Bu dosyada yer alan bilgiler için yararlanılan video kaynağına ulaşmak için tıklayınız...

.urdf formatı, şeklimizi RVIZ da tanımlayan ve sadece eklemlerini gösteren yapıdır. (Örneğin bir arabamız da 4 teker ve bir sensörümüz var burada her bir tekere ve sensöre ayrı bir şekilde eklem diyeceğiz) RVIZ formatında her bir eklemin nasıl hareket edeceği, manuel olarak tamınlanabilir. Kodumuza entegre edeceğimiz plug yapısı ile her tekere ve her sensöre ayrı bir parametre verilip kontrol edilebilir.

Bu **.urdf** formatı Gazebo için değil. Gazeboya aktarırken launch dosyaları ile birlikte robotun tüm hareketlerini ve sensör bilgilerini açacağımız port ile alabiliriz. Bu sayede robotu gerçek dünya ortamında simule edebiliriz.

Bir robotun modelini AutoCAD, Solid Works, Blender ve benzeri CAD uygulamaları ile çizebilirsiniz fakat simulasyon çok daha farklı bir kavramdır. Bir robotun dünyada nasıl hareket edeceğini ve kusurlarını tespit için kullanılır. Biz robot gelişimi için ROS ortamı ve **.urdf(universial robot description format)** kullanarak yapacağız.

Robot Modelleme için ROS paketleri

robot_model : Bu paketin için de tüm önemli paketleri görebiliriz.

- **urdf :** Robotu tarif eden bir .xml dosyasıdır ve C++ çözümleyicisi vardır. Urdf ile sadece katı uzuvlara sahip bir model tanımlanabilir

-**joint_state_publisher :** Bir kullanıcı bunu kullanarak bir robotun eklemleriyle iletişime girebilir. Bu yapı bize bir eklemin hareketlerinin bilgisini, durumunu döndürür veya bir subscribe olarak bizden bu durumu alır.

-**kdl_parser :** Kinematik ve Dinamik kütüphanedir(KDL). Parser araçlarını içeren bir pakettir.

-**robot_state_publisher :** Mevcut robot eklem durumlarını okur ve bir robot uzvunun 3D görüntülerini yayınlar.

-**xacro :** XML Makroları dır. URDF'i daha okunur hale getirir ve karmaşik robot tanımlamalarını oluşturmak için kullanılır.(ROS, **xacro** formatını **xml**'e dönüştürebilir)

URDF, robotun kinematik ve dinamik tanımı, robotun görsel temsilini ve robotun çarpışma modelini temsil eder.

URDF XML Yapısı

- Bir robotun tüm özelliklerini gösterir(<robot> … <robot> en üstteki tüm herşeyi kapsayan tag)

<robot name = “robotum”> // robot tanımlanır

<link> ……. <link> // uzuvlar tanımlanır

<link> ……. <link>

<joint>…….<joint> // eklemler tanımlanır. Linkleri birleştirir. Genel bir kod yapısı

<joint>…….<joint>

<transmission> // eklemleri hareket etmesini sağlayan yapı

</robot>

- **sensor/proposals** : kamera, lazer sensörü vb. Sensörleri tanımlar

- **link** : bir uzvun kinematik ve dinamik özelliklerini tanımlar

- **transmission** : aktarıcılardır ve aktüatör, şanzımanları uzuvlara bağlar, mekanik yapısı temsil eder

- **joint** : Bir eklemin kinematik ve dinamik özelliklerini tanımlar

- **gazebo** : simulasyon özellikleri tanımlanır

- **sensor** : sensör tanımlar

- **model_state** : bir modelin durumunu belirli bir zamanda tanımlar

- **model** : bir robotun, kinematik ve dinamik özelliklerini tanımlar

<link name = …..>

<inertial> // kütlesel olarak atalet matrisi tanımlamaları yapılır

<origin xyz = “0 0 0.5”(konumları) rpy =”0 0 0”(yaw pitch roll eksenleri)/>

……..

</inertial>

<visual>……</visual>

<collision>…..</collision> // çarpışma alanını tanımlatır

</link>

- <geometry> : <link><visual> içinde kullanılması zorunludur. Cismimizin şeklini tanımlar(box,clyinder vb…)

- Bir sistemi oluştururken mesh yapıları olabildiğince en az yapıda tutulmalıdır.

- .DAE yapısı mesh dosya yapısını temsil eder.

- Joint, bir robotun eklemini tarif eder. Bu eklemi tarif ederken bir Parent_link ve Child_link yapısı vardır.(<parent link = “link1”/> <child link = “link2”/>)

limit effort : hareketlerin sınırlamalarını tanımlar.

<Joint type = …..> : revolute, continuous, prismatic, fixed, floating, planar.(hareket eksenleri, bu typlardan birini kullanmalıyız)

<joint><axis> : yapısı şeklin hangi eksen boyunca dönebileceğini belirtir. (1,0,0(varsayılan))

- <joint> yapısının bunun gibi birçok özelliği vardır. Araştırılarak bulunabilir.

<transmission>(Transmisyon) Elemanı

Şimdiye kadar araca sınırlar ve hareket etmesi adına tanımlamalar yaptık. Aracı harekete geçirecek yapı işe bu kısımdır.

