Robot Guidance with Head Movements

Versiyon : v4.0
Tarih : 02/04/2021
Hazırlayan : Elif GENÇ

DÖKÜMAN REVİZYON SAYFASI

VERSIYON	TARİH	SAYFA	AÇIKLAMA
v1.0	31/08/2020	10	Ubuntu Bionic Beaver ve Ros Melodic Kurulumu, Ros Beginner Tutorials
v2.0	06/09/2020	14	Publisher/Subscriber ve Service/Client yapılarının Turtlesim üzerinde örnekleri
v3.0	18/10/2020	11	Turtlebot3 Waffle Pi modelinin kurulması, Gazebo'da bir simülasyon ortamı tasarlanması, Turtlebot3 Waffle Pi modelinin bu ortamda çağrılmasını sağlayacak başlatma dosyalarının hazırlanması ve robotun bu ortamda klavye ile kontrolünün sağlanması
v4.0	02/04/2021	10	Publisher-Subscriber ve Service-Client yapılarını öğrenmek için TurtleBot3 waffle pi robotunun yaricapi x metre olan bir çember boyunca boş dünyada hareketini gerçekleştirmek

İÇİNDEKİLER

1. TANITIM VE KAPSAM

Bu çalışma kapsamında bizden istenilen Publisher - Subscriber ve Service - Client yapısını anlamak ve bu yapıları kullanarak TurtleBot3 Waffle Pi modeli üzerinde bir uygulama gerçekleştirmektir. Düğüm (node)'ler, birbiriyle haberleşen, programlanabilir bağlantılar olarak bilinir. [1] Publisher program içerisinde mesajı yayınlayan düğüm iken, Subscriber ise Publisher' ın gönderdiği mesajları alan ve okuyan düğümdür. Publisher ve Subscriber düğümlerinin kullandığı mesaj dosyaları ROS alanları içerisinde kullanılan mesajları içeren basit yazı dosyalarıdır. Bu mesajlar ve düğümler farklı kodlama dilleriyle yazılabilir ve kullanılabilir. [2]

Servis, Client' tan gelen mesajlara göre bir çıktı üreterek bu çıktıyı Client düğümüne geri gönderen bir düğümdür. Bu servis düğümünü bir kere çalıştırarak Client' tan gelen birden çok isteğe cevap veren bir yapısı vardır. Client ise Servis' te yapılması istenilen işlemler için mesajları gönderen düğümdür.

2. KULLANILAN MATERYALLER

Çalışmalarımız boyunca Publisher-Subscriber ve Service-Client yapıları için kullanacağımız dosya türleri; msg ve srv dosyalarıdır.

Çalışmalar kapsamında kullanacağımız düğümler, msg dosyalarını kullanır. msg dosyaları ROS alanları içerisinde kullanılan mesajları içeren basit yazı dosyalarıdır. msg dosyaları, paketler içinde yer alan msg kütüphanelerinde tutulur. msg' ler, mesajların döndüğü alanların tipi ve bu alanların isimlerini tutar. Bu alan tipleri aşağıdaki listedeki gibidir. [3]

int8, int16, int32, int64

float32, float64

string

time, duration

srv dosyaları da tıpkı msg' ler gibi bir içeriğe sahiptir ve bu içerikler ileti ve dönüt şeklinde iki kısımdır. Bu iki kısım dosya içerisinde "---" şekliyle ayrılmıştır. Aşağıda bir srv dosya örneği gösterilmektedir.

int 64 data

---

int 64 response

Yukarıdaki örnek dosya içeriğinde data ileti (request), response ise dönüt' tür (response). [3]

3. ÖNERİLEN YÖNTEM / PROBLEM ÇÖZÜM AŞAMALARI

Ros komutlarının çalıştırılamamasıdır. Bu problem, yapılan araştırmalar sonucunda ROS paketlerinin güncel olmadığından kaynaklanan bir problem olduğu anlaşıldı. Daha sonra bu problemi nasıl çözeceğimiz ele alındı ve ROS paketlerini güncelleyebilmek için "$ sudo apt-get update" ve "$ sudo apt-get upgrade" komutları kullanılarak sistem ve ROS paketleri güncellenmiştir ve problem ortadan kalkmıştır.

