



IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

Cécile Perrette
Mélissa Durand-Flon
FI2A
Promo 2019

Tutoriels PDR TurtleBot

Sommaire

| | |
|--|----|
| Tutoriel : Créer un workspace | 3 |
| Tutoriel : Créer un package | 4 |
| Tutoriel : Lancer ROS | 5 |
| Tutoriel : Mettre à jour son projet | 6 |
| Tutoriel : Créer un publisher et un subscriber | 7 |
| Tutoriel : Exécuter un fichier .launch ou un fichier .cpp | 9 |
| Tutoriel : Connecter le robot à l'ordinateur | 10 |
| Tutoriel : Lancer le laser | 11 |
| Tutoriel : Faire avancer le robot grâce à la commande teleop | 12 |
| Tutoriel : Faire avancer le robot avec un package | 13 |
| Tutoriel : Connecter la caméra Asus au robot (ne fonctionne pas) | 14 |
| Tutoriel : Afficher les images de la webcam à l'écran | 15 |
| Tutoriel : Détecter un QR Code à partir d'une image | 16 |
| Tutoriel : Détecter un QR Code à partir de la webcam | 17 |
| Tutoriel : Détecter un QR Code avec le robot en mouvement | 18 |
| Tutoriel : GitHub | 19 |
| Petite documentation sur ROS | 21 |
| Petite documentation sur OpenCV | 22 |

Tutoriel : Créer un workspace

Source : http://wiki.ros.org/catkin/Tutorials/create_a_workspace

Objectif : Créer un workspace

Tutoriel :

- Après installation du catkin (notre dossier catkin_ws se trouve dans le dossier cp_pdr dans home), dans le terminal lancer les commandes :

```
$ mkdir -p ~/cp_pdr/catkin_ws/src  
$ cd ~/cp_pdr/catkin_ws/  
$ catkin_make
```

Le workspace est créé.

- Il faut ensuite mettre à jour le setup avec les nouveaux dossiers avec la commande :

```
$ source devel/setup.bash
```

Votre workspace est prêt à être utilisé, prochaine étape : créer votre premier package !

Tutoriel : Créer un package

Source : <http://wiki.ros.org/catkin/Tutorials/CreatingPackage>

Objectif : Créer un package nommé `beginner_tutorials` dans le dossier `catkin_ws/src`

Prérequis :

- Avoir créé un workspace

Tutoriel :

- Dans le terminal, taper les commandes suivantes :

```
$ cd ~/cp_pdr/catkin_ws/src  
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

Un dossier `beginner_tutorials` doit maintenant être dans le dossier `catkin_ws/src`, votre package est prêt !

Tutoriel : Lancer ROS

Source : <http://learn.turtlebot.com/2015/02/01/6/>

Objectif : lancer ROS

Tutoriel :

- Dans un nouveau terminal, lancer la commande :

```
$ roscore
```

ROS est correctement lancé si vous voyez :

```
started core service [/rosout]
```

Vous êtes sûr de pouvoir utiliser ROS !

Tutoriel : Mettre à jour son projet

Objectif : Mettre à jour son projet

Tutoriel :

- Pour mettre à jour son projet dès que des modifications ont été faites dans le dossier catkin_ws, dans un terminal il faut lancer les commandes :

```
$ cd ~/cp_pdr/catkin_ws/  
$ catkin_make  
$ source devel/setup.bash
```

Vos modifications ont été prises en compte !

Tutoriel : Créer un publisher et un subscriber

Source : <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29>

Objectif : Créer un publisher qui envoie à intervalles réguliers des « Hello world » et un subscriber qui les reçoit. Utilité : envoyer des ordres au robot qui va les écouter et les transmettre aux nœuds concernés.

Prérequis :

- Avoir créé un workspace
- Avoir créé un package beginner_tutorials dans le dossier catkin_ws/src

Tutoriel :

- Créer un fichier talker.cpp dans le dossier catkin_ws/src/beginner_tutorial/src
- Copier le contenu du lien ci-dessous dans le fichier talker.cpp

https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/talker/talker.cpp

- Créer un fichier listener.cpp dans le dossier catkin_ws/src/beginner_tutorial/src
- Copier le contenu du lien ci-dessous dans le fichier listener.cpp

https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/listener/listener.cpp

- A la fin du fichier catkin_ws/src/beginner_tutorials/CMakeLists.txt, ajouter le code suivant :

