

Mode d'emploi | Robot JetBot | Projet CoDev 2021

Table des matières

1. Fonctionnement global	1
2. Liens utiles	1
3. Installation du logiciel sur la plateforme	2
4. Connexion à la plateforme	2
5. Utilisation de la plateforme	3
6. Utilisation du code exemple et développé (projet CoDev 2021)	4

1. Fonctionnement global

Jetbot est une plateforme développée par NVIDIA à laquelle vous pouvez accéder via une connexion wifi pour pouvoir programmer différents types d'algorithmes.

Nous allons voir ici :

- L'installation du logiciel sur la plateforme
- La connexion à la plateforme
- L'utilisation de la plateforme
- L'utilisation du code qui a été développé

Nous n'allons pas revenir sur la partie hardware/montage de la plateforme, pour plus d'informations sur cette partie, le mode d'emploi des différents branchements est donné dans le lien utile 4 ci-dessous.

Attention !! Il y a cependant deux trois précautions à respecter :

- Le branchement et le débranchement des composants (caméra, moteurs, écran PIOled...) sur la carte Jetson Nano doivent se faire lorsque celle-ci n'est pas alimentée
- L'alimentation doit se faire en 5V via le connecteur jack (voir la position des cavaliers dans le document sur les branchements disponible sur le lien 4)
- Pour éteindre le système il est recommandé de la faire via la ligne de commande "sudo shutdown now" (détails ci-après)

2. Liens utiles

1. (Installation du logiciel)
<https://github.com/NVIDIA-AI-IOT/jetbot/wiki/software-setup>
2. (Ressources données par NVIDIA)
<https://jetbot.org/master/index.html>
<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit>
3. (Montage et autres ressources concernant la plateforme en bas de l'article)
<https://www.seeedstudio.com/Seeedstudio-JetBot-Smart-Car-Kit-p-4055.html>

4. (Configuration de la connexion wifi)
http://jetbot.org/master/software_setup/wifi_setup.html
5. (Vidéo reprenant les préparations et premiers exemples pour prendre en main le robot)
<https://www.youtube.com/watch?v=9Wv9A6C6U5w>
6. (Notebook avec les codes exemples)
<https://github.com/NVIDIA-AI-IOT/jetbot/tree/master/notebooks>

3. Installation du logiciel sur la plateforme

(Cette étape est déjà réalisée sur la plateforme livrée à la fin du projet CoDev 22 - 2021)

Plusieurs versions du logiciel sont disponibles pour les cartes Jetson Nano, après plusieurs tentatives, nous avons opté pour l'image "jetbot_image_v0p4p0.zip" disponible avec le premier lien utile.

Pour l'installation, il faudra installer les logiciels suivants :

- SD Card Formatter : <https://www.sdcard.org/downloads/formatter/>
- Etcher balena : <https://www.balena.io/etcher/>

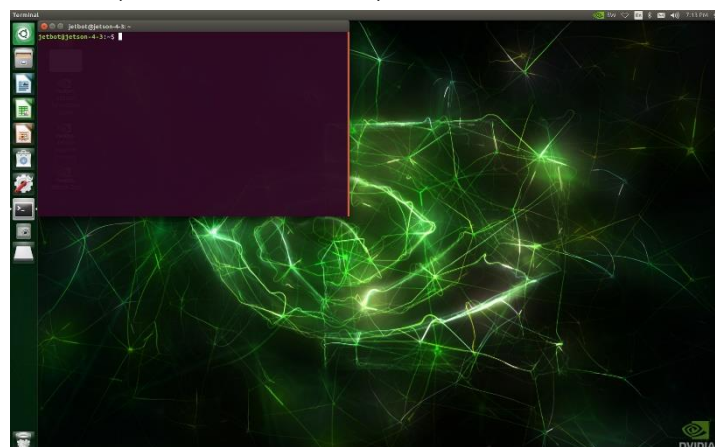
Installation

- Première étape : Formater la carte SD (64Go) avec le premier logiciel
- Deuxième étape : Flasher l'image sur la carte sd, pour cela il faut au préalable télécharger l'image (7Go!) et utiliser le second logiciel sans dézipper le fichier. Un nombre important de disques amovibles vont être créé, c'est normal.
- Troisième étape : éjecter simplement la carte SD (un seul des disques suffit normalement)

4. Connexion à la plateforme

(Vous retrouvez une partie de ces explications dans le lien utile 5)

- Pour se connecter à la plateforme, il faut configurer la connexion wifi de celle-ci, il faudra ensuite se connecter avec son PC au même réseau wifi (attention il peut y avoir des problèmes d'accès pour certains réseaux 'publics', dans notre cas, nous avons utilisé une connexion via un smartphone : hotspot).
- Afin de connecter la plateforme au wifi, nous avons branché un clavier, une souris et un écran via les ports HDMI et USB de la plateforme, ensuite nous venons brancher l'alimentation, ce qui démarrera le robot (détails dans les documents donnés dans le lien 4).
- On arrive alors sur un environnement ubuntu à partir duquel nous pouvons nous connecter à internet (détails dans le lien 5).



