İçindekiler

Bölüm 1

Giriş

1.1 İleri Seviye ROS2 ve Python Kullanımının Önemi

1.1.1 İleri Seviye ROS2 Kullanımının Faydaları

İleri seviye ROS2 kullanımı, robotik uygulamalar geliştirmede önemli faydalar sağlar. İleri düzey bilgi ve becerilere sahip olmak, daha etkin ve karmaşık robot sistemleri geliştirmenize olanak tanır. Ayrıca, performans ve optimizasyonu artırarak daha hızlı ve verimli sistemler oluşturabilirsiniz. İleri düzey ROS2 kullanımının faydalarından bazıları şunlardır:

Etkin Robot Sistemleri

İleri seviye ROS2 kullanımı sayesinde, daha etkin ve karmaşık robot sistemleri geliştirilebilir. Bu, daha gelişmiş robot kontrol ve planlama algoritmaları, insan-robot etkileşimi ve yapay zeka entegrasyonu gibi alanlarda daha başarılı projeler gerçekleştirmenize yardımcı olacaktır.

Yüksek Performans ve Optimizasyon

İleri seviye ROS2 teknikleriyle, performans ve optimizasyonu artırarak daha hızlı ve verimli sistemler oluşturabilirsiniz. Bu, daha düşük gecikme süreleri, daha iyi enerji verimliliği ve daha yüksek güvenilirlik gibi önemli avantajlar sağlar.

1.1.2 Python ile Etkin Geliştirme

Python, etkin ve hızlı bir şekilde robotik uygulamalar geliştirmenize olanak tanır. Python'un sunduğu yapılar sayesinde, hızlı ve esnek kod geliştirme süreçleri sağlanabilir. Ayrıca, Python'un geniş kütüphane desteği, ROS2 projelerinde de kullanılabilir ve bu sayede daha kapsamlı işlemler gerçekleştirilebilir. Python ile etkin geliştirme konusunda şu faydaları göz önünde bulundurabilirsiniz:

Hızlı Kod Geliştirme

Python dilinin sunduğu yapılar sayesinde, hızlı ve esnek kod geliştirme süreçleri sağlanabilir. Bu, prototip oluşturma aşamasından ürünleşmeye kadar her aşamada zaman

kazandırır ve projenizin daha hızlı ilerlemesine yardımcı olur.

Büyük Kütüphane Desteği

Python dilinin geniş kütüphane desteği, ROS2 projelerinde de kullanılabilir ve bu sayede daha kapsamlı işlemler gerçekleştirilebilir. Örneğin, Python'da mevcut olan veri analizi, görüntü işleme ve yapay zeka kütüphaneleri sayesinde, robotik uygulamalarınızın yeteneklerini önemli ölçüde artırabilirsiniz.

1.2 İçeriğin Amacı ve Kapsamı

1.2.1 İçeriğin Amacı

Bu içeriğin amacı, temel ROS2 bilgisine sahip kullanıcıların, ileri düzey ROS2 ve Python kullanımı konusunda bilgi ve beceri sahibi olmalarına yardımcı olmaktır. Bu sayede kullanıcılar, daha karmaşık ve etkin robot sistemleri geliştirebilir, performans ve güvenilirlik açısından daha iyi sonuçlar elde edebilirler.

1.2.2 İçeriğin Kapsamı

Bu içerik, aşağıdaki konuları kapsamaktadır:

1. ROS2 bileşenleri ve mimarisi
2. İleri düzey paket oluşturma ve yönetimi
3. ROS2 güvenlik (SROS2)
4. ROS2 ve Python ile test ve entegrasyon
5. ROS2 ve gerçek zamanlı sistemler (Real-Time)
6. İleri düzey TF2 kullanımı ve optimizasyon
7. ROS2 ile simülasyon ve donanım entegrasyonu
8. İleri düzey araçlar ve kullanımları
9. ROS2 ve ileri düzey uygulamalar ve örnek projeler

İçeriğin kapsamı boyunca, bu konularla ilgili temel kavramlar, yöntemler ve pratik uygulamalar ele alınacaktır.

Bölüm 2

ROS2 Bileşenleri ve Mimarisi

2.1 ROS2'nin Temel Bileşenleri ve İleri Düzey Kullanımı

2.1.1 Node'lar, Topic'ler, Service'ler ve Action'lar

Node'lar, ROS2 sisteminin temel yapı taşlarıdır ve her biri, robotik uygulamanızın belirli bir görevini yerine getirir. Bu görevler, veri işleme, sensör okumaları veya hareket planlaması gibi çeşitli işlemler olabilir. Topic'ler, node'ların birbirleriyle veri paylaşmalarına olanak tanırken, service'ler ve action'lar daha karmaşık istek/yanıt ve uzun süreli etkileşimler için kullanılır.

Şekil 2.1: Node'lar, topic'ler, service'ler ve action'lar arasındaki ilişki (Burada görselde node'lar, topic'ler, service'ler ve action'lar arasındaki ilişkiyi gösteren bir akış diyagramı kullanılmalıdır)

İleri Düzey Node Kullanımı

İleri düzey node kullanımı, performansı ve kaynak kullanımını optimize etmek için node içerisindeki işlemlerin düzenlenmesi ve node'lar arası iletişimin iyileştirilmesi üzerinde durur. Bunun için, iş parçacığı yönetimi ve eşzamanlı işleme gibi teknikler kullanılabilir. Ayrıca, node'lar arası iletişimin güvenliğini ve performansını artırmak için ROS2'nin sunduğu güvenlik ve QoS özelliklerinden yararlanılabilir.

Etkin Topic ve Mesaj İşleme

Etkin topic ve mesaj işleme, büyük veri setleriyle çalışırken ve verilerin doğru bir şekilde filtrelenmesi ve işlenmesi gerektiğinde önemlidir. Bu bağlamda, veri işleme stratejileri ve veri filtreleme teknikleri uygulanarak, istenmeyen verilerin sistem üzerinde gereksiz yük oluşturmasının önüne geçilebilir.