4. GERÇEKLENEN GÖREVİN AŞAMALARI

- Öncelikle msg dosyalarının oluşturulması için msg isimli bir klasör oluşturulur. Bu klasör içerisine bir text dosyası açılır ve bu dosya içerisine tipi int64 ve ismi de "yaricap" olan bir değişken tanımlanır.

```
$ roscd besinci_gorev
$ gedit
```

Daha sonra dosya "yaricap.msg" olarak kaydedilir. Şekil x'te msg dosya içeriği gösterilmektedir.

src > besinci_gorev > msg > yaricap.msg

```
1  int64 yaricap
2
```

Şekil 1: msg dosyası

- Publisher düğümü oluşturmak için öncelikle "scripts" isimli bir klasör oluşturuldu. Bu klasör içerisinde bir text dosyası açılır. Text dosyası içerisinde bir "publishYaricap" isimli bir publisher düğümü oluşturulur. Düğümün çalışması sonlandırılmadığı süre boyunca bu düğüm "yaricap_topic" isimli topic'e 10'ar saniye aralıklarla yarıçap verisini gösterir. Publisher'ın kullanacağı düğüme "publisherYaricap" ismini verdik. Bu düğümün kullanacağı topic'e ise "yaricap_topic" ismi verildi.

src > besinci_gorev > scripts > publisher.py > ...

```
1  #!/usr/bin/env python
2  # -- coding: UTF-8 --
3
4  import rospy
5  from besinci_gorev.msg import yaricap
6
7  rospy.init_node('publisherYaricap', anonymous= True)
8  pub = rospy.Publisher('yaricap_topic', yaricap, queue_size=10)
9  rate = rospy.Rate(10)
10
11 while(not rospy.is_shutdown()):
12     bilgi = yaricap()
13     bilgi.yaricap = 2
14     rospy.loginfo(bilgi)
15     pub.publish(bilgi)
16     rate.sleep()
```

Şekil 2: Publisher Kodları

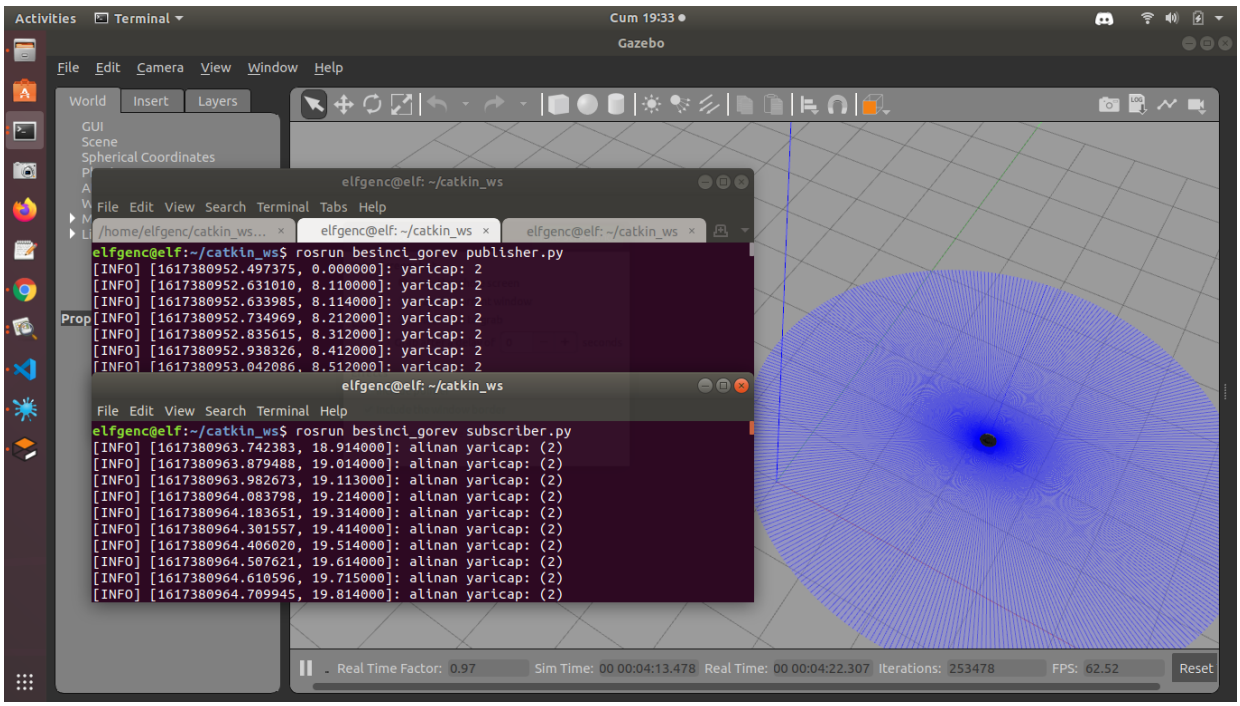
- Subscriber düğümünü oluşturmak için bir text dosyası açılır. Oluşturduğumuz dosya içerisinde bir subscriber bir de publisher düğümü eklenilir. Subscriber düğümü "yaricap_topic" isimli topic'ten veri alırken publisher da "/cmd_vel" isimli topic'e doğrusal ve açısal hız değerlerini gönderir.

