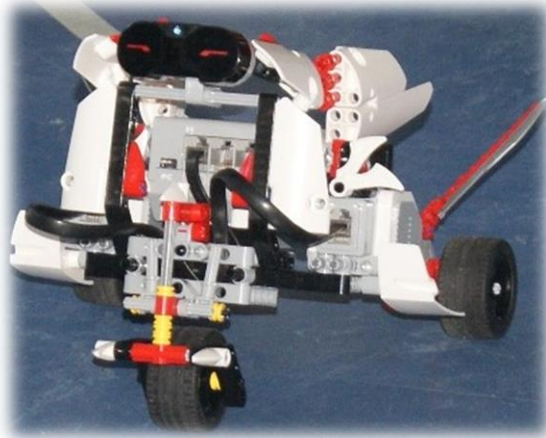


# **Guide du Projet de Simulation Numérique**

**ECAM 3e**

**2024**



## Contenu

I.	Introduction.....	3
II.	Le facteur chinois .....	4
III.	Cahier des charges.....	7
IV.	Déroulement du projet.....	11
V.	Évaluation.....	14
VI.	Guide du développeur .....	15
	<b>Exemple de lecture de fichier txt .....</b>	<b>17</b>
VII.	Manuel de secours .....	18
VIII.	Exemple de trombinoscope.....	19

# I. Introduction

Pour cette cuvée 2024, nous vous proposons de transformer le robot Lego en facteur chinois !



Le robot EV3 est un robot ludique développé par Lego. Comme tout robot, il est constitué de moteurs, de capteurs et d'un calculateur. Il est, comme tout jouet Lego, modulable, au gré de l'imagination de ses utilisateurs. La programmation se fait nativement avec un langage graphique, inspiré par LabView, et accessible (en théorie) dès l'âge de 10 ans. Toutefois, pour ce projet, vous devrez développer en langage Python sous Visual Studio Code. La documentation nécessaire vous sera fournie pour y parvenir.

## II. Le facteur chinois

Nous vous proposons de **développer en Python un programme qui déterminera la tournée optimale d'un facteur !**

Le problème du "facteur chinois"<sup>1</sup> est un cas d'école en théorie des graphes. Il s'agit d'optimiser la tournée de distribution d'un postier. Pour simplifier, on considérera ici qu'il n'y a pas de rues en sens unique. De plus dans la suite on considérera un cas simplifié du problème dans lequel les distances de chaque rue n'interviendront pas. En conséquence, on modélise l'ensemble des rues que doit emprunter le facteur par un graphe connexe non orienté et non pondéré. On cherche donc le plus court chemin qui passe au moins une fois par chaque arête du graphe et revient à son point de départ. Bien entendu, les arêtes représentent les rues et les sommets les carrefours.

La solution idéale est bien entendu de n'emprunter chaque rue qu'une fois mais cela n'est pas toujours possible.

### 1. Quelques rappels de théorie des graphes

Pour la bonne compréhension du problème, il est important d'introduire la notion de *degré* d'un sommet :

**Définition :** le **degré** d'un sommet est égal au nombre d'arêtes qui lui sont connectées.

Par exemple, dans le graphe suivant le sommet 6 a un degré égal à 3, tandis que le sommet 3 est de degré 4. Or, comme vous le verrez rapidement la parité des degrés des sommets a une importance fondamentale dans la résolution du problème.

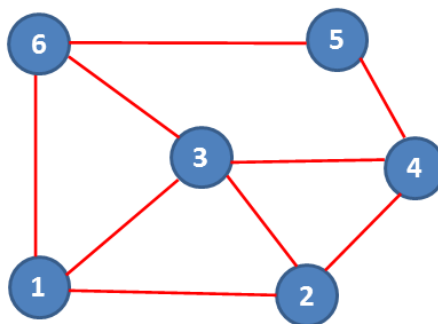


Figure 1

**Définition :** Un graphe est **eulérien** s'il est possible de parcourir avec un crayon toutes ses arêtes sans lever le crayon et sans jamais passer deux fois sur la même arête, et en revenant au point de départ.

