Turtlebot3 docs

1 Hasznos információk

ROS bevezető:

1. ROS

TurtleBot3 - a robot összerakásához az útbaigazítás itt található meg, valamint a kezelése a ROS környezetben billentyűzettel, példaprogramokkal, joystick-el stb.:

1. TurtleBot3 Overview

Rospy - Python kliens a ROS-hoz. Az első linken példaprogramok találhatóak, a második linken a dokumentáció, hogyan kell feliratkozni egy ROS topic-ra, hogyan kell egy topic-ot létrehozni, stb. Itt példázva látható hogyan lehet egy egyszerű script-tel irányítani a robotot:

1. A "getting started" guide for developers interested in robotics
2. Rospy doksi

OpenCR: - A motrokat irányító modul, a motorkontroller. Itt fontos megemlíteni, hogy több könyvtár is van mellyel a motrokat irányíthatjuk. A motrokat tudjuk konfigurálni különböző vezérlési módra, állíthatunk baud rate-t, stb.:

1. Dynamixel2Arduino - gyors és egyszerű vezérlése a motroknak Arduino IDE-ben
2. Dynamixel sdk - ezt a könyvtárat használja a ROS

Dynamixel motrok:

1. Motorleírás

2 ROS

2.1 Ismertető

A ROS-ban a számítások elvégzéséhez ROS-csomópontok néven ismert szoftverkomponenseket használják. A ROS-csomópontok, a mester, a paraméterkiszolgáló, az üzenetek, a témák, a szolgáltatások és a táskák a számítási gráf központi elemei. A gráf minden egyes fogalma más-más módon járul hozzá a gráfhoz.

A ros_comm nevű verem tartalmazza a ROS kommunikációval kapcsolatos csomagjait, beleértve az olyan alapvető klienskönyvtárakat, mint a roscpp és a rospython, valamint az olyan fogalmak megvalósítását, mint a témák, csomópontok, paraméterek és szolgáltatások. Ezek a fogalmak további vizsgálatához ez a verem olyan eszközöket is tartalmaz, mint a rostopic, rosparam, rosservice és rosnode.

A ROS kommunikációs middleware csomagok együttesen alkotják a ROS Graph réteget, a ros_comm stack tartalmazza ezeket a csomagokat.

Az ROS gráfok fogalmai:

- Csomópontok: A számításokat végző folyamatokat csomópontoknak nevezzük, melyek létrehozásához a roscpp és rospy ROS klienskönyvtárakat használjuk. Ezekben a kommunikáció több formáját is megvalósíthatjuk a klienskönyvtári API-k segítségével. Egy robotban számos csomópont lesz a különböző feladatok elvégzésére, valamint ROS kommunikációs protokollok segítségével kommunikálhatnak egymással és oszthatnak meg adatokat. A ROS-csomópontok egyik célja, hogy inkább kis folyamatokat építsünk ki minimális funkcionalitással, mint komplexet, emiatt egyszerűen hibakereshetők.

- Mester: A ROS Master(mester) névkeresést és regisztrációt kínál a többi csomópont számára, enélkül a csomópontok nem tudják megtalálni egymást, nem tudnak kommunikálni egymással, és nem tudnak szolgáltatásokat használni. Egy elosztott rendszerben a mestert egyetlen számítógépen kell futtatni, majd a távoli csomópontok ehhez a mesterhez tudnak csatlakozni, hogy megtalálják egymást.

- Paraméter-kiszolgáló: A paraméterkiszolgáló használatával egy helyen tarthatja az adatokat, ennek segítségével minden csomópont hozzáférhet ezekhez az értékekhez és ellenőrizheti azokat. A ROS Master tartalmaz egy paraméterkiszolgálót.

- Üzenetek: Az üzenetek a csomópontok közötti kommunikáció elsődleges eszközei. Egyszerűen fogalmazva, az üzenet egy olyan adatszerkezet, amely egy tipizált mezővel rendelkezik, amely egy adatgyűjteményt tartalmazhat, és továbbítható egy másik csomópontnak. A ROS-üzenetek egész, lebegőpontos, Boolean és más általános primitív típusokat támogatnak, de szerkeszthetünk saját üzenettípust is akár.

- Témák: A ROS-ban minden kommunikáció egyedi névvel rendelkező topicokon (témakon), megnevezett buszokon keresztül történik. Azt mondhatjuk, hogy egy csomópont egy topicot publikál, amikor egy topicon

keresztül üzenetet küld, valamint feliratkozik egy topicra, amelyen keresztül üzeneteket kap. A feliratkozó csomópont és a publikáló csomópont nem tudnak egymásról, így olyan témákra is lehet feliratkozni, amelyeknek nincs kiadója. Más szóval az információ létrehozása és fogyasztása különálló folyamat. Amíg egy csomópont rendelkezik a megfelelő üzenettípussal, addig hozzáférhet a témához és küldhet rajta keresztül adatokat.