src > besinci_gorev > scripts > subscriber.py > ...

```python
#!/usr/bin/env python
#---coding:UTF-8--
import rospy
from geometry_msgs.msg import Twist
from besinci_gorev.msg import yaricap
import sys

def getAndSendData(data):
    rospy.loginfo('alinan yaricap: (%d)',data.yaricap)
    twist.linear.x = 0.5
    twist.angular.z = 0.5/data.yaricap
    pub.publish(twist)

rospy.init_node('subscriberYaricap',anonymous= True)
pub = rospy.Publisher('/cmd_vel',Twist,queue_size= 1)
twist = Twist()

rospy.Subscriber('yaricap_topic', yaricap, getAndSendData)
rospy.spin()
```

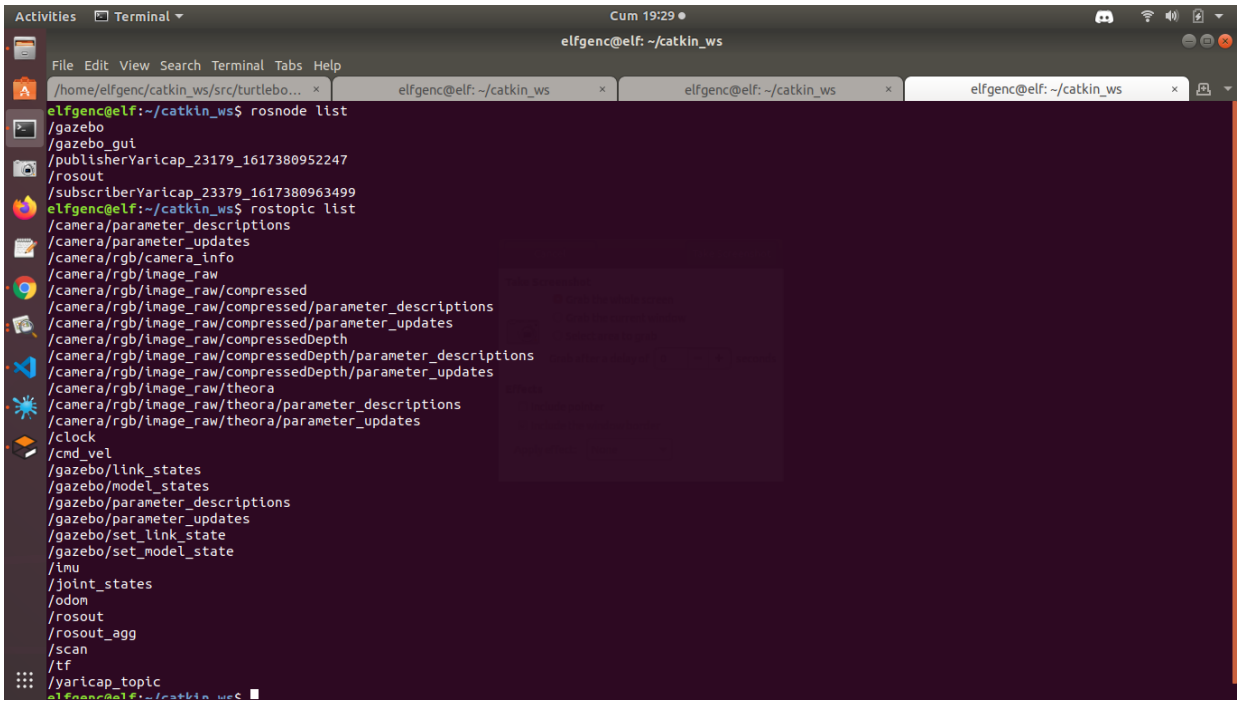
Şekil 3: Subscriber Kodları

- Boş bir simülasyonda öncelikle robotumuzu çalıştırdık. Daha sonra subscriber.py ve publisher.py dosyalarını rosrun komutu ile çalıştırdığımızda şekil x'teki görüntü elde edilmektedir.