2007 yılında, Willow Garage tarafından ROS'un ilk versiyonu geliştirildiğinde, hedef, daha etkili ve hızlı veri paylaşımına olanak sağlayan bir robotik altyapı oluşturmaktı. O günden bu yana, ROS ve ROS2'nin veri işleme ve iletişim özellikleri büyük ölçüde geliştirildi

2.1.2 Kompozit ve Lifecycle Node'lar

Kompozit ve lifecycle node'lar, ROS2 sisteminin modülerliğini ve esnekliğini artırmaya yönelik olarak geliştirilen iki önemli kavramdır. Bu kavramlar sayesinde, robotik uygulamalar daha etkin bir şekilde yönetilir ve geliştirilir.

Kompozit Node Kavramı ve Kullanımı

Kompozit node'lar, birden fazla node'un tek bir süreç içerisinde çalıştırılmasını sağlar. Bu, sistem kaynaklarının daha verimli kullanılmasına ve performansın artırılmasına yardımcı olur. Ayrıca, kompozit node'lar sayesinde, benzer işlevlere sahip node'lar gruplandırılabilir ve kodun yeniden kullanılabilirliği ve bakımı kolaylaştırılır.

Şekil 2.2: Kompozit node örneği ve süreç içerisindeki node'ların bir arada çalışması (Görselde, kompozit node'un süreç içerisinde birden fazla node'u yönettiği ve bu node'ların nasıl bir arada çalıştığı gösterilmelidir)

Lifecycle Node'lar ve Yönetimi

Lifecycle node'lar, robotik uygulamaların yaşam döngüsü yönetimini sağlar. Bu sayede, uygulamanın başlangıç, çalışma, duraklatma ve sonlandırma gibi farklı evrelerinde kontrol ve yönetim sağlanır. Lifecycle node'lar, hata yönetimi ve geri kazanım stratejileri için de kullanılabilir.

Şekil 2.3: Lifecycle node'ların yaşam döngüsü evreleri ve yönetimi (Görselde, lifecycle node'ların yaşam döngüsündeki farklı evreler ve bu evrelerin nasıl yönetildiği gösterilmelidir)

2.2 DDS (Data Distribution Service) ve QoS (Quality of Service) Parametreleri

ROS2, DDS (Data Distribution Service) üzerine kurulmuştur ve bu, ROS2'nin iletişim altyapısının güçlü ve esnek olmasını sağlar. QoS (Quality of Service) parametreleri ise, iletişimin kalitesini ve güvenilirliğini artırarak, robotik uygulamaların performansını optimize etmeye yardımcı olur.

2.2.1 DDS ve ROS2 İçin Önemi

DDS, gerçek zamanlı ve büyük ölçekli sistemlerde veri paylaşımı ve iletişim için kullanılan bir standarttır. ROS2'nin DDS üzerine inşa edilmesi, ROS2'nin ölçeklenebilirliğini, performansını ve güvenliğini artırır.

DDS'nin İletişim Modeli DDS, yayıncı-abone (publisher-subscriber) iletişim modelini kullanır. Bu modelde, yayıncılar (publisher) belirli bir konu (topic) üzerinde veri yayınlar ve aboneler (subscriber) bu konuya abone olarak yayınlanan verileri alır. Bu sayede, yayıncılar ve aboneler arasında doğrudan bağlantı kurmaya gerek kalmadan, veri paylaşımı ve iletişim sağlanır.

Şekil 2.4: DDS'nin yayıncı-abone iletişim modeli (Görselde, yayıncıların ve abonelerin konular üzerinden nasıl iletişim kurduğunu gösteren bir diyagram kullanılmalıdır)

DDS, ayrıca veri önbellekleme ve keşif gibi özellikler sunarak, büyük ve dinamik sistemlerde iletişimin güvenilirliğini ve esnekliğini artırır. Bu özellikler, özellikle robotik sistemlerde önemlidir, çünkü robotlar genellikle karmaşık ve değişken ortamlarda çalışır ve bu nedenle iletişim altyapısının güçlü ve esnek olması gereklidir.

2.2.2 QoS Parametreleri ve Kullanımı

QoS (Quality of Service) parametreleri, ROS2 iletişiminin performansını, güvenilirliğini ve özelleştirilebilirliğini artırmaya yönelik olarak kullanılır. QoS parametreleri, iletişim sürecinde kullanılacak kaynakların, veri iletim sürelerinin ve güvenilirlik düzeyinin kontrol edilmesini sağlar.

QoS Profilleri ve Ayarları

QoS profilleri, belirli bir iletişim senaryosu için uygun QoS parametrelerini içeren yapılandırmaları temsil eder. Bu profiller, belirli bir uygulama veya sistem gereksinimine göre özelleştirilebilir ve hızlı bir şekilde uygulanabilir. ROS2, önceden tanımlanmış birkaç QoS profili sunar, ancak kullanıcılar kendi profillerini de oluşturabilir.

Şekil 2.5: QoS profillerinin ROS2 iletişimine nasıl uygulandığı (Görselde, QoS profillerinin ROS2 yayıncılar ve aboneler arasındaki iletişime nasıl etki ettiğini gösteren bir diyagram kullanılmalıdır)

QoS ile İletişim Kalitesini Artırma

QoS parametrelerinin doğru bir şekilde ayarlanması, ROS2 iletişiminin kalitesini ve performansını önemli ölçüde artırabilir. Örneğin, güvenilirlik (reliability) parametresi, iletişimin güvenilirliğini artırmak için kullanılabilir. Bu parametre, veri iletiminin en uygun yöntemini seçerek, veri kaybının önlenmesine veya minimize edilmesine yardımcı olur.

