Exemple

```
#!/usr/bin/env python3

from ev3dev2.sensor import Sensor, INPUT_1
mon_capteur = Sensor(INPUT_1)
mon_capteur.mode='US-DIST-CM'
while (True):
    print(mon_capteur.value())
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Capteur de couleur

Fonctionnement

6 modes de fonctionnement dont :

- ▶ **COL_AMBIENT** : pourcentage de la lumière (led off)
- ▶ **COL_REFLECT** : pourcentage de la lumière réfléchie(led on)
- ▶ **RGB_RAW** : la proportion de RGB entre 0 et 255 pour chaque composante



Exemple

```
#!/usr/bin/env python3

from ev3dev2.sensor import Sensor, INPUT_1
mon_capteur = Sensor(INPUT_1)
mon_capteur.mode='COL-REFLECT'
while (True):
    print(mon_capteur.value())
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Capteur d'angle

Fonctionnement

3 modes de fonctionnement :

- ▶ **ANGLE** : mesure de l'angle entre 0 et 359°
- ▶ **ANGLE-ACC** : Mesure de l'angle cumulé
- ▶ **SPEED** : Mesure de la vitesse angulaire

2 commandes :

- ▶ **RESET** : pour mettre à 0 l'angle cumulé
- ▶ **CAL** : pour calibrer la position actuelle à 0 et la sauvegarder



Exemple

```
#!/usr/bin/env python3

from ev3dev2.sensor import Sensor, INPUT_1
mon_capteur = Sensor(INPUT_1)
mon_capteur.mode='ANGLE'
while (True):
    print(mon_capteur.value())
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

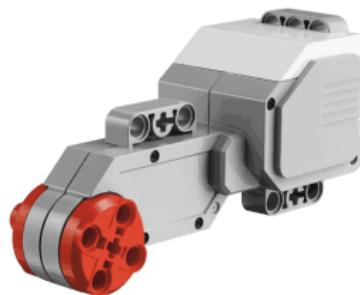
Conclusion

Les actionneurs

Moteurs à Courant Continu

2 types de moteurs sont disponibles

LargeMotor



MediumMotor



Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

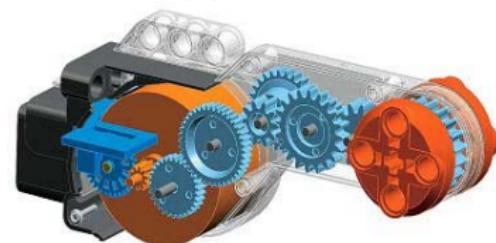
Conclusion

Les actionneurs

Version RCX



Version NXT



	RCX	NXT	EV3	
			LargeMotor	MediumMotor
Poids [g]	28	80	82	39
Vitesse [tr/min]	340	170	175	260
Couple [Ncm]	5.5	50	43	15
Encodeur	X	✓	✓	✓

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Les actionneurs

Comment commander les moteurs ?

- ▶ Pour commander en puissance
⇒ **Moteur non asservi**

- ▶ Pour commander en vitesse et en position
⇒ **Moteur asservi**

Pourquoi les moteurs non asservis ?

- ▶ Découvrir l'asservissement
- ▶ Mettre en œuvre un asservissement
- ▶ Utiliser les encodeurs des moteurs
- ▶ Commandes en vitesse et en position possibles

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Commande de moteur non asservi

Classe Motor

On utilisera en particulier la méthode **on(power)** avec **power** une valeur comprise dans l'intervalle [-100,100].

Il s'agit en fait du pourcentage de la puissance du moteur

- ▶ puissance maximale pour le LargeMotor : 1050 deg/sec
- ▶ puissance maximale pour le MediumMotor : 1560 deg/sec

Et si power \notin [-100,100] ?

Le programme s'arrête automatiquement. Il est donc nécessaire de bien veiller à ce que **power** reste dans l'intervalle [-100,100].

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Commande de moteur non asservi

Programme basique d'un moteur