**Exercice :** le graphe précédent (figure 1) est-il eulérien ? le graphe suivant (figure 2) est-il eulérien ?

---

<sup>1</sup> Le nom du problème vient du mathématicien chinois Meigu Guan qui l'a étudié en 1962.

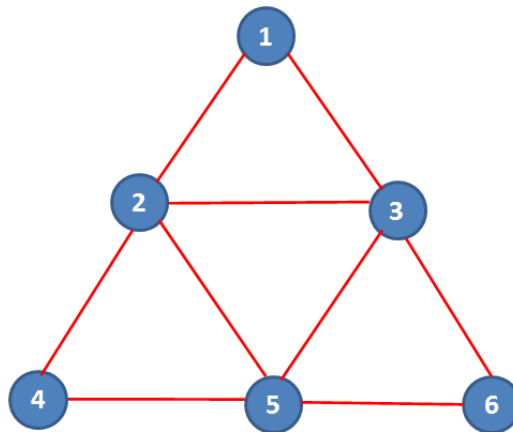


Figure 2

**Propriété** : On montre qu'un graphe est eulérien si et seulement si chaque sommet est de degré pair.

**Exercice** : est-ce le cas du graphe précédent ?

La solution idéale (ne jamais passer 2 fois par la même rue et revenir au point de départ) est donc possible pour un graphe eulérien. Lorsque le graphe n'est pas eulérien, on peut le rendre eulérien en lui ajoutant des arêtes artificielles.

**Définition** : Dans un graphe non orienté, une **chaîne** reliant un sommet  $x$  à un sommet  $y$ , est une suite finie d'arêtes consécutives reliant  $x$  à  $y$  (on parle de chemin pour des graphes orientés).

**Définition** : un graphe non orienté est dit **connexe** s'il existe toujours au moins une chaîne reliant 2 sommets quelconques.

**Définition** : Une arête  $(a, b)$  d'un graphe  $G$  est appelée un **pont** si  $(a, b)$  est l'unique chemin entre les sommets  $a$  et  $b$ .

Ainsi sur l'exemple suivant l'arête  $(3,4)$  est un pont :

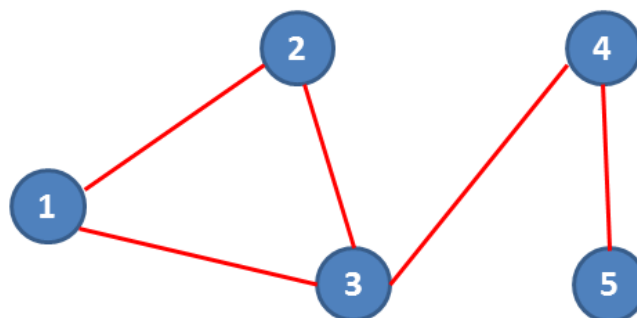


Figure 3

On notera que si on enlève un pont le graphe résultant est non connexe.

Dans la suite,  $n$  désigne le nombre de sommets du graphe initial.

**Définition : la matrice d'adjacence**  $A$  d'un graphe non orienté et non pondéré  $G$  est définie par :  $A(i,j) = 1$  s'il existe une arête entre  $i$  et  $j$  et  $0$  sinon. On pose  $A(i,i)=0$  pour tout  $i$ .

Pour savoir si une arête  $(i,j)$  est un pont, on peut enlever cette arête, calculer la matrice d'adjacence  $A$  du graphe ainsi obtenu, puis la matrice  $S = \sum_{k=0}^{n-1} A^k$ . Si l'élément  $S(i,j)$  est nul, alors cela signifie qu'il n'y a pas de chaîne entre  $i$  et  $j$ , et donc que l'arête  $(i,j)$  était un pont (résultat admis).