Başka bir örnek olarak, geçmiş (history) parametresi, yayıncıdan gelen eski mesajların saklanmasını ve yönetilmesini sağlar. Bu parametre, abonelerin bağlandıktan sonra yayıncıdan daha önce gönderilmiş olan verilere erişmesine olanak tanır. Bu, abonelerin eski verilere dayalı olarak kararlar almasına ve sistem performansının artırılmasına yardımcı olur.

Şekil 2.6: QoS parametrelerinin ROS2 iletişim kalitesini nasıl etkilediği (Görselde, farklı QoS parametrelerinin yayıncılar ve aboneler arasındaki iletişim kalitesi üzerindeki etkilerini gösteren bir diyagram kullanılmalıdır)

Sonuç olarak, QoS parametrelerinin doğru bir şekilde kullanılması ve ayarlanması, ROS2 iletişim performansını ve güvenilirliğini artırarak, robotik uygulamaların daha verimli ve sağlam hale gelmesine olanak tanır.

İletişim Güvenilirliği ve Performansı QoS parametrelerinin doğru şekilde ayarlanması, ROS2 iletişiminin güvenilirliğini ve performansını önemli ölçüde etkiler. Örneğin, gecikme (latency) ve bant genişliği (bandwidth) gibi parametreler, veri iletiminin hızını ve verimliliğini doğrudan etkiler. Bu parametrelerin doğru bir şekilde seçilmesi ve uygulanması, robotik uygulamaların daha hızlı ve daha güvenilir iletişim sağlamasına yardımcı olur.

Ayrıca, abonelerin yayıncıların gönderdiği verilere istedikleri anda erişebilmesini sağlamak için sıralama (ordering) ve öncelik (priority) gibi parametreler de kullanılabilir. Bu parametreler, yayıncılar ve aboneler arasında gerçekleşen iletişimin daha düzenli ve düzgün hale gelmesine yardımcı olur.

QoS ile Kaynak Kullanımının Optimize Edilmesi QoS parametrelerinin kullanımı, ROS2 sistemi içinde kaynak kullanımını optimize etmek için de önemli bir role sahiptir. Örneğin, veri ömrü (lifespan) parametresi, eski ve artık kullanılmayan verilerin sistemden otomatik olarak temizlenmesini sağlar. Bu sayede, kaynak kullanımı azalır ve sistem performansı artar.

Başka bir örnek olarak, abonelerin ve yayıncıların iletişim hızını düzenlemek için sıklık (frequency) veya tarama (throttle) parametreleri kullanılabilir. Bu parametreler, gereksiz veri trafiğini azaltarak ve sistem kaynaklarının daha verimli kullanılmasını sağlayarak, robotik uygulamaların daha hızlı ve daha düşük enerji tüketimiyle çalışmasına olanak tanır.

Bölüm 3

İleri Düzey Paket Oluşturma ve Yönetimi

3.1 Paket Oluşturma, Yapılandırma ve Yönetimi

3.1.1 ROS2 Paketi Oluşturma ve Yapılandırma

Paket Oluşturma ve Gerekli Dosyalar

ROS2 paketlerini oluşturmak için öncelikle ‘ros2 pkg create‘ komutunu kullanırız. Bu komut ile paketimize gerekli dosyalar ve klasörler otomatik olarak oluşturulur. Bu dosyalar arasında ‘CMakeLists.txt‘, ‘package.xml‘, ‘src‘ ve ‘include‘ gibi önemli yapılandırma ve kaynak dosyaları bulunmaktadır.

CMakeLists.txt ve package.xml Dosyaları

‘CMakeLists.txt‘ dosyası, paketimizin derleme ve bağlantı işlemleri için gerekli yapılandırmaları içerir. Bu dosyada bağımlılıklar, hedefler ve derleme seçenekleri gibi bilgiler yer alır.

‘package.xml‘ dosyası ise, paketimizin meta bilgilerini ve bağımlılıklarını içerir. Bu dosyada paket adı, sürümü, lisansı ve yazarları gibi bilgilerin yanı sıra, diğer ROS2 paketleri ile olan bağımlılıklar da belirtilir.

3.1.2 Paket İçindeki Launch Dosyaları ve Kullanımı

Launch Dosyalarının Oluşturulması ve Yapısı

Launch dosyaları, ROS2 düğümlerini başlatmak ve yönetmek için kullanılır. Launch dosyaları ‘.launch.py‘ uzantısıyla oluşturulur ve Python programlama dili kullanılarak yazılır. Bu dosyalar, düğüm yapılandırmalarını ve parametrelerini içerir ve genellikle ‘launch‘ klasörü içinde bulunur.

Paket İçindeki Birden Fazla Node’u Başlatma

Launch dosyaları sayesinde, bir paket içinde birden fazla düğümü aynı anda başlatıp yönetebiliriz. Bu işlem, launch dosyasında ‘Node‘ sınıfı kullanılarak gerçekleştirilir ve

düğümler arasındaki bağımlılıklar ve iletişim ayarları yapılandırılır.

3.1.3 ROS2 Paketleri Arasında Bağımlılıklar ve İletişim

Paket Bağımlılıklarının Yönetimi

Paketler arasındaki bağımlılıklar, 'package.xml' ve 'CMakeLists.txt' dosyalarında yönetilir. Paketler arasındaki bağımlılıklar eklenerek, hizmetler ve iletişim kanalları aracılığıyla veri alışverişi sağlanır.

İletişim ve Hizmetlerle Paketler Arasında Veri Alışverişi

ROS 2 sistemi içerisinde, paketler arasında veri alışverişi sağlamak için iki temel iletişim mekanizması kullanılır: konular (topics) ve hizmetler (services). Konular, yayıncılar (publishers) ve aboneler (subscribers) arasında asenkron veri iletimi sağlarken, hizmetler ise istemci (client) ve sunucu (server) arasında senkron veri iletimi sağlar.