- Szolgáltatások: Ha egy robotalkalmazás kérés-válasz interakciót igényel, a publikálás/feliratkozás megközelítés önmagában nem biztos, hogy elegendő. A kérés/válasz típusú interakcióra akkor lehet szükség, ha elosztott rendszerrel dolgozunk, mivel a publikálás/feliratkozás architektúra lényegében egyirányú szállítási rendszer. Ezekben a helyzetekben ROS-szolgáltatásokat használnak. Lehetséges egy olyan szolgáltatásdefiníció, amely két részből áll, egy a kérésekhez és egy a válaszokhoz. A ROS szolgáltatások segítségével létrehozhatunk egy kiszolgáló csomópontot és egy kliens csomópontot. Amikor az ügyfélcsomópont kérő üzenetet küld a kiszolgálónak, az válaszol, és az eredményt átadja az ügyfélnek. A kiszolgáló csomópont egy név alatt nyújtja a szolgáltatást és lehetséges, hogy az ügyfélnek várnia kell, amíg a kiszolgáló válaszol.

- Táskák: A ROS kommunikációs adatok tárolása és lejátszása táskákban történik. A táskák kulcsfontosságú adattárolási mechanizmust jelentenek az érzékelőadatok számára, amelyek megszerzése kihívást jelenthet, de a robotalgoritmusok létrehozásához és teszteléséhez szükségesek. A bonyolult robotmechanizmusokkal való munka során a táskák rendkívül praktikus funkciót jelentenek.

Az ilyen típusú gráfok az rqt_graph nevű eszközzel generálhatók.

2.2 A ROS csomópontok

A ROS kliens könyvtárak, például a roscpp és a rospy használatával hozhatunk létre ROS-csomópontokat, amelyek számításokat végeznek. A ROS-témákat, szolgáltatásokat és paramétereket egy csomópont használhatja más csomópontokhoz való kapcsolódáshoz. Egy robotban számos csomópont lehet jelen, például olyanok, amelyek feldolgozzák a kameraképeket, kezelik a robot soros kommunikációját, kiszámítják az odometriát stb.

A rendszer hibatűrővé tehető a csomópontok használatával. Egy teljes robotrendszer akkor is tovább működhet, ha egy csomópont összeomlik. Mivel minden egyes csomópont csak egy funkciót kezel, a csomópontok megkönnyítik a hibakeresést.

Minden működő csomópontnak nevet kell adni, hogy a rendszer többi része felismerhesse. Például a /camera_node lehet egy olyan csomópont neve, amely kameraképeket sugároz.

A rosbash eszközzel a ROS-csomópontokat vizsgálhatjuk. A rosnode parancs segítségével információkat kaphatunk egy ROS-csomópontról. Itt vannak a rosnode használatai

- `$ rosnode info [node_name]`

  ez kiírja a csomópontra vonatkozó információkat

- `$ rosnode kill [node_name]`

  Megől egy futó csomópontot

- `$ rosnode list`

  Ez listázza a futó csomópontokat

- `$ rosnode machine [machine_name]`

  Ez felsorolja az adott gépen futó csomópontokat vagy a gépek listáját.

- `$ rosnode ping`

  Ez ellenőrzi a csomópont csatlakoztathatóságát.

- `$ rosnode cleanup`

  Ez eltörli az elérhetetlen csomópontok regisztrációját.

2.3 ROS üzenetek

A ROS-csomópontok üzeneteket cserélnek egymással egy témába való üzenetküldéssel. Az üzenetek egy egyszerű adatstruktúra, amely mezőtípusokkal rendelkezik, ahogyan azt korábban már tárgyaltuk. A ROS-üzenet a szabványos primitív adattípusokat és a primitív típusokból álló tömböket támogatja.

A szolgáltatáshívások egy másik módszer, amellyel a csomópontok kommunikálhatnak. A szolgáltatások üzenetek, és az srv fájl tartalmazza az egyes szolgáltatási üzenettípusok definícióit.

A következő technikával hozzáférhetünk az üzenetspecifikációhoz. Például az std_msgs/String használatával megkaphatjuk az std_msgs/msg/String.msg fájlt. A string üzenet definíciójához a roscpp kliens használata esetén be kell vennünk az std_msgs/String.h állományt.

Annak megállapításakor, hogy a kiadó és az előfizető ugyanazokat az üzenetadattípusokat cseréli-e ki, a ROS az MD5 ellenőrző összegeket is összehasonlítja.

