



**İstanbul Teknik Üniversitesi Robotik Arama Kurtarma Ekibi**

**Görev Raporu**

**Görev Konusu:** Ros Elemanları Araştırması

**Hazırlayan:** Kıvanç Karaüzüm

**Teslim Tarihi:** 14.11.2023

**Ros Workspace:**

Ros projelerini oluşturmak, geliştirmek ve düzenlemek için kullanılan bir kavramdır. ”catkin\_make” komutu ile oluşturulabilir kodda kullanılacak paketler buraya eklenir.

**Nodes:**

Genellikle tek görevi yapmak için hazırlanan bağımsız bir kod bloğudur. Node birbirlerinden bağımsız çalışırlar birinin çökmesi diğerlerini etkilemez. Ancak birbirleriyle pub-sub bağlantılarıyla iletişim kurarak senkron bir şekilde çalışabilirler.

**Topics:**

Topicler pub-sub iletişiminde merkez konumundadır. Eğer pub-sub iletişimini hocanın tahtaya yazdıklarını okuyan öğrenciler olarak alırsak. Publisher hoca, subscriber öğrenci, topic ise karatahtayı temsil eder. Nodelar arasındaki bağlantı topicler üzerinden sağlanır.

**Services:**

Services pub-sub iletişiminden farklı bir iletişim mekanizmasıdır. İstemci-sunucu modeliyle çalışır. Bir node’un başka bir node belirli bir görevi yapması için talep göndermesi ve buna bağlı bir yanıt almasına olanak sağlar.

**Master-Slave Structure:**

Master-slave structure Ros’un temel işleyiş şeklidir. İki parçadan oluşur: Ros Master, Nodes. Nodes yukarda anlatıldığı gibi özel görevlerle ilgilenen bağımsız kod bloklarıdır. Ros Master bu nodelar arasındaki iletişimi sağlar ve nodelerin koordineli bir şekilde çalışmalarını sağlayarak onları bir bütün haline getirir. Master-Slave yapısı sayesined robotlarda kullanılan nodelardan birisi çökse dahi kalanları çalışmaya devam eder bu sayede robotların hata verme olasılığı minimalize edilir. Ayrıca birbirlerinden bağımsız çalışmaları birbirlerindenbağımsız da hazırlanabileceleri anlamına gelir bu da iş paylaşımını kolaylaştırır ve geliştirme sürecini daha hızlı ve verimli hale getirir.

**Parameter Server:**

Sistemin dinamikliğini sağlayan yapılardan biridir. Motorun hızı, aracın konumu vb. parametlerini saklanabildiği ve güncellenebildiği bir serverdır. Nodelar görevlerini yerine getirirken bu parametreleri okuyabilir yada yazabilir bu sayede sistem değişen şartlara karşı kendini güncel tutmayı başarır. Bu güncelleme yetisi sayesinde robotların hassasiyet ve başarı oranları yükselir. Verilerin saklanabilir oluşu hata ayıklama arıza giderme durumlarında sorunun nerede olduğunun görülmesini kolalaştırır.

**.launch:**

Bu komut genellikler birden fazla node çalıştırmak, parametreleri yüklemek için kullanılır. Özellikle birden fazla node’un belirli bir sırayla çalışması gereken karmaşık yazılımlarda kullanılır. Özellikle hareket eden robotlarda sistemlerin sırayla çalışması büyük önem arz eder. Nodeların yanlış sırayla çok kolay bir dönüş veya kol hareketinin başarısızlıkla sonuçlanmasına çok kolay yol açabilir.

**.yaml:**

Genellikle başlatma dosyalarında veya nodelarda kullanılırlar. Veri ve parametreleri düzgün bir yapı ve düzende saklamak ve dağıtmak için kullanılır.. Node’lara parametre yüklemede kullanılır.

**CMakelist.txt:**

İçerisinde bir dizi komut ve açıklama bulunan dosyalardır. Amacı programın kullandığı kütüphaneleri, kaynakları ve hedefini açıklamaktır denilebilir.

**MoveIt:**

Moveit kinetik özelliğe sahip robotların kullanımını kolaylaştıran bir arayüzdür. İçerisinde komut önizlemeleri ve hareket similasyonları yapılabilir. Robotların bileşenleri sisteme eklenerek geliştirme aşamasındaki bir robotun önizlemeleri yapılıp hataları önceden farkedilebilir. Görsel arayüzler 2D düzlemdeki kodlarıd 3D uzaya aktararak yazdığımız kodların çok daha net bir şekilde görülmesini sağlar ve kısa yolların hataların farkedilmesinde büyük öneme sahiptir.

**Gazebo:**

Fizik tabanlı bir similasyon programıdır. Fizik tabanlı olduğu için robotun gerçek dünya tepkileri konusunda daha net bilgiler verir. 3D similasyonlarıyla robotları üretmeden önce tasarımları mükemmelleştirme ve test etme olanağı sağlar. Ros ile entegre çalışabilme özelliği sayesinde robotik alanında sıkça tercih edilen bir similasyon uygulamasıdır. Yaptığımız robotların gerçek dünyadaki cisimlerle olan etkileşimleri özellikle arama kurtarma gibi alanlarda hayati öneme sahiptir. Yaptığımız aracın üretiminde sonraki aşamaya geçmeden önce cisimlerle ve kendisiyle olan etkileşiminin hesaplanması aracın sınırlarını daha belirgin ve kullanım alanlarının daha net belirlenmesini sağlar.

**rViz:**

Gazeboya kıyasla daha çok görselleştirme odaklı bir arayüzdür. Fizik motoru içermediği için gerçek hayat sümilasyonlarında zayıftır. Ancak hata ayıklama, similasyon sonuçlarını değerlendirmek için daha kullanışlıdır. Özellikle sensör verilerini değerlendirmek için kullanılır. Çoğu kod her ne kadar kendi hata ayıklama sistemlerini içerse ve özel renk metodlarıyla hataları bize göstermeye çalışsa da hatalar ne yazık ki kaçınılmazdır. RViz gibi arayüzler bu hataların daha net bir şekilde görülmesini hızlı tepki alınmasını kolaylaştırır. Ayrıca elimizdeki veri yığınını bir sayı kümesinden mantıklı bir sonuca bağlamada da büyük rol oynar.

**Kaynakça:**

[(166) MoveIt Capabilities Overview - YouTube](https://www.youtube.com/watch?v=7KvF7Dj7bz0&ab_channel=PickNikRobotics)

[(166) ROS1 Gazebo ve Rviz simülasyon ortamları tanıtımı - YouTube](https://www.youtube.com/watch?v=NFo9Buk8mKs&ab_channel=NevzatBol)

[What is the Difference Between rviz and Gazebo? – Automatic Addison](https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/#:~:text=The%20difference%20between%20the%20two,you%20what%20is%20really%20happening.%E2%80%9D)