Bunun yanı sıra, daha karmaşık senaryolar için eylemler (actions) adı verilen bir başka iletişim mekanizması kullanılabilir. Eylemler, uzun süreli, önceden tanımlanmış hedeflere yönelik işlemler için kullanılır ve eylem istemcisi ile eylem sunucusu arasında iletişim sağlar.

3.2 Özel Mesaj, Hizmet ve Eylem Türleri Oluşturma

ROS 2, kullanıcının özel mesaj, hizmet ve eylem türlerini tanımlamasına ve kullanmasına olanak tanır. Bu türler, paketler arasında veri alışverişinde kullanılır ve kullanıcının ihtiyaçlarına göre özelleştirilebilir.

3.2.1 Özel Mesaj Türleri

Özel mesaj türlerinin tanımlanması, belirli bir paket içerisinde yapılır. Önce, paket içinde "msg" adında bir klasör oluşturulur ve bu klasörde, özel mesaj türlerine karşılık gelen ".msg" uzantılı dosyalar tanımlanır. Bu dosyalar, özel mesaj türünün yapısını ve elemanlarını içerir. ROS 2, özel mesaj türlerini kullanarak paketler arasında veri alışverişinde bulunabilir.

3.2.2 Özel Hizmet Türleri

Özel hizmet türleri, paketler arasında istemci-sunucu iletişimi sağlamak için kullanılır. Özel hizmet türlerini tanımlamak için, paket içinde "srv" adında bir klasör oluşturulur ve bu klasörde, özel hizmet türlerine karşılık gelen ".srv" uzantılı dosyalar tanımlanır. Bu dosyalar, özel hizmet türünün isteği ve yanıtını içeren yapıları içerir. ROS 2, bu özel hizmet türlerini kullanarak paketler arasında senkron veri alışverişi gerçekleştirir.

3.2.3 Özel Eylem Türleri

Özel eylem türleri, uzun süreli, önceden tanımlanmış hedeflere yönelik işlemler için kullanılır. Özel eylem türlerini tanımlamak için, paket içinde "action" adında bir klasör

oluşturulur ve bu klasörde, özel eylem türlerine karşılık gelen ".action" uzantılı dosyalar tanımlanır. Bu dosyalar, özel eylem türünün hedef, sonuç ve geri bildirim yapılarını içerir. ROS 2, bu özel eylem türlerini kullanarak paketler arasında uzun süreli işlemler gerçekleştirir ve bu işlemler sırasında geri bildirim sağlar.

Özel mesaj, hizmet ve eylem türleri oluşturulduktan sonra, bu türlerin ROS 2 sistemi içinde kullanılabilmesi için paketin bağımlılıkları ve "CMakeLists.txt" dosyası düzenlenmelidir. Bu düzenlemelerle birlikte, özel türler ROS 2 düğümleri arasında veri alışverişi için kullanılabilir hale gelir.

Sonuç olarak, ROS 2, paketler arasında veri alışverişi sağlamak için iletişim ve hizmetlerle çalışırken, kullanıcıların özel mesaj, hizmet ve eylem türlerini tanımlayarak ve kullanarak bu alışverişi daha esnek ve özelleştirilebilir hale getirir. Bu özellik, kullanıcıların karmaşık ve özelleştirilmiş robotik uygulamalar geliştirmesine olanak tanır.

Bölüm 4

ROS2 Güvenlik (SROS2)

ROS2'nin güvenlik çözümü olan SROS2 (Secure ROS2), robotik sistemlerde güvenli iletişim sağlamak için tasarlanmıştır. SROS2, iletişim kanallarını şifreleyerek ve kimlik doğrulama ile yetkilendirme sağlayarak, sistemlerin güvenliğini artırmaktadır.

4.1 SROS2 ile Güvenlikli İletişim Sağlama

4.1.1 SROS2'nin İşlevi ve Avantajları

SROS2, ROS2 düğümleri arasındaki iletişimi güvence altına almak için kullanılır. Bu güvenlik önlemleri, robotik sistemlerin hassas verilerini ve işlemlerini koruma altına alır. SROS2'nin avantajları şunlardır:

Şifreli iletişim: Veri sızıntılarını ve izinsiz erişimi önlemek için iletişim kanallarında şifreleme kullanır.

Kimlik doğrulama: Düğümler arasındaki iletişimi gerçekleştiren katılımcıların kimliğini doğrular.

Yetkilendirme: İzin kontrolleri sağlayarak, düğümlerin belirli işlemleri gerçekleştirme yeteneğini sınırlar.

4.1.2 SROS2 İle Güvenli İletişim

SROS2, ROS2 düğümleri arasındaki iletişimi güvence altına almak için DDS-Security (Data Distribution Service Security) üzerine kurulmuştur. DDS-Security, şifreleme, kimlik doğrulama ve yetkilendirme sağlayarak, ROS2 düğümleri arasındaki iletişimin güvenliğini sağlar.

Güvenli İletişim İçin Ayarlar

Güvenli iletişim sağlamak için, ROS2 düğümleri arasında şifreleme, kimlik doğrulama ve yetkilendirme ayarlarının yapılandırılması gerekir. Bu ayarlar, güvenlik profilleri ve sertifikalar kullanılarak yapılır. Güvenlik profilleri, düğümlerin güvenlik politikalarını ve yapılandırmalarını tanımlar, sertifikalar ise kimlik doğrulama sürecinde kullanılır.

SROS2 İle İletişim Örneği

SROS2 ile güvenli iletişim kurmak için şu adımları izleyin

SROS2 araçlarını kullanarak, güvenlik profilleri ve sertifikalar oluşturun. Bu adım, düğümlerin kimlik bilgilerini ve güvenlik politikalarını tanımlar.