```
add_executable(talker src/talker.cpp)
target_link_libraries(talker ${catkin_LIBRARIES})
add_dependencies(talker beginner_tutorials_generate_messages_cpp)

add_executable(listener src/listener.cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
add_dependencies(listener beginner_tutorials_generate_messages_cpp)
```

- Ouvrir un nouveau terminal, lancer les commandes suivantes :

```
$ cd ~/cp_pdr/catkin_ws
$ catkin_make
$ source ./devel/setup.bash
$ roslaunch beginner_tutorials talker
```

Des « Hello world » sont envoyés.

```
[ 16:24:12 ]$ roslaunch pdr talker
[INFO] [1522851872.569399543]: hello world 0
[INFO] [1522851872.669647139]: hello world 1
[INFO] [1522851872.769606403]: hello world 2
[INFO] [1522851872.869640968]: hello world 3
[INFO] [1522851872.969671058]: hello world 4
[INFO] [1522851873.069905566]: hello world 5
[INFO] [1522851873.169409117]: hello world 6
[INFO] [1522851873.269795442]: hello world 7
[INFO] [1522851873.369627776]: hello world 8
[INFO] [1522851873.469768319]: hello world 9
[INFO] [1522851873.569347535]: hello world 10
```

- Ouvrir un second terminal, lancer les commandes suivantes :

```
$ cd ~/cp_pdr/catkin_ws
$ catkin_make
$ source ./devel/setup.bash
$ roslaunch beginner_tutorials listener
```

Les « Hello world » sont reçus.

```
[ 16:24:26 ]$ roslaunch pdr listener
[INFO] [1522851872.870936840]: I heard: [hello world 3]
[INFO] [1522851872.970380377]: I heard: [hello world 4]
[INFO] [1522851873.070744389]: I heard: [hello world 5]
[INFO] [1522851873.170127161]: I heard: [hello world 6]
[INFO] [1522851873.270479622]: I heard: [hello world 7]
[INFO] [1522851873.370599090]: I heard: [hello world 8]
[INFO] [1522851873.470471938]: I heard: [hello world 9]
[INFO] [1522851873.569916888]: I heard: [hello world 10]
```

Vous avez envoyé et reçu vos premiers messages !

Tutoriel : Exécuter un fichier .launch ou un fichier .cpp

Objectif : Exécuter un fichier .launch, ou un fichier .cpp

Prérequis :

- Avoir mis à jour son projet

Tutoriel :

- Pour exécuter un launchfile, dans le terminal, lancer la commande :

```
$ roslaunch nomDuPackage nomDuFichier.launch
```

- Pour exécuter un fichier C++, dans le terminal, lancer la commande :

```
$ rosrune nomDuPackage nomDuFichier
```

Tutoriel : Connecter le robot à l'ordinateur

Source : http://wiki.ros.org/turtlebot_bringup/Tutorials/indigo/TurtleBot%20Bringup

Objectif : Connecter le robot à l'ordinateur avec le minimal bring up

Tutoriel :

- Brancher le robot à l'ordinateur avec un cable usb male/femelle
- Allumer le robot
- Dans un terminal, écrire la commande :

```
$ roslaunch turtlebot_bringup minimal.launch
```

Le robot émet un petit bruit si la commande a fonctionné.

Votre robot est connecté !

- Pour déconnecter le robot, faire un ctrl+c dans le terminal

Par la suite, le minimal bring up pourra éventuellement être directement intégré à un fichier .launch

Tutoriel : Lancer le laser

Objectif : lancer le laser

Tutoriel :

- Avoir branché le laser à l'ordinateur
- Dans un terminal, écrire la commande :

