Comment installer Turtlebot3 et Gazebo sur une VM Ubuntu 22.04 ARM64 avec ROS2 Humble.

Contexte

Le package utilisé pour faire fonctionner Turtlebot3 avec Gazebo fait partie de la distribution officielle et s'installe avec cette simple commande

sudo apt install ros-humble-turtlebot3-gazebo

Un problème survient car il n'existe aucun binaire compatible avec l'architecture ARM64, mais seulement AMD64. Ces outils n'étant plus maintenus, il n'est pas possible de récupérer un binaire.

Il faut donc récupérer tous les éléments nécessaires à la source et refaire le build à la place de l'installer.

Concernant Turtlebot3, la parade consiste à cloner les packages accessibles sur un Github distant dans le workspace local ros2_ws que l'on a créé lors de la séance précédente.

Turtlebot3 fonctionne avec Gazebo 11 classic, qui n'est lui aussi pas disponible en binaire pour ARM64 et qu'il faudra installer manuellement au préalable. De plus, ROS2 réutilise des packages ROS1 pour faire fonctionner Gazebo Classic, qui sont eux aussi indisponibles pour ARM64 et qu'il faudra télécharger depuis la source.

1 - Importer les packages Turltebot3

Executer les commandes suivantes dans un terminal pour cloner les packages depuis Git.

cd ~/ros2_ws/src
git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3.git
git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git

Si d'autres packages venaient à être manquants, il est possible de récupérer tout ce qui est disponible pour ARM64 ici (évidemment, turtlebot3-gazebo n'y figure pas)

sudo apt install ros-humble-turtlebot3*

Il est nécessaire de faire un build des packages, mais cette étape intervient à la fin, lorsque tout est installé.

2 - Installer Gazebo 11 Classic

Etape 1 : Récupérer toutes les dépendances

On commence par installer toutes les dépendances nécessaires pour compiler les sources de gazebo.

Auteur : Julien Niol

```
sudo apt update
sudo apt install -y \
  g++ cmake make python3-dev libboost-all-dev \
  libtinyxml2-dev libprotobuf-dev protobuf-compiler \
  libltdl-dev libfreeimage-dev libfreeimageplus-dev \
  libgts-dev libgdal-dev libtbb-dev \
  libgles2-mesa-dev libgl1-mesa-dev libglu1-mesa-dev libxmu-dev libxi-dev \
  libqt5core5a libqt5gui5 libqt5widgets5 qtbase5-dev \
  libsdformat9-dev libignition-math6-dev
```

Etape 2 : Récupérer le code source de Gazebo

```
cd ~
git clone -b gazebo11 https://github.com/gazebosim/gazebo.git gazebo11_src
cd gazebo11_src
```

Ici, le code source se copie directement dans un dossier à la racine du home. C'est dans celui-ci qu'il faudra faire le build.

Etape 3 : Compiler Gazebo

```
mkdir build
cd build

cmake .. \
  -DCMAKE_INSTALL_PREFIX=/opt/gazebo11 \
  -DCMAKE_BUILD_TYPE=Release

make -j$(nproc)
sudo make install
```

Il faut bien faire attention à réaliser l'installation dans un dossier /opt pour que ros2 puisse y avoir accès ensuite.

Le build peut échouer une première fois à cause de sdformat9. Le fichier came attend une version 9.8 alors que l'on a installé une version 9.7 avec les dépendances.
La version 9.7 peut fonctionner, il suffit de modifier simplement le fichier « ***SearchForStuff.cmake*** » que l'on trouve dans **~/gazebo11_src/cmake**. Il faut modifier la version à la ligne 647 et remplacer par 9.7

```
set(SDF_MIN_REQUIRED_VERSION 9.8)
```

S'il est nécessaire de relancer le build, penser à ne pas créer un nouveau dossier build dans le dossier existant.

Etape 4 : Déclarer les variables d'environnement

Pour que Gazebo soit vu correctement par ROS2, il est nécessaire de lui indiquer où chercher.

```
export GAZEBO_HOME=/opt/gazebo11
```

```
export PATH=/opt/gazebo11/bin:$PATH
export LD_LIBRARY_PATH=/opt/gazebo11/lib:$LD_LIBRARY_PATH
export GAZEBO_MODEL_PATH=/opt/gazebo11/share/gazebo-11/models:$GAZEBO_MODEL_PATH
export GAZEBO_PLUGIN_PATH=/opt/gazebo11/lib/gazebo-11/plugins:$GAZEBO_PLUGIN_PATH
```

Il est ensuite possible de tester que Gazebo est bien détecté.

```
source ~/.bashrc
which gazebo
gazebo --version
```

3 - Installer les dépendances manquantes pour ROS

Turtlebot3 s'appuie sur un package gazebo_ros pour faire fonctionner Gazebo, package qui est lui aussi absent de la distribution officielle disponible pour ARM64. Le package peut également se télécharger dans le workspace local. Il est nécessaire d'installer un plugin complémentaire (camera-info-manager) pour éviter une erreur lors du build.

```
cd ~/ros2_ws/src
git clone -b ros2 https://github.com/ros-simulation/gazebo_ros_pkgs.git
sudo apt update
sudo apt install ros-humble-camera-info-manager
```

4 - Builder Turtlebot3

Maintenant que tous les packages sont bien présents, il est nécessaire de builder Turtlebot3 afin de créer un exécutable que la commande **ros2 launch** pourra faire démarrer.

```
cd ~/ros2_ws
rosdep install --from-paths src --ignore-src -y
colcon build --symlink-install
```

La commande rosdep peut renvoyer une erreur si celui-ci n'est pas installé, mais ça ne semble pas être bloquant (rosdep est un package python qui peut s'installer par pip).

5 - Lancer Turtlebot3

On n'oublie pas de bien sourcer à la fois la distribution native ros et le workspace local, puis on peut ensuite reprendre le TP.

```
source /opt/ros/humble/setup.bash
source ~/ros2_ws/install/setup.bash
export TURTLEBOT3_MODEL=burger
ros2 launch turtlebot3_gazebo empty_world.launch.py
```

Conclusion

Ceci devrait permettre de faire fonctionner Gazebo et Turtlebot3 sur une VM Ubuntu 22.04 ARM64 et une distribution ROS2 Humble, que ce soit avec UTM ou Parallels. J'ai essayé beaucoup de choses pour arriver à cette solution, il n'est pas impossible que quelque chose ait été oublié ou que certaines configurations produisent des erreurs que je n'ai pas rencontré. Votre LLM favori devrait être d'une grande aide dans ce cas de figure.

Pour les autres distributions, Ubuntu 24 ou ROS2 Jazzy, ce support n'est d'aucune utilité.