## 2. Algorithme de Fleury

Si  $G$  est eulérien, il suffit d'appliquer l'algorithme suivant pour trouver une solution idéale au problème du facteur chinois :

**Algorithme**  $C = \text{Fleury}(G)$

Partir d'un sommet  $x$  (choisi arbitrairement)

Initialiser la chaîne  $C = \{x\}$

**Tant que**  $G$  possède au moins une arête

Prendre  $y$  un successeur de  $x$

**Tant que**  $(x,y)$  est un pont ) ET ( il existe un autre successeur de  $x$ ) alors

Prendre  $y$  successeur de  $x$  suivant

// en sortie  $(x,y)$  est donc une arête qui n'est un pont que s'il n'y a pas d'autre choix  
ajouter  $y$  à la fin de la chaîne  $C$

détruire l'arête  $(x,y)$  de  $G$

$x = y$

retourner  $C$

On considèrera dans la suite avoir à faire à des graphes de degrés pair, donc eulériens.

On notera qu'aucune notion de distance n'apparaît dans l'algorithme de Fleury, pourquoi ?

Cela justifie bien que les graphes considérés soient non pondérés.

# III. Cahier des charges

On classe les exigences du client par niveau croissant. Vous pouvez travailler indépendamment sur les niveaux 0 et 2 et donc vous partager le travail efficacement.

## Niveau 0 :

**En entrée :** 1 fichier txt nommé *facteurN\_i.txt* avec N= nombre de sommets et i un nombre.

- La première ligne est un entier spécifiant le nombre de sommets.
- Les lignes suivantes contiennent un couple représentant une arête (peu importe l'ordre puisque le graphe est non orienté).

Par exemple, le graphe de la figure 2 est représenté par le fichier ci-dessous (on notera que l'arête (2,1) n'apparaît pas mais est implicite puisque l'arête (1,2) est donnée).

6
1 2
1 3
2 3
2 4
2 5
3 5
3 6
4 5
5 6

Vous devez écrire un programme qui lira le fichier et construira la matrice d'adjacence du graphe. Vous trouverez en annexe un exemple de lecture d'un fichier *txt*. Les intervenants seront également à votre disposition pour vous aider.

## Niveau 1 :

Implémentez l'algorithme de Fleury et affichez la chaîne trouvée.

## Niveaux > 1 :

Pour la deuxième partie concernant le robot, on représentera au sol une carte correspondant au graphe.

Les rues seront représentées par un adhésif de couleur orange, les carrefours (sommets) par un adhésif noir. La largeur de la route noire sera de 5 cm. Les marques noires auront la forme d'un rectangle de largeur et de longueur d'environ 5 cm. Pour simplifier, les virages à effectuer au niveau des carrefours seront tous à 90° +/- 2°. Par contre les virages le long des rues seront variables mais d'angle aigu. Ainsi

la carte ressemblera à la figure 5 ci-dessous sans les numéros rouges. On prendra des exemples relativement simples de manière à éviter des intersections "factices" de rues.