```
$ rosrun urg_node urg_node
```

Le laser est lancé !

Tutoriel : Faire avancer le robot grâce à la commande teleop

Source : <http://learn.turtlebot.com/2015/02/01/6/>

Objectif : Faire avancer le robot grâce à la commande teleop

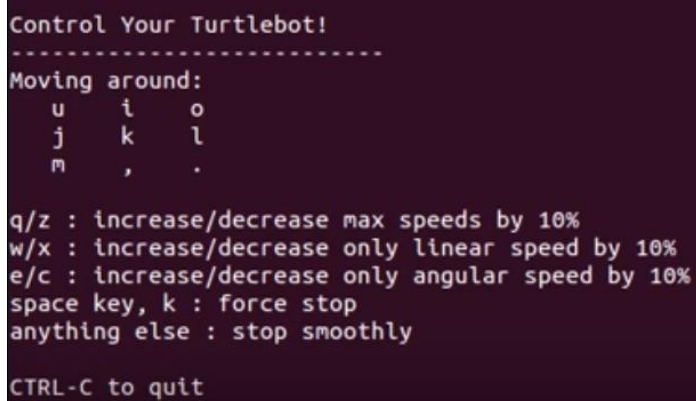
Prérequis :

- Avoir connecté le robot à l'ordinateur

Tutoriel :

- Dans un **nouveau** terminal, lancer la commande :

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```



```
Control Your Turtlebot!
-----
Moving around:
  u   i   o
  j   k   l
  m   ,   .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit
```

Pour faire bouger le robot, il suffit alors d'utiliser les touches du clavier comme par exemple i, o, u...

Tutoriel : Faire avancer le robot avec un package

Objectif : créer un premier package simple pour faire avancer le robot à une distance prédéterminée

Prérequis :

- Avoir récupéré sur le Git <https://github.com/cecileperrette/pdr.git> le projet pdr qui contient notamment le package qrmove
- Avoir mis à jour son projet
- Avoir branché le robot à l'ordinateur et l'avoir allumé

Tutoriel :

Le package qrmove se compose d'un répertoire src, d'un répertoire launch, du fichier CMakeLists.txt et du fichier package.xml. Le répertoire src contient le fichier C++ « main_simplemove.cpp » pour déplacer le robot. Le répertoire launch contient le fichier de configuration « turtle_simplemove.launch ». Le fichier CMakeLists.txt permet de compiler le nœud et le component catkin. Le fichier package.xml permet de définir les dépendances.

- Dans le terminal, lancer la commande :

```
$ roslaunch qrmove turtle_simplemove.launch
```

Le robot se connecte à l'ordinateur, bouge de 2m vers l'avant et de 50cm vers la droite.

Pour terminer le programme proprement, il suffit de faire ctrl+c et le robot se déconnectera.

Tutoriel : Connecter la caméra Asus au robot

(ne fonctionne pas...)

Source : <http://learn.turtlebot.com/2015/02/01/8/> et <https://edu.gaitech.hk/turtlebot/openKinect-turtlebot.html>

Objectif : Connecter la caméra Asus au robot

Tutoriel :

- Vérifier le capteur 3D du TurtleBot en lançant la commande et en confirmant la sortie :

```
$ echo $TURTLEBOT_3D_SENSOR  
# Output: asus_xtion_pro
```

- Définir la valeur par défaut dans .bashrc avec la commande :

```
$ echo "export TURTLEBOT_3D_SENSOR=asus_xtion_pro" >> .bashrc
```

- Installer le package openni :

```
$ cd ~/cp_pdr/catkin_ws  
$ sudo apt-get install ros-kinetic-openni-* ros-kinetic-openni2-*
```

- Fermer tous les terminaux, ouvrir un nouveau terminal et connecter le robot :

```
$ roslaunch turtlebot_bringup minimal.launch
```

- Ouvrir un autre terminal et exécuter :