İlgili düğümlerde SROS2 yapılandırmasını etkinleştirin. Bu, düğümlerin güvenlik ayarlarını ve sertifikalarını kullanarak iletişim kurmalarını sağlar.

Şifreleme, kimlik doğrulama ve yetkilendirme ayarlarını uygulayarak, düğümler arasında güvenli iletişimi başlatın. Bu adımda, düğümler arasındaki veri alışverişi şifrelenir ve kimlik doğrulama ve yetkilendirme kullanılarak düğümler arasındaki iletişim sınırlandırılır.

4.2 Sertifika Yönetimi ve Güvenlik Politikaları

4.2.1 Sertifika Oluşturma ve Yönetimi

SROS2, düğümlerin kimlik doğrulamasını sağlamak için X.509 sertifikalarını kullanır. Sertifikalar, düğümlerin güvenli iletişim için gereken kimlik bilgilerini içerir. SROS2 araçları, sertifika otoritesi (CA) sertifikaları ve düğümlere ait sertifikaları oluşturmak için kullanılabilir. Sertifikalar, güvenlik ayarlarının yapılandırılmasında ve düğümlerin kimlik doğrulamasında kullanılır.

4.2.2 Güvenlik Politikaları ve Uygulama

Güvenlik politikaları, ROS2 düğümleri arasındaki iletişimde uygulanacak güvenlik kurallarını tanımlar. Bu politikalar, düğümler arasında hangi veri alışverişinin şifreli olacağını, hangi düğümlerin kimlik doğrulaması yapacağını ve hangi düğümlerin belirli işlemleri gerçekleştirmeye yetkili olduğunu belirler.

Güvenlik Politikalarını Ayarlama

Güvenlik politikalarını ayarlamak için, güvenlik profilleri kullanılır. Güvenlik profilleri, XML dosyaları olarak tanımlanır ve düğümlerin güvenlik politikalarını ve yapılandırmalarını içerir. Güvenlik politikalarını ayarlamak için, düğümlere ait güvenlik profillerini düzenleyerek, şifreleme, kimlik doğrulama ve yetkilendirme ayarlarını değiştirebilirsiniz.

Güvenlik Politikalarını Uygulama

Güvenlik politikalarını uygulamak için, düğümlerin güvenlik profilleri ile çalıştırılması gerekir. Bu, düğümlerin politikalara göre yapılandırılmış güvenlik ayarlarını kullanarak iletişim kurmalarını sağlar. Güvenlik politikalarını uygulamak için şu adımları izleyin:

Düğümlere ait güvenlik profillerini düzenleyerek, şifreleme, kimlik doğrulama ve yetkilendirme ayarlarını belirleyin.

Güvenlik profillerini ve sertifikaları ROS2 düğümlerine uygulayın. Bu, düğümlerin güvenlik politikalarına göre yapılandırılmış iletişim kurmalarını sağlar.

ROS2 düğümlerini güvenlik profilleri ve sertifikalarla başlatın. Bu adımda, düğümler güvenlik politikalarına göre şifreleme, kimlik doğrulama ve yetkilendirme kullanarak iletişim kurarlar.

Güvenlik politikalarının uygulanması, ROS2 düğümleri arasında güvenli iletişim sağlar ve robotik sistemlerin güvenliğini artırır. Bu yaklaşım, hassas verilerin ve işlemlerin korunmasına yardımcı olarak, robotik uygulamaların daha güvenilir ve güvenli hale gelmesine olanak tanır.

Bölüm 5

ROS2 ve Python ile Test ve Entegrasyon

5.1 Birim ve Entegrasyon Testleri Yazma

5.1.1 Birim Testlerinin Önemi ve Kapsamı

Birim testleri, bir yazılımın en küçük işlevsel birimlerinin düzgün bir şekilde çalışıp çalışmadığını kontrol eder. Testler genellikle yazılım geliştirme sürecinin erken aşamalarında, kodun ilk kez yazıldığı anda oluşturulur ve bu, hataların erken ve düşük maliyetle tespit edilmesini sağlar.

5.1.2 Python ve ROS2 İle Birim Testleri Oluşturma

Python'un "unittest" kütüphanesi, Python ve ROS2 ile birim testleri oluşturmanın temelini sağlar. ROS2'nin içerdiği rclpy kütüphanesi de ROS2 özelliklerinin birim testlerine olanak sağlar.

Test Süreci ve Kullanılacak Araçlar

Python ve ROS2 için test süreci, öncelikle testlerin yazılması, ardından bunların bir test koşucusu (örneğin, "unittest" kütüphanesinin içindeki test koşucusu) kullanılarak koşulması ve sonuçların değerlendirilmesi aşamalarını içerir.

Örnek Birim Test Senaryosu

Örnek bir birim test senaryosunda, bir ROS2 düğümünün başlatılması ve bir mesajın yayınlanıp yayınlanmadığının kontrol edilmesi olabilir. Bu, rclpy'nin Node ve Publisher özelliklerinin kullanılmasıyla gerçekleştirilebilir.

5.1.3 Entegrasyon Testlerinin Önemi ve Kapsamı

Entegrasyon testleri, bir yazılımın farklı bileşenlerinin birlikte düzgün bir şekilde çalışıp çalışmadığını kontrol eder. Birim testlerinden farklı olarak, entegrasyon testleri genellikle daha karmaşıktır ve genellikle birden fazla bileşenin bir araya gelmesini gerektirir.

5.1.4 Python ve ROS2 İle Entegrasyon Testleri Oluşturma

TODO

Entegrasyon Test Süreci ve Kullanılacak Araçlar

Entegrasyon test süreci, genellikle birim test sürecine benzer, ancak testler genellikle daha karmaşık olduğu için daha fazla hazırlık ve düşünme gerektirir.

Örnek Entegrasyon Test Senaryosu

Örnek bir entegrasyon test senaryosu, birden fazla ROS2 düğümünün başlatılması ve bir düğümün diğerine bir mesaj gönderip gönderemediğinin kontrol edilmesi olabilir.