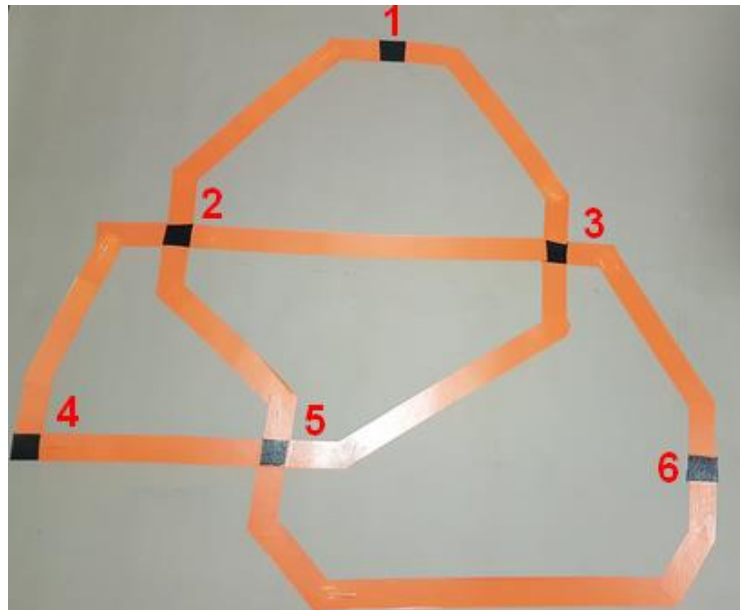


Figure 4

La carte sera modélisée par 1 matrice appelée Orientation, telle que Orientation(i,j) désigne le point cardinal (de type caractère = 'E', 'O', 'S' ou 'N' pour Est, Ouest, Sud ou Nord) indiquant la direction à prendre pour aller du carrefour i vers le carrefour j. S'il n'y a pas d'arête entre i et j, la valeur sera X.

Ainsi, sur l'exemple précédent, on aurait :

$$Orientation = \begin{pmatrix} X & O & E & X & X & X \\ N & X & E & O & S & X \\ N & O & X & X & S & E \\ X & N & X & X & E & X \\ X & N & E & O & X & S \\ X & X & N & X & S & X \end{pmatrix}$$

Cette matrice sera définie dans un fichier *Orientation.txt*, ligne par ligne, le séparateur étant un espace.



## **Niveau 2 :**

Vous pouvez orienter votre robot comme vous le souhaitez au départ.

Vous devez construire et programmer un robot qui puisse aller d'un carrefour i à un carrefour j tel que l'arête (i,j) existe, en suivant correctement la rue associée à l'arête (i,j). Les valeurs de i et j seront à lire dans un fichier *carrefours.txt*.

Bonus : sélectionner les carrefours à l'aide des touches de la brique intelligente.

Vous disposerez tout au long du projet d'une "carte" test en salle Amorgos (S209, département informatique). Il vous appartiendra de faire des tests suffisamment tôt pour valider votre modèle de robot.

Vous disposez dans votre kit<sup>2</sup>, entre autres, de 2 gros moteurs, pouvant actionner des roues, d'un système de "roue folle", d'un capteur de couleurs devant être très proche du sol (3 cm max) et d'un capteur gyroscopique vous permettant de contrôler l'angle de vos virages. Les caractéristiques techniques de ces différents éléments se trouvent facilement sur Internet.

Vous avez toute latitude pour la construction de votre modèle. En cas de difficultés, l'intervenant se tient à votre disposition pour vous guider dans vos choix.

## **Niveau 3 :**

Orientez votre robot vers le Nord (il sera indiqué sur la carte au sol).

Vous devez effectuer sur une carte du type de l'exemple précédent un parcours de ce type :

0. Partir du carrefour 1. Tournez à droite et suivre la rue jusqu'au carrefour 3.
1. Tourner à gauche et suivre la rue jusqu'au carrefour 6.
2. Continuer tout droit et suivre la rue jusqu'au carrefour 5.
3. Tournez à droite et suivre la rue jusqu'au carrefour 3.
4. Tourner à gauche et suivre la rue jusqu'au carrefour 2.

Ce parcours vous sera donné sous la forme d'un fichier "*trajet.txt*" avec ce codage :

- D pour "tourner à droite"
- G pour "tourner à gauche"
- C pour "continuer tout droit"

Ces symboles seront écrits sur une seule ligne. Sur l'exemple précédent cela donnerait :

D G C D G

On tiendra compte du temps de parcours, mais il faudra surtout essayer de ne pas avoir à intervenir manuellement.

---

<sup>2</sup> Il y a 2 familles de kits EV3 : une version grand public et une version éducation. Nous avons complété ces 2 versions de manière à les rapprocher au mieux et rendre ce projet équitable.

**Niveau 4** : Effectuer un parcours à l'aide d'une carte.

Soit un chemin représenté par une liste de carrefours telle que 1-3-6-5-3-2-4-5-2-1 et défini dans un fichier "*chemin.txt*". Grâce à la matrice Orientation, en déduire le parcours à effectuer sous forme d'une liste d'actions décrites comme au niveau 3.

**Niveau 5** : Faire suivre au robot le chemin optimal trouvé par Fleury.

Cette dernière étape consiste à "rassembler" les niveaux précédents.