```
#!/usr/bin/env python3

from time import sleep
from ev3dev2.motor import Motor, OUTPUT_A

mon_moteur=Motor(OUTPUT_A) # Moteur branché en A

mon_moteur.on(100) # En avant à fond
sleep(1)
mon_moteur.on(0) # Moteur à l'arrêt
sleep(1)
mon_moteur.on(-20) # En arrière doucement
sleep(1)
```

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Commande de moteur non asservi

Autres méthodes

De nombreuses méthodes ont été créées pour contrôler les moteurs, seules quelques unes seront utilisées dans ce module :

- ▶ `on(power)` pour appliquer une puissance au moteur.
- ▶ `on_for_seconds(40,3)` pour faire tourner le moteur dans le sens horaire à 40% de sa puissance maximale pendant 3 secondes (équivalent à `on(40)` suivi de `sleep(3)`).

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Les encodeurs des moteurs

Récupérer la position du moteur

► **position**

Cet attribut stocke la position angulaire du moteur [deg].
Peut être utilisé pour réinitialiser la position angulaire du moteur.

► **speed**

Cet attribut stocke la vitesse angulaire du moteur [deg/sec].

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Exercices niveau 1

Objectifs :

- ▶ savoir afficher des informations sur l'écran LCD et sur VSCode
- ▶ savoir utiliser le capteur de contact
- ▶ savoir envoyer une consigne de puissance à un moteur

Exercices :

1. Au début du programme, les LEDs sont jaunes et "**Let's blink LEDs**" doit s'afficher sur l'écran LCD. Lors de l'appui sur le capteur de contact, la LED gauche alterne entre le rouge et le vert toutes les 200ms (inversement pour la LED droite). Les 2 LEDs s'éteignent quand le capteur de contact est relâché et recommencent à clignoter lorsqu'il est de nouveau pressé.
2. Suite à l'appui sur le bouton **LEFT** (réciproquement **RIGHT**) du LEGO, le moteur tourne à 30% (réciproquement -30%) pendant 2 secondes. Si le bouton **UP** est appuyé, le moteur s'arrête et c'est la fin du programme
3. Faire tourner un moteur à 10% et augmenter sa puissance de 10% toutes les secondes jusqu'à 80%. Afficher le pourcentage de puissance sur le LCD et VSCode.

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Exercices niveau 1 - suite

Objectifs :

- savoir récupérer les informations de l'encodeur du moteur

Exercices :

4. Faire tourner un moteur à 30% pendant (environ) 3 tours.
Afficher l'angle effectué sur VSCode.
5. À chaque appui sur le capteur de contact, un moteur effectue un tour à 30% de sa puissance.

Objectifs

Introduction

Les bases

Affichage & Boutons

Capteurs

Actionneurs

Exercices niveau 1

Programmation

Automatique

Conclusion

Exercices niveau 1 - fin

Objectifs :

- ▶ savoir utiliser les autres types de capteur
- ▶ changer le comportement des moteurs en fonction de la valeur des capteurs

Exercices :

6. Si le capteur de couleur est au dessus d'une zone noire, le moteur s'arrête, sinon il fonctionne à 30%.
7. Un moteur fonctionne à 30% lorsqu'il est entre 20 et 40cm d'un obstacle sinon il est à l'arrêt.
8. La puissance du moteur dépend de la valeur de l'angle lue par le capteur angulaire. Par exemple si le capteur indique 100° le moteur doit aller à 100%. Si le capteur indique -50° le moteur fonctionne à 50% dans l'autre sens.

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Plan

Objectifs du module “Automatique”

Introduction

Les bases

Programmation avancée

Mémorisation des signaux

Timer

Exercices niveau 2

Notion d'automatique

Conclusion

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Mémorisation des signaux

Quels signaux ?

- ▶ Les informations envoyées aux actionneurs.
- ▶ Les informations mesurées par les capteurs.

Pourquoi ?

Afin d'analyser et d'interpréter les résultats.

Comment ?

En enregistrant les valeurs dans un fichier.

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Enregistrement des données

Rappel

Pour écrire dans un fichier des données stockées dans une liste