```
$ roslaunch turtlebot_bringup 3dsensor.launch
```

- Si vous avez des lignes commençant par `process[camera...` la caméra est connectée au robot. Cependant, nous avons toujours eu des erreurs que nous n'avons pas pu résoudre raison pour laquelle nous avons décidé de travailler avec la webcam.
- Pour voir les images de la caméra, lancer la commande :

```
$ rosrunc image_view image_view image:=/camera/rgb/image_raw
```

Tutoriel : Afficher les images de la webcam à l'écran

Objectif : se connecter à la caméra par défaut (en général la webcam du pc) et lire le flux vidéo de la caméra

Prérequis :

- Avoir récupéré sur Git <https://github.com/cecileperrette/pdr.git> le projet pdr qui contient notamment le package qrmove
- Avoir mis à jour son projet

Tutoriel :

Le package qrmove se compose d'un répertoire src, d'un répertoire launch, du fichier CMakeLists.txt et du fichier package.xml. Le répertoire src contient le fichier C++ « main_camconnection.cpp » pour connecter la caméra et lire le flux vidéo avec la librairie OpenCV. Le fichier CMakeLists.txt permet de compiler le nœud et le component catkin. Le fichier package.xml permet de définir les dépendances.

- Dans le terminal, lancer la commande :

```
$ rosrn qrmove camconnection
```

Le flux vidéo de la webcam s'affiche à l'écran !

Tutoriel : Détecter un QR Code à partir d'une image

Objectif : réussir à détecter un QR Code à partir d'une image et l'entourer

Prérequis :

- Avoir récupéré sur Git le projet pdr qui contient notamment le package qrmove
- Avoir mis à jour son projet

Tutoriel :

Le package qrmove se compose d'un répertoire src, d'un répertoire launch, du fichier CMakeLists.txt et du fichier package.xml. Le répertoire src contient le fichier C++ « main_qrreadImage.cpp » pour détecter le QR Code à partir d'une image et l'entourer notamment grâce à la librairie zbar. Le fichier CMakeLists.txt permet de compiler le nœud et le component catkin. Le fichier package.xml permet de définir les dépendances. Dans le dossier catkin_ws/devel/lib/qrmove, il y a l'image « index.png » du QR Code à détecter.

- Dans le terminal, lancer les commandes :

```
$ cd ~/cp_pdr/catkin_ws/devel/lib/qrmove  
$ ./qrreadImage
```

L'image avec le QR Code s'affiche et le QR Code est entouré par un rectangle bleu, il a bien été détecté.

Tutoriel : Détecter un QR Code à partir de la webcam

Objectif : réussir à détecter un QR Code placé devant la webcam et l'entourer

Prérequis :

- Avoir récupéré sur Git <https://github.com/cecileperrette/pdr.git> le projet pdr qui contient notamment le package qrmove
- Avoir lancé ROS
- Avoir mis à jour son projet

Tutoriel :

Le package qrmove se compose d'un répertoire src, d'un répertoire launch, du fichier CMakeLists.txt et du fichier package.xml. Le répertoire src contient le fichier C++ « main_qrCam.cpp » pour détecter le QR Code à partir de la webcam et l'entourer notamment grâce à la librairie zbar. Le fichier CMakeLists.txt permet de compiler le nœud et le component catkin. Le fichier package.xml permet de définir les dépendances.

- Dans le terminal, lancer les commandes :