**Bonus** : un obstacle peut se trouver sur votre trajet. Vous devez le détecter, vous arrêter, émettre un bip puis repartir après avoir enlevé à la main l'obstacle.

# IV. Déroulement du projet

## Le déroulement d'une séance

Au début de chaque séance, chaque élève doit :

- émarger sur la feuille de son équipe épinglée sur le panneau en liège près du Petit Amphi,
- noter les instructions de la séance figurant sur ce même panneau (horaire de passage avec les différents référents s'il y a lieu).
- Pour chaque équipe, le premier élève arrivé doit indiquer sur la feuille de présence, dans quelle(s) salle(s) se trouve l'équipe.

Les robots seront stockés à l'entrée du département informatique, dans des casiers cadenassés. Le numéro du casier et le code d'accès vous seront communiqués en début de projet. Un élève par équipe sera responsable du kit robotique. Il devra venir chercher le robot en début de séance et l'indiquer sur la feuille *ad hoc*, et ramener le robot en fin de séance. **Ne pas oublier le câble USB !**

## Le contexte du projet

Vous êtes salariés d'une société de conseil en informatique. Un de vos clients (l'ECAM Rennes) a émis un appel d'offre, concrétisé par un cahier des charges, auprès de différents prestataires. Chaque équipe projet appartient à une société différente et elle conçoit une proposition en réponse à cet appel d'offre. En plus des exigences du cahier des charges, vous devez tenir compte d'autres contraintes du client :

- Date de remise de votre proposition
- Date de fin du projet

... et de votre propre société :

- Le respect des méthodes de travail de votre société.
- Le nombre de collaborateurs composant l'équipe projet.

Le commercial de votre société a réussi à avoir une idée du budget dont dispose le client. Celui-ci est surtout sensible au respect du délai et du cahier des charges. Votre société est très attachée à la réussite de ce projet dont découle l'obtention d'autres projets.

Chaque proposition est validée par la direction de chaque société. Elle peut être amenée à vous demander de la modifier avant la remise au client.

## Quelques éléments peuvent évoluer en cours de projet.

Un cahier des charges vous est remis dans ce guide. À partir de celui-ci, vous devez remettre une proposition financière à Christophe Boisramé, en début de séance 3

## Planification

Séance	Interventions de l'école	Modalités	Livrables attendus par l'école	Encadrants (*)
Présentation du sujet+ Gestion de Projet		Toute la promo		Tous
Séance 1	Notions sur Python Objet. Intervention GeP avec les chefs de projet.	Avec les responsables développement. 1h avec CB avec l'ensemble des chefs de projet	Trombinoscope de l'équipe avec répartition des rôles. Identification des tâches à réaliser. Affectation de ces tâches à des membres de l'équipe.	NG + JML + CB
Séance 2	Séance non encadrée			
Séance 3	Revue de la proposition financière Point développement.	30 mns par équipe au complet	Proposition financière envoyée avant la séance. Suivi de projet à jour. Diagramme des classes	CB + NG + JML
Séance 4	Séance non encadrée			
Séance 5	Revue de projet. Point développement.	30 mn par équipe pour le GeP	Suivi de projet à jour. Niveaux 1 et 2 terminés.	CB + NG + JML
Séance 6	Séance non encadrée			
Séance 7	Point développement.		Niveau 3 et 4 terminé.	NG + JML
Séance 8		1 h avec l'ensemble des chefs de projet en début de séance.	Niveau 5 terminé	NG + JML + CB
Validation	<b>Recette</b>		Tous les livrables de la GeP à jour. Formulaire audit de GeP complété.	Tous

(\*) NG = Noussaïba GASMI, CB = Christophe BOISRAMÉ, JML = Jean-Marc LAFERTÉ

## La gestion de projet

Christophe Boisramé interviendra :

- Lors de la présentation du projet avec toute la promotion)
- En début de séance 2 à distance ou présentiel auprès des CdP
- En séance 3 pour une rencontre individuelle d'environ 30 mn avec chaque équipe complète (présentation de la proposition financière)
- En séance 5 pour une rencontre individuelle avec chaque équipe complète (suivi du projet)
- En début de séance 8 à distance ou présentiel auprès des CdP
- Lors de l'audit final

Il consultera régulièrement le suivi de projet déposé sur l'espace partagé (drive).