- Une fois la connexion WiFi établie, vous pouvez éteindre la plateforme via le menu ubuntu ou avec le code suivant : "sudo shutdown now".
- Ensuite, vous devez débrancher le clavier, la souris et l'écran puis redémarrez le robot en le branchant l'alimentation de nouveau.
- Après une minute, l'écran PIOled doit s'allumer en affichant l'adresse IP du robot, l'espace de stockage disponible et l'utilisation de la mémoire.
- Ouvrez alors un navigateur web et entrez l'adresse <http://adresseIP:8888> pour vous connecter au robot, une fenêtre Jupyter Lab doit s'afficher, il suffit alors de taper le mot de passe 'jetbot' et vous serez connecté au robot.

Remarques :

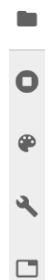
- Si un mot de passe est demandé, ce sera nécessairement "jetbot"
- Une alternative à cette solution de connexion est possible (voir le lien 5)

Configuration

A l'heure actuelle, nous n'avons pas trouvé d'autre moyen pour configurer la connexion WiFi que d'utiliser à nouveau un écran, une souris et un clavier sur la carte Jetson Nano

5. Utilisation de la plateforme

Jupyter Lab se découpe en cinq menus :

- 
- L'explorateur des fichiers qui donne accès à tous les documents présents sur le robot.
 - Un gestionnaire des Kernels (processus permettant de faire le lien entre le code et les différents éléments de l'interface : widgets...)
 - Une recherche de commandes référençant toutes les commandes utilisables
 - Notebook Tools mettant à disposition tous les outils de développement pour utiliser le notebook
 - Open tabs : mettant tout simplement en évidence les différents tabs ouvertes

Les deux principaux menus qui seront utilisés sont l'explorateur des fichiers et le gestionnaire des Kernels.

Jupyter Lab permet de lire une partie des fichiers présents (photos, texte, code python...) en y accédant via l'explorateur de fichiers. Vous pourrez ainsi créer des fichiers notebook python (.ipynb) et les exécuter. Vous pourrez aussi exécuter l'invite de commande (terminal) pour installer des bibliothèques, des répertoires GitHub ou éteindre le robot par exemple avec la commande "sudo shutdown now".

Remarques :

- Le mot sudo devant une commande indique que celle-ci sera fait en tant qu'administrateur (le mot de passe "jetbot" vous sera demander la première fois). Il est parfois nécessaire de le mettre pour installer des bibliothèques ou faire des mises à jour.
- Nous vous invitons à explorer l'interface pour comprendre les différentes interactions et commandes possibles

- Il est préférable de stopper le Kernel d'un fichier qui n'est pas utilisé pour éviter des problèmes de compatibilité entre les différents codes et limiter l'utilisation de la mémoire du robot qui reste limitée.

6. Utilisation du code exemple et développé (projet CoDev 2021)

Commençons par le code exemple, un répertoire GitHub est disponible (lien 6) avec plusieurs exemples d'utilisation pour le robot. On y retrouve les dossiers suivants :

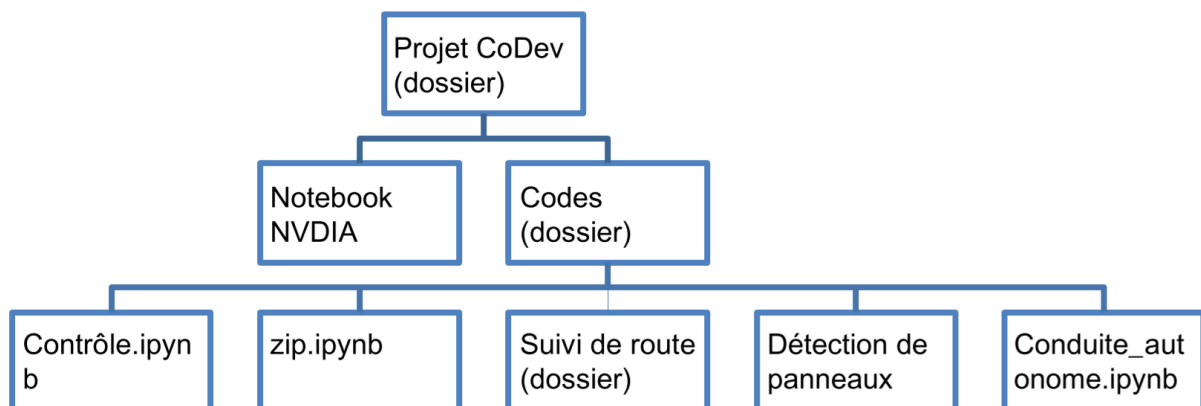
- `basic_motion` : permettant d'appréhender les différentes commandes et bibliothèques disponibles pour contrôler le robot (contrôle des roues, vitesse, retour caméra...)
- `collision_avoidance` : donnant les étapes pour d'une part capturer des photos avec le robot, puis d'autre part de les analyser pour créer un modèle à l'aide d'un réseau de neurones et de l'utiliser pour différencier une situation où une route est bloquée ou libre.
- `object_following` : ce dossier contient les codes permettant de faire en sorte que le robot puisse suivre un objet (parmi en ensemble d'objets connus) (non utilisé dans le cadre du projet CoDev 22 - 2021)
- `road_following` : ici on nous explique comment faire en sorte que le robot suive une route
- `teleoperation` : ce dossier reprend une partie de `collision_avoidance` pour permettre de prendre des photos et commander le robot à l'aide d'une manette (du type xbox)