Şekil 4: Publisher ve Subscriber Yapılarının Çalıştırılması

- Kullanılan node ve topic listesine baktığımızda oluşturduğumuz düğümler ve topicler görülmektedir.

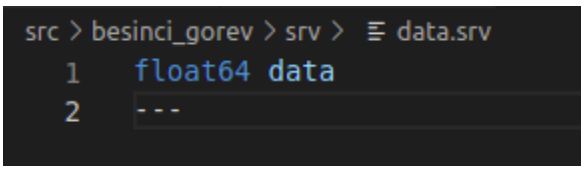


Şekil 5: Kullanılan topic ve node listesi

- Öncelikle srv dosyalarının oluşturulması için srv isimli bir klasör oluşturulur. Bu klasör içerisine bir text dosyası açılır ve bu dosya içerisine tipi float64 ve ismi de “yaricap” olan bir değişken tanımlanır.

  $ roscd besinci_gorev
  $ mkdir srv
  $ gedit

Daha sonra dosya “data.srv” olarak kaydedilir. Şekil x’te srv dosya içeriği gösterilmektedir.



Şekil 6: srv dosyası

- Servis düğümü için bir text dosyası açtık. Oluşturduğumuz dosya içerisine aşağıdaki kodları yazdık. “service_yaricap” isimli bir servis düğümü oluşturduk. Bu servis gelen float64 tipindeki “data” isimli bilgi ile doğrusal ve açısal hız elde ederiz. Bir publisher düğümü kullanarakta “/cmd_vel” isimli topic’e bu bilgiler gönderilir.

src > besinci_gorev > scripts > service.py > ...

Set as interpreter

```python
#!/usr/bin/env python
# -- coding: UTF-8 --

import rospy
from geometry_msgs.msg import Twist
from besinci_gorev.srv import data

def getAndSendData(bilgi):
    rospy.loginfo('alinan yaricap: (%2f)',bilgi.data)
    twist.linear.x = 0.5
    twist.angular.z = 0.5/bilgi.data
    pub.publish(twist)
    return

rospy.init_node('service_server',anonymous=True)
service = rospy.Service('service_server', data, getAndSendData)
twist = Twist()
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
rospy.spin()
```

Şekil 7: Service Kodları

- Client düğümü için bir text dosyası açtık. Oluşturduğumuz dosya içerisinde “service_client” isimli bir client düğümü oluşturduk. Client düğümü servis düğümüne veri götüren ve servisin döndüğü veriyi alan düğümdür.

src > besinci_gorev > scripts > client.py > ...

Set as interpreter

```python
#!/usr/bin/env python
# -- coding: UTF-8 --

import rospy
from geometry_msgs.msg import Twist
from besinci_gorev.srv import data,dataRequest
import sys

rospy.init_node('service_client')
rospy.wait_for_service('service_server')
client = rospy.ServiceProxy('service_server',data)
request_object = dataRequest()
request_object.data = 2
result = client(request_object)
```

Şekil 8: Client Kodları

5.Sonuçlar

Yapılan çalışmalar kapsamında Publisher-Subscriber ve Service-Client yapılarını, bu yapıların gerektirdiği ayarlamaların nasıl yapıldığını ve bu yapıları kullanarak Turtlebot3 Waffle Pi robotu üzerinde çalışmalar yapılmıştır. Msg ve srv yapısını kullanabilmek için package.xml ve CMakeLists.txt dosyalarında gerekli ayarlamaların ve bu yapılar sayesinde terminaller arası iletişimin nasıl gerçekleştirildiği öğrenildi. Bu yapıları kullanarak verileri düğümler arasında aktarılması sağlanıldı. Turtlebot3 Waffle Pi robotu bu yapılar içerisinde yapılan işlemde üzerinde çalışırken arka planda kullanılan düğüm, servis ve topic listeleri incelendi.

6. KAYNAKÇA

[1] https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/ [Erişim tarihi: Ekim, 2020]

[2] https://medium.com/@cennttceylnn/gazebo-nedir-ros-nedir-19983c017818 [Erişim tarihi: Ekim, 2020]

[3] https://pngio.com/images/png-a619904.html [Erişim tarihi: Ekim, 2020]