```
import csv
# Création de listes
data = [] # Pour stocker toutes les données
data_en_cours = [] # Pour stocker les données actuelles

# Ajouts de valeur dans la liste
for i in range(0,3):
    data_en_cours = [1*i, 2*i, 3*i]
    data.append(data_en_cours)

# Enregistrement dans un fichier csv
with open('data.csv', 'w', encoding='utf-8') as fichier :
    writer = csv.writer(fichier, delimiter=';')
    writer.writerow(data)
```

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Exemple

Enregistrer les valeurs d'un encodeur

```
#!/usr/bin/env python3

import csv
from ev3dev2.motor import Motor, OUTPUT_A

# Initialisation des données
mon_moteur = Motor(OUTPUT_A)
mon_moteur.position=0 # Pour initialiser l'encodeur
puissance=30
data = [] # Pour stocker toutes les données
data_en_cours = [] # Pour stocker les données actuelles

# Boucle pour gérer le moteur et l'enregistrement
while mon_moteur.position<1080:
    mon_moteur.on(puissance) # Le moteur tourne
    data_en_cours = [puissance , mon_moteur.position] # Enregistrement des
    # données actuelles
    data.append(data_en_cours) # Stockage des données

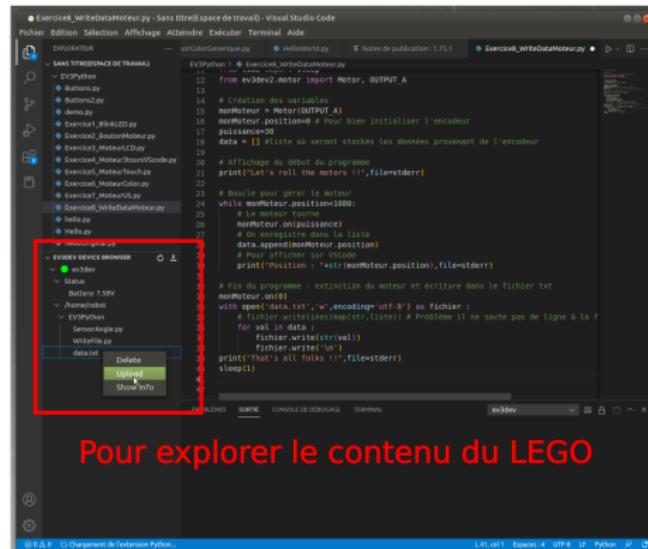
# Fin du programme : extinction du moteur et écriture dans le fichier csv
mon_moteur.on(0)
with open('data.csv', 'w',encoding='utf-8') as fichier :
    writer = csv.writer( fichier , delimiter=';')
    writer .writerows(data)
```

Exemple

Récupération du fichier

Dans l'onglet EV3DEV Device Browser de VS Code :

- ▶ développer le dossier du projet
 - ▶ click droit puis Upload



Gestion du temps

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Plusieurs possibilités :

- ▶ La méthode `sleep` permet d'attendre avant d'exécuter une instruction.
- ▶ La méthode `time` permet de récupérer le temps actuel.
- ▶ La classe `Timer` permet de répéter périodiquement une méthode.

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Gestion du temps - la méthode sleep

Méthode `sleep`

Comme vu jusqu'à maintenant, il est possible d'attendre X secondes avant de passer à l'instruction suivante

Exemple

```
#!/usr/bin/env python3
from time import sleep
from ev3dev2.motor import Motor, OUTPUT_A

mon_moteur=Motor(OUTPUT_A) # Moteur branché en A
mon_moteur.on(50) # Moteur à 50% de sa puissance
sleep(3.5) # Attente de 3.5 secondes
mon_moteur.on(0) # Moteur éteint
```

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Gestion du temps - la méthode time

Méthode `time`

La méthode `time` permet de récupérer le temps écoulé depuis l'*epoch* (01/01/1970 par défaut)

Exemple