Transmisyon elemanı, bir aktüatör(mekanik hareketi sağlayan motor) ile mafsal(eklem) arasındaki ilişkiyi tanımlamak için kullanılan URDF robot tanımlama modelinin uzantısıdır. Bir aracımız ve 4 tekerimiz var. Biz transmission ile bu araç ve tekerin kesistiği yere sanal bir aktüatör yerleştirmiş oluyoruz ve onu harekete geçiriyoruz.

```
 <transmission name="left_rear"> // bu transmisyona adını verdik
  <type>transmission_interface/SimpleTransmission</type>
 <joint name="body_to_left_rear">//transmisyonu aktifleştirdik ve bu isme bağladık
<hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
// Yukarıda dairesel,continuous bir dönme hareketini verecek bir yapı tanımladık
  </joint>
  <actuator name="body_to_left_rear_motor">
   <mechanicalReduction>1</mechanicalReduction> // Burada ne verildiyse o kadar ve demiş oldu. Aradaki 1’i 50 yapsaydık, 1000 verirsek 20 ver demiş olurduk
  </actuator>
 </transmission>
```

-Bir sistemi oluştururken, rviz, urdf, launch, meshes vb. Klasörler açılabilir.

-Biz robotu **.urdf** verileri ile Rviz de görebiliyor ve bir robotun eklemlerini oluşturabiliyoruz. Fakat bu yapıyı Gazebo da kullanabilmek için **URDF'i XACRO** dosyasına çevirmeliyiz.

- URDF formatı, katı bir haldir herhangi bir sürekli ve paralel hareket edemez.

- URDF'i yeniden programlamayız ve bize esneklik sağlamaz, örneğin bir tekerleği 4 defa kullanacaksak bunu 4 defa oluşturmalıyız, fakat xacro da bunu oluşturmak çok daha kolaydır. Bir tekeri **.urdf** ile bir defa oluştururusun ve buna 'Teker' adını verirsin. Daha sonra XACRO da 4 defa 'Teker' fonksiyonunu kullanarak 4 teker elde edersin. Sistemimiz karmaşıklaştıkça XACRO formatı kullanılır.

-URDF tek bir dosyadır ve içine başka dosyalar eklenemez.

-XACRO urdf in temizlenmiş sürümüdür, yaptığı şey robot tanımlamasında makrolar oluşturmak ve bu makroları yeniden kullanmaktır. **İstenilirse XACRO'lar bazı ros araçlarıyla kolayca urdf formatına dönüştürülebilir.**

-URDF de tanımladığımız bir sabiti xacro ile belirtebilir ve istenildiğinde değiştirilebilir

-URDF de matematiksel ifade kullanılmaz fakat xacro da kullanılır.

-YAML bir veri değişim formatıdır. Konfigurasyon dosyaları için ve verilerin saklandığı veya iletildiği uygulamalarda kullanılır.

- (Bilgisayar tanımı gelecek) :~/learn_ros2_ws/src/my_first_pkg/urdf$ urdf_to_graphiz my_car.urdf bu yapı ile urdf dosyasının şematiğini pdf olarak veriyor.

- base_footprint, robotun yerdeki konumunun bir temsilidir. Aracın engellerden kaçınması ve tüm erleşimi için gerekli ve önemli olan yapıdır.

- Bütün birimler, metre, saniye, kilogram cinsinden dir.

- property tanımlaması ile sabit değerler tanımlanır.

<property name="M_PI" value="3.141592" /> ….. gibi her parametriyi bu sekilde parametre olarak tanımlarsak, kodu daha esnek bir yapıya kavuşturabiliriz.

GAZEBODA Sensör ekleme Hokuyo(lazer) sensörü

- Sensörlerin eklenmesi daha kolaydır çünki genelde fixed olarak tanılanır ve sadece konumu tanımlanır. Bunun için link ve joint olarak tanımlanması yeterlidir fakat sensörleri gazebo için tanımlamak biraz daha karmaşıktır.

<origin xyz="${base_radius-hokuyo_size/2} 0 ${base_height+hokuyo_size/4}" rpy="0 0 0" /> gibi

- sensör oluştururken önemli olan gazebo dur. Gazebonun bize verdiği parametreler dikkate alınır. Bir publisher veriyi yayınlayacak ve biz dinleyeceğiz.(urdf/XML/sensor/proposals), Gazebo tarafında ise API referanslarından sensor lere bakmak gerekli.

- <plugin name=”gazebo_ros_head_hokuyo_controller” filename=”libgazebo_ros_laser.so” />

<topicName>/scan</topicName>

<frameName>hokuyo_link</frameName>

</plugin> // Bu ifadede “libgazebo_ros_laser.so” plugin dosyası üzerinden haberleşiyor. ROS gazebo üzerinde çalışırken bize yayın yapacak ve bu “scan” topic adı üzerinden gözükecek

- daha detaylı bilgiler için “ros_plugin” ile alabilirsiniz. <plugin> yapısı içinde ki parametreler detaylı olarak incelenmeli.

- bir hız için, x,y,z de maksimum ve minumum hız büyüklüklerinin ne olacağı, sarsıntının, tork, ivmenin vb tüm parametrelerin en olacağının tanımlamaları yapılır. Bu parametreler detaylı olarak incelenmeli.(En başta verilen kaynakta 8.video 18. dk ve sonrası)

Dosya Yapısı

gazebo.launch dosyası ROS1 için, gazebo.launch.py dosyası ROS2 içindir.