2. Hogyan kezdjünk el a ROS2-vel dolgozni?

Kiegészítő anyag

Robot operációs rendszerek és fejlesztői ökoszisztémák

BMEVIIIAV55

Összeállította: Gincsainé Szádeczky-Kardoss Emese

szadeczky.emese@vik.bme.hu

Budapesti Műszaki és Gazdaságtudományi Egyetem

Irányítástechnika és Informatika Tanszék

2025

Telepítés

A ROS (Robot Operating System) a neve ellenére nem egy operációs rendszer, ezért először a megfelelő OS-ről kell gondoskodnunk. A különböző ROS disztribúciók különböző operációs rendszerekkel használhatók, ezekről a honlapon lehet tájékozódni (https://www.ros.org/).

A ROS1-ből már nincs támogatott verzió. Az utolsó ROS1 disztribúció, a Noetic Ninjemys, csak 2025 májusáig kapott támogatást (https://wiki.ros.org/Distributions), de még sok helyen lehet vele találkozni. A ROS2 disztribúciók itt tekinthetők meg: https://docs.ros.org/en/rolling/Releases.html. Választhatjuk például a Humble Hawksbill-t, amihez Ubuntuból a 22.04 (Jammy Jellyfish) verziót kell telepíteni. A további támogatott operációs rendszerek és a telepítési útmutatók szintén elérhetők a hivatalos honlapon: https://docs.ros.org/en/humble/Installation.html. Ha számunkra az a kényelmesebb, akkor virtuális gépre is telepíthető a megfelelő operációs rendszer és a kiválasztott ROS disztribúciónk (pl. VMware Workstation).

Érdemes még egy fejlesztői környezetet is telepíteni (pl. Visual Studio Code), és megfelelően konfigurálni, hozzáadni a fejlesztést segítő kiegészítőket. Továbbá szükség van még egy compiler-re is (pl. build-essential package).

Ahhoz, hogy parancssorból elérhessük a ROS szolgáltatásait, a következő utasításra van szükségünk minden egyes új terminál ablak megnyitásakor:

```
ros_user@ubuntu:~$ source /opt/ros/humble/setup.bash
```

Természetesen, ha más ROS2 verziót használunk, a „humble” helyett annak nevét kell írnunk.

Ha nem akarjuk ezt minden újabb terminál megnyitásakor begépelni, a `.bashrc` állomány végére is beszúrhatjuk:

```
ros_user@ubuntu:~$ echo "source /opt/ros/humble/setup.bash" >> .bashrc
```

Ha majd saját package-eket akarunk futtatni, akkor a fentiekhez hasonlóan be kell majd állítani a munkakörnyezetet.

Első próbálkozások

A telepítés ellenőrzése és ismerkedés gyanánt érdemes pár node-ot futtatni. (A ROS1-gyel ellentétben most nem kell a `roscore` utasítással ROS Mastert indítani.)

Például futtassuk egy terminálban a `turtlesim` package-ből a `turtle_sim` node-ot:

```
ros_user@ubuntu:~$ ros2 run turtlesim turtlesim_node
```

Egy másik terminálban ugyanebből a package-ből a `turtle_teleop_key`-t is indítsuk el:

```
ros_user@ubuntu:~$ ros2 run turtlesim turtle_teleop_key
```

Ha utóbbi terminálban a billentyűzet nyilait nyomkodjuk, a kis teknős mozog a `turtle_sim` node grafikus felületén. (Leállítani a node-okat például a `Ctrl+C` billentyűkombinációval lehet.)

Colcon workspace és saját package létrehozása

Ha saját package-eket akarunk létrehozni, ahhoz szükség van egy úgynevezett Colcon munkatérre (pl. `colcon_ws`). A package-ek a munkatéren belül egy `src` nevű könyvtárba kerülnek majd. Ezért hozzuk létre ezeket a mappákat:

```
ros_user@ubuntu:~$ mkdir -p ~/colcon_ws/src
```

A munkatér könyvtárába belépve érdemes a – még üres – munkateret buildelni a `colcon build` utasítással. Figyeljünk arra, hogy ezt az utasítást mindig a colcon workspace mappából (`colcon_ws`) adjuk ki!

```
ros_user@ubuntu:~$ cd ~/colcon_ws
ros_user@ubuntu:~/colcon_ws$ colcon build
```

A build során az `src` mellé új könyvtárak jöttek létre. Ezeket kilistázhatjuk:

```
ros_user@ubuntu:~/colcon_ws$ ls
```

Az `install` mappába is érdemes belenézni. Itt is vannak `setup` állományok. A `setup.bash`-t vagy a `local_setup.bash`-t kell source-olni minden megnyitott terminálban, ha ott a saját package-ből szeretnénk állományokat futtatni:

```
ros_user@ubuntu:~/colcon_ws$ ls install
ros_user@ubuntu:~/colcon_ws$ source install/local_setup.bash
```

Ha nem szeretnénk ezt a sort minden megnyitott terminálba begépelni, akkor itt is működik, hogy kiegészítjük a `.bashrc` állományt:

```
ros_user@ubuntu:~/colcon_ws$ cd ..
ros_user@ubuntu:~$ echo "source ~/colcon_ws/install/local_setup.bash" >>
.bashrc
```

A colcon munkatér `src` mappájából létre tudunk hozni egy új package-et. Ennek meg kell adni a nevét, és hogy milyen további package-eket, library-ket használ (dependencies). Hozzunk most létre egy `my_pkg` nevű package-et, amiben használni fogjuk a `rclcpp` szolgáltatásait:

```
ros_user@ubuntu:~ $ cd colcon_ws/src
ros_user@ubuntu:~/colcon_ws/src$ ros2 pkg create my_pkg --dependencies
rclcpp
```

Az utasítás hatására létrejöttek új fájlok (`package.xml`, `CMakeLists.txt`) és új mappák (`my_pkg` és pár alkönyvtár). Tegyünk eleget a megjelenő `TODO` kéréseknek, és nézzük meg, egészítsük ki a manifest fájlt:

```
ros_user@ubuntu:~/colcon_ws/src$ gedit my_pkg/package.xml
```

A fájl elején meg lehet adni számos adatot, utána pedig a függőségek szerepelnek. Ha a package létrehozásakor (`ros2 pkg create` utasítás) esetleg nem adtunk meg minden szükséges függőséget, akkor azt később itt lehet beállítani.

Ha C++ nyelven szeretnénk node-okat implementálni, akkor a forráskódot a `my_pkg/src` könyvtárába mentsük. A `CMakeLists.txt` állományt is módosítani kell ilyenkor. Ha kész vagyunk a forráskóddal, build-elni a már korábban látott `colcon build` utasítással kell a `colcon_ws` könyvtárból:

```
ros_user@ubuntu:~/colcon_ws/src$ cd ..
ros_user@ubuntu:~/colcon_ws$ colcon build
```

A package létrehozása után a megnyitott terminálokban source-olni kell a local_setup.bash-t:

```
ros_user@ubuntu:~/colcon_ws$ source install/local_setup.bash
```

Az implementáció, fordítás, futtatás részleteit a következő alkalomhoz készített kiegészítő anyag tartalmazza.