```
$ cd ~/cp_pdr/catkin_ws/  
$ rosrun qrmove qrCam
```

Le flux de vidéo de la webcam s'affiche à l'écran. Si un QR Code est placé devant la webcam, il est entouré par un rectangle coloré, il a bien été détecté. Dans le terminal, la taille du QR Code s'affiche, ainsi que le mot qui compose le QR Code.

Si le QR Code renvoie au mot « pdr », il est entouré par un rectangle bleu. S'il renvoie au mot « coucou », il est entouré par un rectangle vert. S'il renvoie au mot « robot », il est entouré par un rectangle rouge. Tous les autres QR Codes sont entourés par un rectangle jaune. Les QR Codes « pdr », « coucou » et « robot » sont disponibles sur le GitHub.

Tutoriel : Détecter un QR Code avec le robot en mouvement

Objectif : mettre le robot en mouvement, lorsqu'un QR Code sera repéré devant la webcam, le robot s'arrêtera

Prérequis :

- Avoir récupéré sur Git le projet pdr <https://github.com/cecileperrette/pdr.git> qui contient notamment le package qrmove
- Avoir mis à jour son projet

Tutoriel :

Le package qrmove se compose d'un répertoire src, d'un répertoire launch, du fichier CMakeLists.txt et du fichier package.xml. Le répertoire src contient le fichier C++ « main_fuuusion.cpp » pour détecter le QR Code lorsque le robot est en mouvement. Le répertoire launch contient le fichier « basic.launch » qui permet de lancer ROS avec la commande roscore, de connecter le robot à l'ordinateur avec le minimal bring up et de lancer le laser avec la commande urg_node. Le fichier CMakeLists.txt permet de compiler le nœud et le component catkin. Le fichier package.xml permet de définir les dépendances.

- Dans le terminal, lancer les commandes :

```
$ cd ~/cp_pdr/catkin_ws/  
$ roslaunch qrmove basic.launch
```

Le flux de vidéo de la webcam s'affiche à l'écran. Le robot se met en marche, il fait un mouvement en forme de spirale. Si un QR Code est placé devant la webcam, il est entouré par un rectangle et le robot s'avance jusqu'à ce que la taille du QR Code sur la webcam soit suffisante. Il s'arrête alors et le programme se ferme proprement.

Si le QR Code renvoie au mot « pdr », il est entouré par un rectangle bleu. S'il renvoie au mot « coucou », il est entouré par un rectangle vert. S'il renvoie au mot « robot », il est entouré par un rectangle rouge. Tous les autres QR Codes sont entourés par un rectangle jaune. Les QR Codes « pdr », « coucou » et « robot » sont disponibles sur le GitHub.

Si le robot rencontre un obstacle devant lui, il tourne sur lui-même jusqu'à ce qu'il ne voit plus d'obstacle.

Tutoriel : GitHub

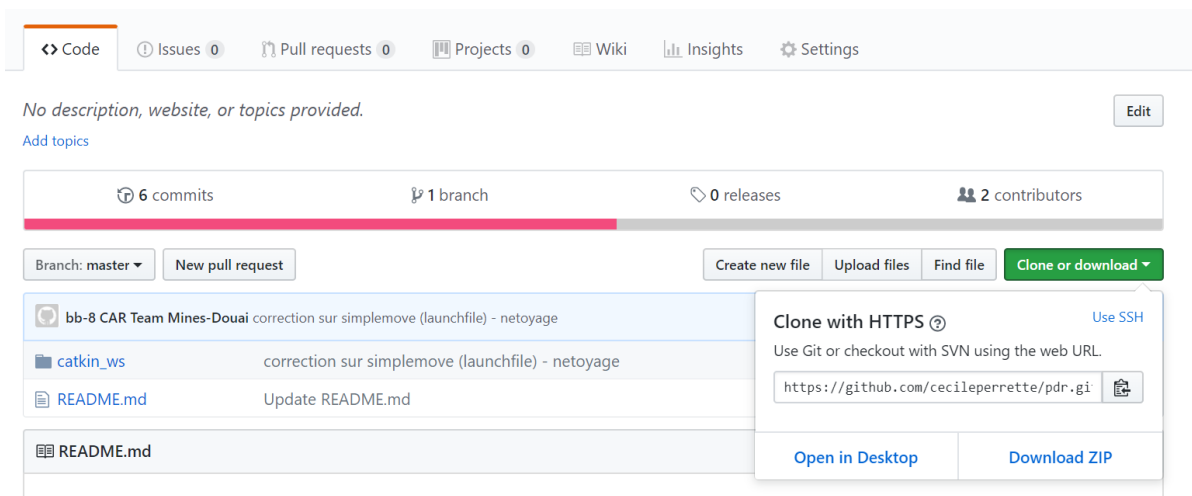
Objectif : se servir du git correctement afin de travailler à plusieurs sur un même projet.

Pour faire ses premiers pas sur GitHub, suivre le tutoriel :

<https://guides.github.com/activities/hello-world/>

Tutoriel :

- Sur GitHub, dans la fenêtre Code du projet, copier le lien du projet en cliquant sur « clone or download »



- Pour transférer le projet sur son ordinateur dans le dossier cp_pdr, dans un terminal, lancer les commandes :