Chaque équipe disposera d'un suivi de projet à jour régulièrement ainsi qu'à la fin du projet (toutes les charges remplies), lors de l'audit.

Pour la séance de rencontre avec les équipes, les élèves le rejoignent dans une salle prédéfinie, à la fréquence d'un groupe toutes les 35mn. Excepté le 1<sup>er</sup> groupe, qui se présente à l'heure prévue, chaque groupe est prévenu par le groupe précédent qu'il doit se présenter à la réunion de gestion de projet. Cela évite les attentes en cas de retard.

Les créneaux horaires sont mentionnés dans la présentation initiale. Toutes les ressources (documents, tutos) se trouvent sur le moodle.

Le groupe doit être en mesure de projeter tous les documents de gestion de projet lors de la séance (PC portable, clef USB, accès à l'espace partagé).

## Les livrables

### La proposition financière

L'équipe fournira une proposition financière au début de la séance 3.

Elle sera disponible AVANT le début de la séance 3, la date du fichier faisant foi. Les versions ultérieures devront avoir une version différente.

Voir les détails dans la présentation de la Gep.

### Les fichiers de suivi projet

Ce fichier regroupe les données d'avancement du projet. Ce fichier qui évolue tout au long du projet, est mémorisé, avec un nom différent, avant chaque séance de Gestion de Projet.

Il doit donc y avoir un fichier par séance en fin de projet (donc 8 fichiers)

## V. Évaluation

La validation est réalisée en 3 temps décrits ci-dessous donnant lieu à 3 notes. L'ordre de passage sera affiché en début de séance sur le panneau habituel.

1. **Évaluation de la Gestion de projet – Auditeur : Christophe BOISRAMÉ – Durée : 25 mns - élèves : le chef de projet + les équipiers désirant assister à cette recette.**

Lors des séances 3 et 5 une évaluation de votre suivi de projet sera réalisée.

La recette consiste en un audit de la GeP par l'intervenant en GeP, avec le chef de projet et les équipiers désirant y assister. Les points abordés lors de cette recette sont précisés dans la présentation de la GeP (slide « la soutenance pour la GeP »). Cet audit nécessite la communication au préalable de tous les éléments nécessaires. Il s'appuie sur le formulaire « audit » se trouvant dans le fichier de suivi de projet. Ce formulaire devra donc être rempli avant l'audit.

2. **Recette algorithme sur PC – Noussaïba GASMI – Durée : 25 mns – élèves : développeurs**

Les niveaux 0 et 1 seront évalués sur ordinateur et un audit du processus de développement du programme sera réalisé. Votre programme doit fonctionner pour un nombre quelconque de sommets.

3. **Recette Robot - Jean-Marc LAFERTÉ – Durée : 25 mns – élèves : reste de l'équipe**

