Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Tesztfuttató keretrendszer készítése ROS-hez

Hajdu Csaba (H7J6MU), (BSc) mérnök informatikus szakos hallgató
Konzulens: dr. Micskei Zoltán, MIT
Informatikai technológiák szakirány, Rendszertervezés ágazat
Önálló laboratórium összefoglaló
2013/14. II. félév

A robotok fejlesztése során több száz funkciópontot valósítunk meg. Amikor ezen funkciókat szeretnénk tesztelni be kell rendezni egy környezetet tele objektummal, a robotot működtetni kell és kell legalább egy ember, aki megfigyeli a robot helyes működését. Összeségében ez sok pénz és idő, ami a robot fejlesztése során nagymértékben növekszik, hiszen egy kis változtatás során a robot helyes működésének biztosításához az összes szükséges funckiót újra le kell tesztelni ugyanazzal az elrendezéssel.

A robotok tesztelését szeretnénk automatizálni, ezáltal felszabadítva a teszteléshez egyébként szükséges erőforrásokat. Ehhez szükséges a robotot és környezetét szimulálni, a megfigyelést és a konfigurálást pedig egy programmal felügyelni. Elvárjuk, hogy a megfigyelés során generált kimenet olvasható és egyértelmű legyen, amit más programok tudnak értelmezni. Elvárás még a program minél finomabb és újrahasználható konfigurációja, amit egy bemeneti fájllal tehetünk meg.

Az *ROS* egy keretrendszert biztosít a valós vagy szimulált robot komponenseinek összekötésére a számítógépekkel, azon keresztül a felhasználóval. Egy teljes kommunikációs modellt biztosít aminek megfigyelésével feljegyezhetjük a rendszer eseményeit és a robot működését, tualjdonképpen az egész *ROS* működése felfogható egy számítási gráfként, amelyben a csúcspontok a futó folyamatok, az élek a kommunikcáiós csatornák, a kapacítások az elküldött üzenetek. Az *ROS* integrált része emellett egy szimulációs program (*Gazebo*), amiben objektumokkal berendezhetünk és megfigyelhetünk egy tesztkörnyezetet. Az *ROS* programok fejlesztése *C++*-ban vagy *Python*-ban lehetséges.

A robotok teszteléséhez a robot funkcionalitását is konfigurálni kell és előkészíteni a működéshez. A robot egy fő funkcionalitása a navigáció, vagyis annak képessége, hogy egy robot aktuális állapotából eljusson egy kijelölt pontba a környezetében, miközben a felmerülő akadályokat megfelelően kezeli. A probléma egyik része a navigáció előkészítése és navigációs célok kezelése, a keretrendszerben ez egy *C++* programot jelent. Egy tipikus alkalmazás a robotoknál a *SLAM* (*Szimultán Lokalizáció és Térképkészítés, Simultaneous Localization and Mapping*), ahol a navigációhoz szükséges egy minimális térkép készítése a navigáció elkezdéséhez, ehhez pedig a környezet részleges bejárása. Mindezt vezérelni a program felelőssége.

A fő komponense a problémának a szimuláció és a robot egyes komponenseinek megfigyelése, ennek megvalósítása egy *Python* program. A külsőleg konfigurálható program feladata a folyamatok kommunikációjának és a szimuláció objektumainak megfigyelése, majd az eredményekből egy XML formázott kimeneti fájl generálása. A monitorozás folyamatában a legnagyobb prioritást a geometriai adatok megfigyelése kap. A geometriai adatokat a *Gazebo* szolgáltatja, amiknek struktúráját a program előre ismeri. A program megfigyelhet még bármilyen előre nem ismert struktúrájú kommunikációt is. Elkülönítünk kvázifolytonos folyamszerű és diszkrét eseményszerű adatokat, előbbit mintavételezi, utóbbit elfogja. A keretrendszer alprogramjai közötti kommunikációt és felügyeletet ez a program végzi, az eseményeket a kimenetbe menti. A program a konfigurációban beállított időmennyiségig fut, utána leállítja a monitorozást, opcionálisan a teljes szimulációt is.