5.2 Sürekli Entegrasyon ve Sürekli Dağıtım (CI/CD) Süreçlerinin Uygulanması

5.2.1 CI/CD'nin Önemi ve Kapsamı

CI/CD, yazılımın sürekli olarak test edilmesini ve dağıtılmasını sağlar. Bu, hataların daha hızlı tespit edilmesine ve yazılımın daha hızlı bir şekilde kullanıcılara ulaştırılmasına yardımcı olur.

5.2.2 CI/CD Araçları ve Uygulama Süreci

CI/CD için genellikle Jenkins, Travis CI, CircleCI ve Github Actions gibi araçlar kullanılır. Bu araçlar, yazılımın otomatik olarak test edilmesini ve dağıtılmasını sağlar.

Sürekli Entegrasyon (CI) İçin Konfigürasyon ve Uygulama

Sürekli entegrasyon, her kod değişikliği sonrası yazılımın otomatik olarak test edilmesini içerir. Bunun için bir CI aracının doğru bir şekilde konfigüre edilmesi gereklidir.

Sürekli Dağıtım (CD) İçin Konfigürasyon ve Uygulama

Sürekli dağıtım, her başarılı test sonrası yazılımın otomatik olarak dağıtılmasını içerir. Bunun için bir CD aracının doğru bir şekilde konfigüre edilmesi gereklidir.

Bölüm 6

ROS2 ve Gerçek Zamanlı Sistemler (Real-Time)

6.1 Gerçek Zamanlı Sistemlerin Önemi ve Gereklilikleri

6.1.1 Gerçek Zamanlı Sistemlerin Tanımı ve Özellikleri

Gerçek zamanlı sistemler, belirli bir süre içinde işlem yapma ve sonuç verme gerekliliği olan sistemlerdir. Bu sistemler, genellikle kontrol sistemleri, telekomünikasyon, endüstriyel otomasyon ve robot teknolojilerinde yaygın olarak kullanılır. Gerçek zamanlı sistemlerin performansı, sadece işlemlerin doğruluğuyla değil, aynı zamanda işlemlerin zamanında tamamlanmasıyla da belirlenir.

6.1.2 Gerçek Zamanlı Sistem Gereklilikleri

Gerçek zamanlı sistemlerin temel gereklilikleri, belirlenmiş bir zaman çerçevesinde veri işleme, düşük gecikme süreleri ve yüksek zaman duyarlılığı içerir. Sistem, belirtilen süre zarfında bir yanıt vermek zorundadır, aksi takdirde sistemin başarısız olduğu kabul edilir.

Zaman Kısıtlamaları ve Öncelikler

Gerçek zamanlı sistemlerde, belirli işlemler belirlenmiş bir zaman sınırı içinde tamamlanmalıdır. Ayrıca, belirli işlemler genellikle diğerlerinden daha yüksek önceliğe sahip olacak şekilde düzenlenir. Bu öncelikler, işlemlerin zamanında tamamlanmasını sağlamak için kullanılır.

Kaynak Yönetimi ve Planlama

Gerçek zamanlı sistemler, kaynakları etkin bir şekilde yönetmek ve işlemleri planlamak için karmaşık algoritmalar kullanır. Kaynaklar, genellikle belirli bir önceliğe göre tahsis edilir ve planlama, işlemlerin zamanında tamamlanmasını sağlamak için kullanılır.

6.2 ROS2 ve Gerçek Zamanlı Performans İyileştirmeleri

6.2.1 ROS2'nin Gerçek Zamanlı Özellikleri

ROS2, gerçek zamanlı sistemler için bir dizi özellik sunar. Bu özellikler arasında, belirli işlemler için öncelik ayarları, yüksek performanslı iletişim için DDS (Data Distribution Service) desteği ve gelişmiş kaynak yönetimi bulunur.

6.2.2 ROS2 Gerçek Zamanlı Performans İyileştirmeleri

ROS2, gerçek zamanlı performansını iyileştirmek için bir dizi mekanizma sunar. Bunlar arasında, işlem önceliklerini düzenleyen ve belirli işlemlerin zamanında tamamlanmasını sağlayan gerçek zamanlı işletim sistemi (RTOS) desteği bulunur.

Gerçek Zamanlı İletişim ve DDS

ROS2, gerçek zamanlı iletişim için DDS'yi kullanır. DDS, genellikle gerçek zamanlı sistemler için yüksek performanslı veri dağıtımı sağlar ve düşük gecikme süreleri ve yüksek verimlilik sunar.

Gerçek Zamanlı İşlemler ve Ayarlar

ROS2, gerçek zamanlı işlemler için bir dizi ayar sunar. Bu ayarlar, işlem önceliklerini düzenlemeyi, işlem zamanlayıcıları ayarlamayı ve belirli işlemlerin zamanında tamamlanmasını sağlamak için gerekli kaynakları tahsis etmeyi içerir.

ROS2 Gerçek Zamanlı Uygulama Örneği

ROS2 ile gerçek zamanlı bir uygulama örneği olarak, bir robotun hareketlerini kontrol eden bir sistem verilebilir. Bu sistemde, robotun hareketleri belirli bir zaman çerçevesinde düzgün bir şekilde gerçekleştirilmeli, aksi takdirde robotun güvenli ve etkin bir şekilde işlemesi mümkün olmayacaktır.

Bölüm 7

İleri Düzey TF2 Kullanımı ve Optimizasyon

7.1 TF2 İle Karmaşık Dönüşüm Zincirleri ve Optimizasyon

7.1.1 Karmaşık Dönüşüm Zincirlerinin Oluşturulması

TF2, birden çok dönüşümü kapsayan karmaşık dönüşüm zincirlerinin oluşturulmasını ve yönetimini kolaylaştırır. Bu zincirler genellikle, bir robotun farklı bileşenlerinin birbiriyle olan ilişkisini tanımlamak için kullanılır. Bir dönüşüm zinciri oluştururken, dönüşüm sırası, dönüşüm türü ve bağlantılar dikkatlice belirlenmelidir.