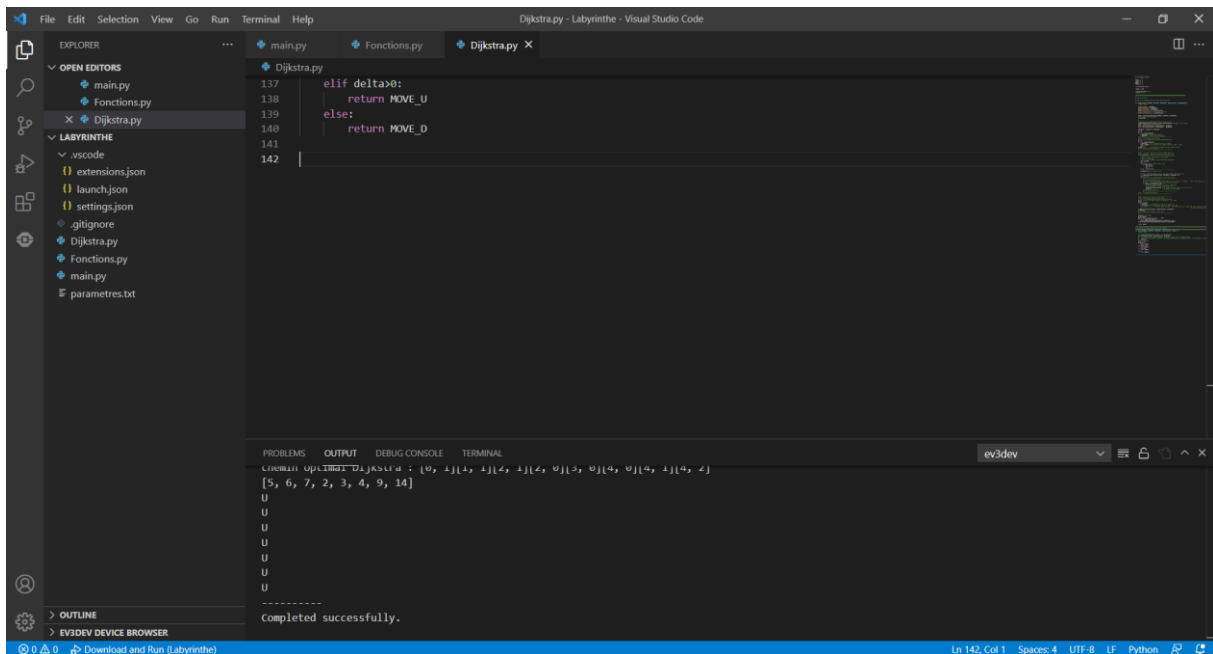
Les niveaux  $\geq 2$  seront testés durant cette partie. La carte sera représentée au sol comme expliqué dans ce document. La recette aura lieu dans la salle Amorgos (Département Informatique). Les fichiers txt seront chargés via un PC. **Vous devrez venir avec votre robot démarré pour ne pas perdre de temps et avec le câble USB !** On appréciera la navigabilité du robot, sa rapidité, sa fiabilité et à un degré moindre ... son design !

## VI. Guide du développeur

**Installation de Visual Studio Code (VS Code) :**

<https://education.lego.com/en-us/product-resources/mindstorms-ev3/teacher-resources/python-for-ev3>

**Vue de l'interface de VSCode :**



À gauche, on a l'explorateur de projets avec les différents fichiers du projet ouvert (si cette vue n'est pas activée par défaut, aller dans le menu View/Explorer). À droite en haut, on trouve l'éditeur avec le code en cours. Enfin en bas à droite, on a la console où peuvent s'afficher les résultats (print).

**Création d'un projet sous VS Code<sup>3</sup> :**



Pour créer un nouveau projet sous VS Code, cliquer sur l'icône de la barre de boutons à gauche.

<sup>3</sup> NB : si vous utilisez un PC de l'école, il faut se rendre dans extension > puis rechercher EV3 et cliquer sur installer.

Pour ajouter un fichier, positionnez la souris dans le cadre de gauche au niveau du nom du projet puis cliquer sur l'icône "New File" (avec un +). Pensez à changer le nom du fichier avec la bonne extension !

Pour exécuter un programme, reliez votre robot au PC avec le câble USB puis aller dans le menu Run.

Vous pouvez exécuter en mode debug, ce qui vous permet notamment de voir affichés sur votre ordinateur tous les print, mais il faut que le robot soit connecté au PC, ce qui n'est pas très pratique ! Si c'est impossible ou si vous êtes sûr de votre code, vous pouvez exécuter en mode normal. La première fois il vous faudra sans doute demander à se connecter au robot (voir message). Attention le robot va bouger rapidement mais vous pouvez l'arrêter en cliquant sur le bouton en haut à gauche de la brique. L'exécution provoque le chargement du programme sur la brique. Vous pouvez ensuite exécuter votre programme à partir de la brique en allant dans le menu "File Browser", puis sélectionner le bon projet puis le module main contenant le programme principal (nous vous conseillons de l'appeler main).