Les codes sont très bien commentés et assez intuitifs, il faudra cependant être attentif aux bibliothèques et modules nécessaires pour les faire fonctionner.

Passons maintenant au code que nous avons développé/modifié.

Nous vous invitons à tester les codes donnés par NVIDIA au préalable afin de tester des cas simples dont le temps des calculs reste faible.

Les différents notebooks sont placés dans le dossier Projet Codev qui possède l'architecture suivante :



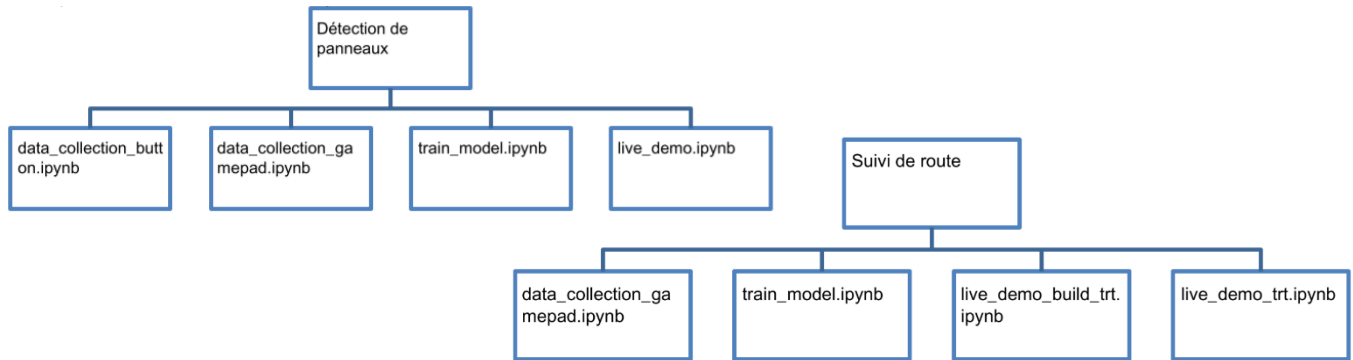
Où le Notebook NVIDIA regroupe les exemples expliqués précédemment.

Le fichier `contrôle.ipynb` est un résumé des différentes commandes possibles pour contrôler le robot (avancer, reculer...).

Le fichier zip.ipynb reprend deux commandes simples zipper et dézipper mais qui sont indispensables dans l'utilisation de la plateforme car pour télécharger ou téléverser un fichier, il est nécessaire de le compresser.

Le fichier conduite_autonome.ipynb n'a pas pu être abordé dans notre projet, il avait pour but d'aborder le développement d'un comportement autonome avec différentes décisions à prendre.

Les dossiers "Suivi de route" et "Détection de panneaux" ont la structure suivante :



Pour la détection de panneaux :

La suite logique d'utilisation est la suivante :

- data_collection_button.ipynb OU data_collection_gamepad.ipynb qui permettent de faire des photos des obstacles, panneaux, piéton et d'une voie de circulation libre. Pour l'utiliser il suffit de suivre l'enchaînement des parties que nous avons commenté directement sur le Notebook.
 - Création des dossiers pour stocker les photos
 - Création des différents boutons
 - Affichage des boutons
 - Recadrage des photos prises pour les panneaux vitesses et directions
- Train_model.ipynb, dans ce fichier, on vient créer trois modèles pour détecter les différents panneaux et obstacles/situations
- Live_demo.ipynb qui doit permettre de mettre en place cette détection avec l'image reçue par la caméra.

Vous trouverez davantage d'explications spécifiques aux différentes utilisations dans les différents notebooks.

Pour le suivi de route :

Les codes sont ceux développés par NVIDIA, les résultats (modèles) ont été laissés (sur le robot) pour que le suivi de route soit opérationnel.

L'enchaînement des notebooks est le suivant :

- Data_collection_gamepad.ipynb : récupération des données (photos de la route et des directions à prendre à chaque fois)
- Train_model.ipynb : création du modèle grâce au réseau de neurones
- Live_demo_build_trt.ipynb : création d'un nouveau modèle basé sur l'ancien permettant d'améliorer la trajectoire pour rester sur la route
- Live_demo.ipynb : test du modèle en réalisant le suivi de route.