```
#!/usr/bin/env python3
from time import time
from ev3dev2.motor import Motor, OUTPUT_A

mon_moteur=Motor(OUTPUT_A) # Moteur branché en A
temps = 0 # Pour stocker le temps écoulé
t0 = time() # pour définir le temps de départ

while(temps<3.5)
    temps = time() - t0 # Mise à jour du temps
    mon_moteur.on(50) # Moteur à 50% de sa puissance
    mon_moteur.on(0) # Moteur éteint
```

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Gestion du temps - la classe Timer

Classe Timer

La classe **Timer** permet de répéter périodiquement l'exécution de certaines méthodes.

Création d'une classe personnalisée

```
from threading import Timer
# Définition d'une nouvelle classe
class Montimer:
    # Le constructeur avec comme arguments le délai et la fonction à répéter
    def __init__(self, delai, fonction):
        self.delai = delai
        self.fonction = fonction
        self.timer = Timer(delai, self.run) # création du Timer
    # Lors de l'appel à start, elle lance le Timer
    def start(self):
        self.timer.start()
    # A son exécution, elle relance le Timer pour qu'elle se répète
    def run(self):
        self.timer = Timer(self.delai, self.run)
        self.timer.start()
        self.fonction()
    # Lors de l'appel à cancel, le Timer s'arrête
    def cancel(self):
        self.timer.cancel()
```

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Gestion du temps - la classe Timer

Comment utiliser la classe `MonTimer` ?

Il suffit de créer une instance de cette classe en passant en argument la période de répétition et la méthode à exécuter.

Exemple

```
from threading import Timer
# Définition d'une nouvelle classe
class MonTimer:
    # Le code de la classe décrit précédemment

    def tic():
        print("tic")

    def tac():
        print("\t tac")

    print("start")
tache1 = MonTimer(0.1,tic) # La 1ère tâche est répétée toutes les 0.1s
tache2 = MonTimer(0.5,tac) # La 2nde tâche est répétée toutes les 0.5s
tache1.start() # Lancement de la 1ère tâche
tache2.start() # Lancement de la 2nde tâche
while(condition_de_sortie):
    # du code ici
tache1.cancel() # Arrêt de la 1ère tâche
tache2.cancel() # Arrêt de la 2nde tâche
```

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Analyse des données

Rappel sur **Matplotlib**

La librairie **Matplotlib** permet de faire des graphes pour visualiser des données par exemple.

Attention, il ne faut pas exécuter ce code dans le lego mais en python simplement.

Il ne faut donc pas ajouter `#!/usr/bin/env python3` en début de code et choisir **python** et non pas **ev3dev** à l'exécution du code.

Installer la librairie **Matplotlib**

Il est possible que la librairie **Matplotlib** ne soit pas installée sur les ordinateurs.

Dans ce cas, dans le terminal de VSCode, il suffit de lancer :

pip install Matplotlib

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Analyse des données

Exemple

```
import matplotlib.pyplot as plt
data_x=[]
data_y=[]

# Pour récupérer les données
with open('data.csv', 'r' ,encoding='utf-8') as fichier :
    reader = csv.reader( fichier , delimiter=';')
    for row in reader:
        data_x.append(float(row[0]))
        data_y.append(float(row[1]))