7.1.2 TF2 ile Dönüşüm Zincirlerinin Optimizasyonu

TF2'nin güçlü API'leri sayesinde, dönüşüm zincirlerinin optimizasyonu ve performansını artırmak mümkündür. Bu genellikle, gereksiz dönüşümlerin kaldırılması, dönüşüm hesaplamalarının etkinleştirilmesi veya dönüşüm zincirinin uygun bir şekilde düzenlenmesi ile elde edilir.

Optimizasyon Teknikleri ve Araçları

TF2'nin API'leri, dönüşüm zincirlerini optimize etmek için çeşitli teknikler ve araçlar sunar. Bu teknikler genellikle, gereksiz dönüşümlerin kaldırılması, dönüşüm hesaplamalarının etkinleştirilmesi veya dönüşüm zincirinin uygun bir şekilde düzenlenmesi gibi işlemleri içerir.

Örnek Optimizasyon Senaryosu

TF2 ile gerçekleştirilebilecek örnek bir optimizasyon senaryosu, bir robotun bir dönüşüm zinciri üzerinde hareket etmesi ve bu hareket sırasında dönüşüm hesaplamalarının en aza indirgenmesi olabilir.

7.2 TF2 Performans Analizi ve İyileştirmeleri

7.2.1 TF2 Performansını İzleme ve Değerlendirme

TF2 performansını izlemek ve değerlendirmek için çeşitli araçlar ve teknikler vardır. Bu genellikle, dönüşüm zinciri boyunca dönüşüm sürelerinin ölçülmesini ve analiz edilmesini içerir.

7.2.2 TF2 İyileştirmeleri ve Optimizasyonları

TF2 performansını artırmak için çeşitli iyileştirmeler ve optimizasyonlar yapılabilir. Bunlar genellikle, dönüşüm sürelerini azaltmayı, hesaplama verimliliğini artırmayı ve dönüşüm zincirini daha etkin bir şekilde düzenlemeyi içerir.

Performans İyileştirme Teknikleri

TF2 performansını artırmak için kullanılan teknikler genellikle, dönüşüm hesaplamalarının optimizasyonu, dönüşüm sürelerinin azaltılması ve dönüşüm zincirlerinin etkin düzenlenmesi gibi işlemleri içerir.

Örnek Performans İyileştirme Senaryosu

TF2 performansını artırmak için uygulanan bir senaryo, bir robotun dönüşüm zinciri üzerinde daha hızlı ve daha verimli bir şekilde hareket etmesini sağlamak olabilir. Bu genellikle, dönüşüm sürelerini azaltmayı ve hesaplama verimliliğini artırmayı içerir.

Bölüm 8

ROS2 ile Simülasyon ve Donanım Entegrasyonu

8.1 Gazebo Simülasyonu ve ROS2 Entegrasyonu

8.1.1 Gazebo Simülasyonunun Tanıtılması

Gazebo, bir 3D dinamik simülatör olan ve özellikle robotik uygulamalar için tasarlanmış bir yazılımdır. Gerçek dünyayı modellemek için fizik motorlarına sahiptir ve kullanıcıların robotları, sensörleri ve karmaşık ortamları 3 boyutlu olarak oluşturmasına izin verir.

8.1.2 ROS2 İle Gazebo Simülasyon Entegrasyonu

ROS2 ile Gazebo'nun entegrasyonu, simülasyon ortamında robotları test etme ve önceden belirlenmiş görevleri yerine getirme imkanı sağlar. Bu entegrasyon, ROS2'nin mesaj geçiş özellikleri ve Gazebo'nun dinamik simülasyon yeteneklerini birleştirerek karmaşık robotik sistemlerin geliştirilmesini kolaylaştırır.

Gazebo Simülasyonu İçin ROS2 Paketleri

Gazebo ile ROS2'yi entegre etmek için gazebo_ros_pkgs gibi bir dizi ROS2 paketi bulunmaktadır. Bu paketler, Gazebo'nun simülasyon yeteneklerini ROS2 ekosistemi ile kullanmak için gerekli araçları ve kütüphaneleri sağlar.

Örnek Gazebo ve ROS2 Entegrasyon Senaryosu

Gazebo ve ROS2 entegrasyonunun bir örneği olarak, bir robotun simülasyon ortamında belirli bir hedefe gitmesi ve orada bir görevi tamamlaması durumu düşünülebilir. Bu senaryo, robotun hareketini kontrol etmek ve sensör verilerini işlemek için ROS2 düğümlerini ve Gazebo simülasyon ortamını kullanır.

8.2 Farklı Robot Donanımı ve Sensörlerle Çalışma

8.2.1 ROS2 ve Robot Donanım Entegrasyonu

ROS2, farklı robot donanımlarıyla entegrasyonu kolaylaştırır. ROS2 düğümleri, sürücüler ve donanım arayüzleri kullanılarak bir dizi farklı robot donanımı kontrol edilebilir.

8.2.2 ROS2 ve Sensör Entegrasyonu

ROS2, farklı sensör türleriyle de entegre olabilir. Sensör verileri, ROS2 düğümleri ve mesajları kullanılarak toplanabilir ve işlenebilir, bu da robotik sistemlerin geniş bir sensör yelpazesinden veri almasını sağlar.

Donanım ve Sensör Entegrasyonu İçin ROS2 Paketleri

ROS2, farklı robot donanımları ve sensörlerle entegrasyonu kolaylaştıran çeşitli paketler sağlar. Bu paketler, sürücüler, donanım arayüzleri ve sensör verilerini işlemek için gereken araçları içerir.