```
$ cd
$ git clone https://github.com/cecileperrette/pdr.git cp_pdr
```

- Faire les modifications de son projet
- Pour savoir ce qui a été modifié, taper les commandes :

```
$ cd ~/cp_pdr/catkin_ws
$ git status
```

- Pour faire un commit, taper les commandes :

```
$ cd ~/cp_pdr/catkin_ws
$ git commit -am "correction sur simplemove (launchfile) - netoyage"
```

Les guillemets permettent de faire des commentaires sur les modifications que vous avez effectuées.

- Remettre à jour son projet avec la commande :

```
$ cd ~/cp_pdr/catkin_ws  
$ catkin_make
```

- Pour faire un push, lancer les commandes :

```
$ cd ~/cp_pdr/catkin_ws  
$ gitpush
```

Petite documentation sur ROS

Robot Operating System (ROS), est un ensemble d'outils informatiques open source permettant de développer des logiciels pour la robotique.

ROS permet de nombreuses fonctionnalités : abstraction du matériel, contrôle des périphériques de bas niveau, mise en œuvre de fonctionnalités couramment utilisées, transmission de messages entre les processus et gestions des packages installés.

ROS n'est pas dépendant d'un langage. Il est possible de programmer ROS en Python, en Lisp ou en C++.

ROS offre une architecture souple de communication inter-processus et inter-machine. Les processus ROS sont appelés des nœuds et chaque nœud peut communiquer avec d'autres via des topics. La connexion entre les nœuds est gérée par un master et suit le processus suivant :

1. Un premier nœud avertit le master qu'il a une donnée à partager
2. Un deuxième nœud avertit le master qu'il souhaite avoir accès à une donnée
3. Une connexion entre les deux nœuds est créée
4. Le premier nœud peut envoyer des données au second

Un nœud qui publie des données est appelé un publisher et un nœud qui souscrit à des données est appelé un subscriber. Un nœud peut être à la fois publisher et subscriber.

Un nœud peut correspondre à un capteur, un moteur, un algorithme de traitement, de surveillance...

ROS permet une communication inter-machine, des nœuds s'exécutant sur des machines distinctes, mais ayant connaissance du même master peuvent communiquer de manière transparente pour l'utilisateur.

Le package est l'unité principale d'organisation logicielle de ROS. Un package est un répertoire qui contient les nœuds, les bibliothèques externes, des données, des fichiers de configuration et un fichier de configuration xml.

Plus de documentation sur <http://wiki.ros.org/>

Petite documentation sur OpenCV

OpenCV (Open Computer Vision) est une bibliothèque graphique libre, spécialisée dans le traitement d'images en temps réel. La bibliothèque OpenCV met à disposition de nombreuses fonctionnalités très diversifiées permettant de créer des programmes partant des données brutes pour aller jusqu'à la création d'interfaces graphiques basiques.

Elle propose la plupart des opérations classiques en traitement bas niveau des images :

- Lecture, écriture et affichage d'une image ou d'une vidéo (depuis un fichier ou une caméra)
- Calcul de l'histogramme des niveaux de gris ou d'histogrammes couleurs ;
- Lissage, filtrage ;
- Seuillage d'image
- Segmentation
- Morphologie mathématique
- Détection de droites, de segments et de cercles
- Détection de visages par la méthode de Viola et Jones
- Détection de mouvement, historique du mouvement

Plus de documentation sur <https://opencv.org/>