# Pour faire un graphe
plt.figure()
plt.plot(data_x,data_y,"r--")
plt.title ("y = f(x)")
plt.xlabel ("x [unit]")
plt.ylabel ("y [unit]")
plt.show()
```

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Analyse des données

Ressources

Vous pouvez télécharger la plupart des codes présentés dans ce cours sur moodle dans le module Automatique / section Ressources ou en cliquant sur les fichiers ci-dessous :

- ▶ Pour écrire un fichier texte :
WriteFile.py
- ▶ Pour traiter les données et faire un graphe :
PlotData.py
- ▶ Pour créer un Timer qui permet de lancer des fonctions de manière périodique :
TimerPeriodic.py

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Exercice niveau 2

Objectifs :

- ▶ Comprendre la gestion du temps avec la méthode **time**.
- ▶ Enregistrer les données dans un fichier.

Exercice 2.1 :

1. Faire tourner un moteur à 30% de sa puissance pendant 5 secondes en utilisant la méthode **time**.
2. Enregistrer la position angulaire du moteur dans une liste. De même pour le temps.
3. Enregistrer ces listes dans un fichier texte.
4. Avec un programme Python, tracer la position angulaire en fonction du temps.
5. Quelle est la vitesse de rotation moyenne du moteur ? Est ce cohérent avec la vitesse maximale de 1050 deg/s ?
6. Quelle est le temps minimum, maximum et moyen entre chaque acquisition de mesure ?

Objectifs

Introduction

Les bases

Programmation

Mémorisation des signaux

Timer

Exercices niveau 2

Automatique

Conclusion

Exercices niveau 2

Objectifs :

- ▶ Comprendre la gestion du temps avec la classe **TimerPeriodic**.
- ▶ Enregistrer les données dans un fichier.

Exercice 2.2 :

1. Augmenter la puissance d'un moteur de 20% toutes les 2 secondes jusqu'à atteindre 80%
2. Le programme se termine au bout de 2 secondes à 80% **ou** suite à l'appui sur le capteur de contact.
3. Les valeurs de la position du moteur doivent être enregistrées toutes les 50ms.
4. Calculer la vitesse de rotation expérimentale instantanée et tracer la en fonction du temps.
5. Tracer également la valeur théorique de la vitesse de rotation. Pour rappel, la vitesse maximale du moteur est de 1050 deg/s.

Objectifs

Introduction

Les bases

Programmation

Automatique

Commande de mon premier robot

Conclusion

Plan

Objectifs du module “Automatique”

Introduction

Les bases

Programmation avancée

Notion d'automatique

Commande de mon premier robot

Conclusion

Objectifs

Introduction

Les bases

Programmation

Automatique

Commande de mon premier robot

Conclusion

Premier Robot



Ultrasonic Sensor

Objectifs

Introduction

Les bases

Programmation

Automatique

Commande de mon premier robot

Conclusion

Exercices niveau 3

Objectifs :

- ▶ Réaliser un asservissement
- ▶ Mettre à jour les données des capteurs périodiquement
- ▶ Agir sur les actionneurs en fonction des mesures

Exercice 3 :

1. Réaliser le robot Ultrasonic à l'aide des instructions ^a.
2. A l'aide du capteur **Ultrasonic**, faire en sorte que le robot reste à 10cm d'un obstacle immobile.
3. Quelle type de commande est la plus appropriée (P, PI, PID ou PD) ?
4. Dans un programme, tracer la puissance du moteur, la consigne et la mesure de la position en fonction du temps.
5. Interpréter les résultats.
6. Refaire l'expérience avec un obstacle mobile.

a. Cliquer ici pour télécharger les instructions

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Plan

Objectifs du module “Automatique”

Introduction

Les bases

Programmation avancée

Notion d'automatique

Conclusion

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Conclusion

- ▶ Récupération les informations d'un capteur
- ▶ Commander un moteur
- ▶ Gérer le temps d'exécution des commandes
- ▶ Enregistrer les données en vue d'une analyse *a posteriori*
- ▶ Concevoir et commander un robot mobile

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Pour aller plus loin

- ▶ Réaliser un asservissement en boucle ouverte
- ▶ Réaliser un asservissement en boucle fermée
- ▶ Asservissement d'un pendule monté sur un robot mobile
- ▶ Concevoir et commander un robot de type SCARA

Objectifs

Introduction

Les bases

Programmation

Automatique

Conclusion

Références

- ▶ Le site ev3dev pour EV3
<https://www.ev3dev.org/>
- ▶ “Programmation NX - Application à la commande d'un robot NXT LEGO”
by Richard MOREAU, diaporama de cours INSA de Lyon, 2013
- ▶ “Programmation LeJOS - Application à la commande d'un robot EV3 LEGO”
by Richard MOREAU, diaporama de cours INSA de Lyon, 2015