Örnek Donanım ve Sensör Entegrasyon Senaryosu

Bir örnek senaryo olarak, bir ROS2 düğümü, bir lidar sensöründen gelen verileri toplayabilir ve bu verileri robotun çevresini algılamak ve haritalamak için kullanabilir. Bu tür bir entegrasyon, robotun çevresini daha iyi anlamasına ve daha etkili bir şekilde hareket etmesine olanak sağlar.

Bölüm 9

İleri Düzey Araçlar ve Kullanımları

9.1 ROS2 CLI ve Gelişmiş Komutlar

9.1.1 ROS2 CLI'nin Genel Bakışı

ROS2 CLI ve kullanım amacı hakkında detaylı bilgi.

9.1.2 Gelişmiş ROS2 CLI Komutları

İleri düzey ROS2 CLI komutları ve kullanım örnekleri hakkında detaylı bilgi.

Komut Grupları ve Kullanımları

Farklı komut grupları ve kullanımları hakkında detaylı bilgi.

Örnek Gelişmiş CLI Kullanım Senaryosu

İleri düzey bir ROS2 CLI kullanım senaryosu ve uygulanması hakkında detaylı bilgi.

9.2 Performans Analizi ve Optimizasyon Araçları

9.2.1 ROS2 Performans Analizi Araçları

ROS2 performans analizi için kullanılan araçlar hakkında detaylı bilgi.

9.2.2 ROS2 Optimizasyon Araçları

ROS2 performansını optimize etmek için kullanılan araçlar hakkında detaylı bilgi.

Performans Analizi ve Optimizasyon Teknikleri

Performans analizi ve optimizasyon teknikleri hakkında detaylı bilgi.

Örnek Performans Analizi ve Optimizasyon Senaryosu

Bir performans analizi ve optimizasyon senaryosu ve uygulanması hakkında detaylı bilgi.

9.3 Gelişmiş RViz ve RQT Eklentileri

9.3.1 RViz İleri Düzey Özellikler ve Eklentiler

İleri düzey RViz özellikleri ve eklentiler hakkında detaylı bilgi.

9.3.2 RQT İleri Düzey Özellikler ve Eklentiler

İleri düzey RQT özellikleri ve eklentiler hakkında detaylı bilgi.

Eklenti Geliştirme ve Kullanımı

ROS2 için gelişmiş eklenti geliştirme ve kullanımı hakkında detaylı bilgi.

Örnek Gelişmiş Eklenti Kullanım Senaryosu

Gelişmiş bir eklenti kullanım senaryosu ve uygulanması hakkında detaylı bilgi.

Bölüm 10

İleri Düzey ROS2 Uygulamaları ve Örnek Projeler

10.1 Otonom Navigasyon ve Haritalama Sistemleri

10.1.1 Otonom Navigasyonun Temel Bileşenleri

Otonom navigasyon sistemlerinin temel bileşenleri ve işleyişi hakkında detaylı bilgi.

10.1.2 Haritalama ve Yol Planlama Algoritmaları

Haritalama ve yol planlama algoritmaları hakkında detaylı bilgi.

İleri Düzey Navigasyon Paketleri

İleri düzey ROS2 navigasyon paketleri ve kullanımı hakkında detaylı bilgi.

Örnek Otonom Navigasyon Projesi

Otonom navigasyon için örnek bir proje ve uygulanması hakkında detaylı bilgi.

10.2 Robot Kontrol ve Planlama Algoritmaları

10.2.1 Kontrol Algoritmaları

İleri düzey robot kontrol algoritmaları hakkında detaylı bilgi.

10.2.2 Planlama Algoritmaları

İleri düzey robot planlama algoritmaları hakkında detaylı bilgi.

İleri Düzey Kontrol ve Planlama Paketleri

İleri düzey ROS2 kontrol ve planlama paketleri ve kullanımı hakkında detaylı bilgi.

Örnek Robot Kontrol ve Planlama Projesi

Robot kontrol ve planlama için örnek bir proje ve uygulanması hakkında detaylı bilgi.

10.3 İnsan-Robot Etkileşimi ve Yapay Zeka Entegrasyonu

10.3.1 İnsan-Robot Etkileşimi

İleri düzey insan-robot etkileşimi konseptleri ve teknikleri hakkında detaylı bilgi.

10.3.2 Yapay Zeka Entegrasyonu

Yapay zeka entegrasyonu ve uygulamaları hakkında detaylı bilgi.

İleri Düzey İnsan-Robot Etkileşimi ve Yapay Zeka Paketleri

İleri düzey insan-robot etkileşimi ve yapay zeka entegrasyonu için ROS2 paketleri ve kullanımı hakkında detaylı bilgi.

Örnek İnsan-Robot Etkileşimi ve Yapay Zeka Projesi

İnsan-robot etkileşimi ve yapay zeka entegrasyonu için örnek bir proje ve uygulanması hakkında detaylı bilgi.

Bölüm 11

Özet ve Sonuç

11.1 Öğrenilenlerin Özeti

Bu içerikte öğrenilen ileri düzey ROS2 ve Python konularının özeti ve genel değerlendirmesi.

11.1.1 Ana Konular ve Kazanımlar

Öğrenilen ana konuların ve kazanımların kısa bir özeti.

11.1.2 İçeriğin Önemi ve Değeri

İçeriğin önemi ve değeri hakkında genel değerlendirme.

11.2 İleride Yapılacaklar ve Öneriler

İleri düzey ROS2 ve Python kullanımıyla ilgili gelecekte yapılacaklar ve öneriler.

11.2.1 Potansiyel Gelişmeler ve Araştırmalar

ROS2 ve Python kullanımıyla ilgili potansiyel gelişmeler ve araştırmalar.

11.2.2 İleri Düzey Projeler ve Uygulamalar

İleri düzey ROS2 ve Python projeleri ve uygulamaları için öneriler.