A ROS-üzenetekkel kapcsolatos információkhoz a ROS beépített eszközöket tartalmaz, a rosmsg-et. A következő parancsokat használhatjuk:

- `$ rosmsg show [message]`

  Ez mutatja az üzenet leírását

- `$ rosmsg list`

  Ez felsorolja az összes üzenetet

- `$ rosmsg md5 [message]`

  Ez megjeleníti az üzenet md5összegét

- `$ rosmsg package [package_name]`

  Ez felsorolja a csomagban lévő üzeneteket

- `$ rosmsg packages [package_1] [package_2]`

  Ez felsorolja az üzeneteket tartalmazó csomagokat

2.4 ROS témák

A buszok olyan ROS-témák, amelyeken keresztül a ROS-csomópontok kommunikálnak egymással. A topikok anonim publikálási és feliratkozási képessége azt jelenti, hogy az üzenetek létrehozása és fogyasztása különálló folyamatok. A ROS-csomópontok csak a téma nevét ellenőrzik, valamint azt, hogy a kiadó és a feliratkozó üzenettípusa megegyezik-e; nem érdekli őket, hogy melyik csomópont publikálja vagy iratkozik fel a témákra.

A topikok csak egyirányú kommunikációt tesznek lehetővé; ha kérés-válasz kommunikációt akarunk kiépíteni, akkor ROS-szolgáltatásokat kell használnunk.

A ROS-csomópontok a TCP/IP-alapú TCPROS transzportot használják a topikokhoz való csatlakozáshoz. A ROS-ban ezt a technikát használják alapértelmezett szállítási mechanizmusként. A kommunikáció másik formája az UDPROS, amely kizárólag a távműködtetésre alkalmas, és alacsony késleltetésű, laza szállítást biztosít.

A ROS téma eszközzel a ROS témákról kaphatunk információkat. Példa:

- `$ rostopic bw /topic`

  Ez a parancs megjeleníti az adott téma által használt sávszélességet.

- `$ rostopic echo /topic`

  Ez a parancs kiírja a megadott téma tartalmát.

- `$ rostopic find /message_type`

  Ez a parancs megkeresi a megadott üzenettípust használó témákat.

- `$ rostopic hz /topic`

  Ez a parancs megjeleníti az adott téma publikálási arányát.

- `$ rostopic info /topic`

  Ez a parancs információkat nyomtat ki egy aktív témáról.

- `$ rostopic list`

  Ez a parancs felsorolja az összes aktív témát a ROS rendszerben.

- `$ rostopic pub /topic message_type args`

  Ezzel a paranccsal közzétehet egy értéket egy üzenettípusú témában.

- `$ rostopic type /topic`

  Ez megjeleníti az adott téma üzenettípusát.

2.5 ROS szolgáltatások

A ROS-szolgáltatásokat minden olyan esetben használni kell, amikor kérés/válasz típusú kommunikációra van szükség. A ROS-témák egyirányú jellege miatt ez a fajta kommunikáció nem lehetséges. Az elosztott rendszerek az esetek többségében ROS-szolgáltatásokat alkalmaznak.

A ROS-szolgáltatásokat egy üzenetpár határozza meg. Egy srv fájlban meg kell adnunk a kérés adattípusát és a visszatérés adattípusát. Egy csomag belső srv alkönyvtárában találhatók az srv fájlok.

Egy csomópont egyszerre szolgál kliensként és szerverként a ROS-szolgáltatások számára, lehetővé téve, hogy a kliensek szolgáltatásokat kérjenek a szerverektől. Az eredmények akkor kerülnek elküldésre a szolgáltatás kliensének, ha a szerver sikeresen végrehajtja a szolgáltatási eljárást.

A következő megközelítéssel például elérhetjük a ROS szolgáltatásdefiníciót, például ha a my_package/Image elérheti a my_package/srv/Image.srv állományt.

Van egy MD5 ellenőrző összeg, amely a ROS szolgáltatások csomópontjait is ellenőrzi. Ha az összeg megegyezik, csak a szerver válaszolhat az ügyfélnek.

Két ROS-eszköz áll rendelkezésre a ROS-szolgáltatás megismeréséhez. A különböző szolgáltatástípusok megismeréséhez használja az első eszközt, a rossrv-t, amely megegyezik a rosmsg-gel. A következő parancs, a rosservice, az aktuálisan aktív ROS-szolgáltatások listázására és lekérdezésére szolgál.