### Éléments de langage

Attention pyBricks est basé sur MicroPython, un langage adapté aux systèmes embarqués. La syntaxe est bien du Python mais tous les modules que vous trouvez habituellement ne sont pas utilisables ici. Ainsi de numpy. Nous travaillerons donc avec des listes plutôt que des tableaux.

De nombreux import vous seront nécessaires. En voici une liste non exhaustive :

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                  InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile
```

Initialisation du robot avec 2 moteurs :

```
ev3 = EV3Brick()
# Initialize the motors.
left_motor = Motor(Port.B) # vérifier vos branchements !
right_motor = Motor(Port.C)
# Initialize the drive base.
robot = DriveBase(left_motor, right_motor, wheel_diameter = 43,
                  axle_track=138) # toutes les distances sont en mm
robot.settings(220, 100) # speed and acceleration
ev3.speaker.beep() # émet un son pour s'assurer qu'il fonctionne
```



Mouvements basiques (programmation de type "objet") :

```
robot.turn(45) # angle en degrés (positif ou négatif)
                # NB : raisonne dans le sens horaire !
robot.straight(100) # distance en mm (si négatif, recule)
robot.stop()
```

Capteurs (exemple avec le capteur à ultrasons) :

```
obstacle_sensor = UltrasonicSensor(Port.S4)
while obstacle_sensor.distance() > 300:
    wait(10)
```

Afficher du texte sur l'écran LCD (print ne fonctionne pas sur le robot) :

```
ev3.screen.draw_text(50, 50, "hello") # positionné en (50,50)
```

Vous trouverez une documentation détaillée sur <https://pybricks.com> ou plus synthétique sur [http://numerique.ostralo.net/robotLego\\_python](http://numerique.ostralo.net/robotLego_python) (oubliez les fautes d'orthographe !).

### Exemple de lecture de fichier txt

Exemple de code lisant un fichier *test1.txt* de ce type :

```
6 4
5 4 11 5 11 8
7 2 18 11 4 5
```

```
file = open("test1.txt", "r")
lignes = file.readlines()
for ligne in lignes:
    print(ligne)
    liste = ligne.split(" ")
    print(liste)
    for x in liste:
        n = int(x)
        print(n)
file.close()
```

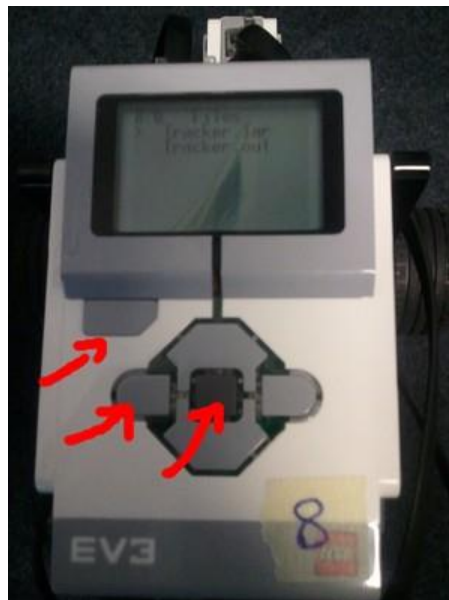
## VII. Manuel de secours

Si l'ordinateur ne trouve pas le robot, débranchez puis rebranchez le câble USB.

Rappel : vous pouvez arrêter brutalement le programme en cliquant sur le bouton en haut à gauche.








Si le robot ne "répond" plus :

- reboot robot en appuyant simultanément sur les boutons suivants : haut gauche, central et gauche plusieurs secondes ...
- En dernier recours : enlever la batterie et la remettre.



## VIII. Exemple de trombinoscope

**Equipe Projet A1**



**Maxime Veirier**  
Scrum Master

**Samuel Toukio**  
Responsable Robot

**Adrien Itsweire**  
Assistant Développement

**Gwenole Le Calonnec**  
Responsable Développement

**Frank Filoda**  
Responsable Etudes