Az alábbiakban néhány utasítást olvashat, hogyan használhatja a rosservice eszközt, hogy többet tudjon meg a jelenleg aktív szolgáltatásokról:

- `$ rosservice call /service args`

  Ez az eszköz meghívja a szolgáltatást a megadott argumentumokkal.

- `$ rosservice find service_type`

  Ez a parancs a megadott szolgáltatástípusba tartozó szolgáltatásokat keresi.

- `$ rosservice info /services`

  Ez információt nyomtat ki az adott szolgáltatásról.

- `$ rosservice list`

  Ez a parancs felsorolja a rendszerben futó aktív szolgáltatásokat.

- `$ rosservice type /service`

  Ez a parancs kiírja az adott szolgáltatás típusát.

- `$ rosservice uri /service`

  Ez az eszköz kiírja a szolgáltatás ROSRPC URI-ját.

2.6 ROS táskák

A ROS-ban a témák és szolgáltatások üzenetadatait bag-fájlokban tárolják. A bag fájlokat a.bag kiterjesztés jelöli.

A rosbag parancs egy vagy több témára való feliratkozással és az üzenet adatainak a fogadáskor történő elmentésével hoz létre bag-fájlokat. Ez a fájl újra lejátszhatja ugyanazokat a témákat, amelyekből rögzítették őket, vagy újra leképezheti az aktuális témákat.

Az adatnaplózás a rosbag elsődleges felhasználási területe. A robotadatok offline is megjeleníthetők és feldolgozhatók, valamint naplózhatók.

A rosbag fájlokkal való munkához a rosbag parancsra van szükség. A bag fájl rögzítésére és lejátszására szolgáló parancsok a következők:

- `$ rosbag record [topic_1] [topic_2] -o [bag_name]`

  A megadott témákat ez a parancs rögzíti a parancs által megadott zsákfájlba. A -a argumentummal bármilyen témát rögzíthetünk.

- `$ rosbag play [bag_name]`

  Lejátssza a meglévő táska tartalmát.

A GUI eszköz amivel táskákkal dolgozhatunk az rqt_bag.

2.7 ROS mester

A DNS-kiszolgáló hasonló a ROS Masterhez. Minden induló ROS-csomópont elkezdi megkeresni a ROS Master-t és regisztrálni a nevét. Ennek eredményeképpen a ROS Master minden olyan csomópontról tud, amely jelenleg aktív a ROS-rendszerben. Visszahívást generál, és frissíti a legfrissebb információkkal, amikor bármelyik csomópont adatai megváltoznak. Az egyes csomópontokhoz való csatlakozáshoz ezek a csomópontadatok hasznosak.

Amikor egy csomópont elkezd közzétenni egy témát, a csomópont értesíti a ROS Master-t a téma nevéről és adattípusáról. Ha vannak további csomópontok, amelyek feliratkoztak ugyanarra a témára, a ROS Master ellenőrzi őket. A kiadó csomópont információit a ROS Master megosztja az előfizető csomóponttal, ha ugyanarra a témára több csomópont is előfizetett. A két csomópont a csomópontadatok átvétele után a TCP/IP-alapú TCPROS protokoll segítségével csatlakozik. A ROS Master nem játszik szerepet a két csomópont kezelésében, miután azok összekapcsolódtak. A preferenciáinktól függően megállíthatjuk akár a kiadó csomópontot, akár az előfizető csomópontot. Újra ellenőrzi a ROS Masterrel, ha bármelyik csomópont leállt. A ROS-szolgáltatások ugyanezt az eljárást alkalmazzák.

A ROS kliens könyvtárak, köztük a roscpp és a rospy a csomópontok létrehozására szolgálnak. Ezek a kliensek XMLRPC-alapú API-kon keresztül kommunikálnak a ROS Masterrel, amelyek a ROS rendszer API-k rendszerhátttereként szolgálnak.

A ROS Master IP-címe és portja a ROS_MASTER_URI környezeti változóban található. Ez a változó lehetővé teszi, hogy a ROS-csomópontok megtalálják a ROS Master-t. A csomópontok közötti kommunikáció nem fog megtörténni, ha ez a változó ki van kapcsolva. A localhost IP-cím vagy név használható, ha a ROS-t egyetlen rendszerben használjuk. Azonban egy elosztott hálózatban, ahol a feldolgozás sok fizikai gépen zajlik, helyesen kell definiálnunk a ROS_MASTER_URI-t. Csak így lesznek képesek a távoli csomópontok megtalálni és kommunikálni egymással. Egy elosztott rendszerben csak egy Masterre van szükségünk, és ahhoz, hogy a távoli ROS-csomópontok elérjék a Master-t, annak olyan számítógépen kell futnia, amelyet az összes többi számítógép sikeresen tud pingelni. [**?**]