TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DURUM MAKİNESİ İLE ROBOT DAVRANIŞ MODELLEME

20011030 — Muhammed Kasap

BILGISAYAR PROJESİ

Danışman
Dr. Ögr. Üyesi Erkan Uslu

Ocak,2024

TEŞEKKÜR

Bu proje sürecinde bana rehberlik ederek ve değerli önerileriyle beni destekleyerek projemi başarıyla tamamlamama katkı sağlayan danışman hocam Dr. Ögr. Üyesi Erkan Usluya teşekkür ediyorum. Projenin her aşamasında sunduğu bilgi birikimi ve deneyim sayesinde, durum makinesi kullanarak robot davranışlarını modelleme konusundaki hedeflerime daha etkili bir şekilde ulaşabildim.

Sizinle çalışmak, hem mesleki hem de kişisel gelişimime önemli katkılarda bulundu. Projedeki başarılarımın arkasındaki desteğiniz ve rehberliğiniz için size minnettarım. Her daim gelişme gösterememiş olmam konusunda, takındığınız o sakin ve pozitif tutum beni önümdeki süreçlere daha istekli hale getirdi. Siz olmadan bu başarıyı elde etmek mümkün olmazdı.

Yardımları ve ilgisi için çok teşekkür ederim

Muhammed Kasap

İÇİNDEKİLER

KISALTMA LİSTESİ

ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

DURUM MAKİNESİ İLE ROBOT DAVRANIŞ MODELLEME

Muhammed Kasap

Bilgisayar Mühendisliği Bölümü
Bilgisayar Projesi

Danışman: Dr. Ögr. Üyesi Erkan Uslu

Günümüzde robot davranışlarını modelleme konusunda yaygın olarak kullanılan 2 ana yaklaşım mevcut, birisi davranış ağacı ile robot davranış modelleme, diğeri ise durum makinesi ile robot davranış modellemedir. Projemiz, ROS (Robot Operating System), Gazebo simülasyon ortamı ve Python programlama dili üzerinde SMACH kütüphanesi kullanarak, robot davranışlarını bir durum makinesi kullanarak modellemeyi hedefledi. Bu modellemeyi gösterebilmek adına bir örnek bir görev dizisini yerine getirmeyi amaçladık. Görev dizimiz bir etrafında dönme ile başlayıp, daha sonra içerisine konduğu haritanın henüz keşfedilmemiş noktalarını keşfeden bir keşif aşaması içerdi. Keşif bittiğinde başlangıç noktasına geri dönerek görev dizisini bitirdi. Bu görevleri yaparken kullanacağımız , farklı boyutlarda 3 harita ve sayısız başlangıç noktası vardır.

Anahtar Kelimeler: ROS (Robot Operating System), Gazebo, SMACH, Durum Makinesi, Görev Senaryosu

ABSTRACT

ROBOT BEHAVIOR MODELLING WITH STATE MACHINES

Muhammed Kasap

Department of Computer Engineering
Computer Project

Advisor: Assist. Prof. Dr. Erkan Uslu

Currently, two primary approaches are widely employed in modeling robot behaviors: one involves using a behavior tree, while the other utilizes a state machine.

Our project focuses on modeling robot behaviors using the SMACH library on the Robot Operating System (ROS), Gazebo simulation environment, and the Python programming language. To illustrate this modeling approach, we aimed to execute a sample task sequence.

The task sequence commenced with the robot rotating around a specific point, followed by an exploration phase where the robot systematically uncovered points on the map. After completing the exploration, the robot returned to its initial position, thereby concluding the task sequence.

Throughout these tasks, we had access to various maps of different sizes and multiple starting points.

Keywords: ROS (Robot Operating System), Gazebo, SMACH, State Machine, Task Scenario

1
Giriş

Günümüzde, robotik sistemlerin çeşitliliği ve karmaşıklığı arttıkça, bu sistemleri etkili bir şekilde programlamak ve kontrol etmek giderek daha önemli hale gelmektedir. Bu bağlamda, Robot Operating System (ROS) ve Gazebo simülasyon ortamı, robot programlama süreçlerini optimize etmek ve geliştirmek için güçlü araçlar haline gelmiştir. Bu proje, ROS ve Gazebo kullanarak, durum makinelerini entegre ederek, robotun karmaşık bir görevi daha etkili bir şekilde yerine getirmeyi amaçlamaktadır.

1.1 Gereklilikler

ROS ve Gazebo Bilgisi Proje ekibi, ROS ve Gazebo simülasyon ortamlarını etkili bir şekilde kullanabilmek için bu teknolojilere sağlam bir hakimiyete sahip olmalıdır.

Durum Makinesi Kütüphanesi Projenin başarılı bir şekilde tamamlanabilmesi için uygun durum makineleri kütüphanesi seçilmeli ve bu kütüphane üzerinde gerekli bilgiye sahip olunmalıdır.

Robot Modeli Geliştirilecek durum makineleri, simülasyon ortamında bir robot modeli üzerinde test edilecektir. Bu nedenle, kullanılacak robot modelinin URDF dosyası ve Gazebo simülasyonu için uygun konfigürasyonları sağlanmalıdır.

1.2 Amaç

Bu projenin temel amacı, ROS ve Gazebo kullanarak, durum makinelerini entegre ederek, robotlun karmaşık bir görevi daha etkili bir şekilde yerine getirmesini sağlamaktır. Durum makinesi, robot programlamasını modülerleştirerek, yazılımın esnekliğini ve sürdürülebilirliğini artırabilir.

1.3 Hedef

1. ROS ve Gazebo üzerinde durum makinelerini entegre etmek.
2. Belirlenen bir robot modeli üzerinde durum makinelerini kullanarak karmaşık bir görevi gerçekleştirmek.
3. Geliştirilen sistem ile geleneksel programlama yöntemleri arasında performans karşılaştırmaları yapmak.
4. Durum makinelerinin robot programlamasındaki avantajlarını ve olası zorlukları değerlendirmek.
5. Sonuçları raporlamak ve elde edilen bulgulara dayanarak gelecekteki robot programlama projeleri için önerilerde bulunmak.

1.4 Motivasyon

Bu gereklilikler, amaç ve hedefler çerçevesinde gerçekleştirilecek proje, robotik sistemlerin programlama ve kontrol süreçlerinde durum makinelerinin etkili bir biçimde nasıl kullanılabileceğini anlamamıza olanak tanıyacaktır. Bu bilgi, gelecekteki robot programlama projelerinde kullanılabilir ve robotik sistemlerin daha güvenilir ve esnek bir şekilde yönetilmesine katkıda bulunabilir.

2
Ön İnceleme

Robot teknolojisi hızla ilerlerken, boyutları küçülen, maliyetleri düşen, daha kolay temin edilebilir hale gelen ve yüksek verimlilik ile etkinlik sağlayan donanımsal yapılar ortaya çıkmaktadır. Bu gelişmeler, robot uygulama alanlarını genişletmektedir. Ancak, bu yaygınlaşan robot uygulamaları çeşitli sorunların çözümünü gerektirmektedir. Özellikle donanımsal gelişmelerle paralel olarak, yazılımsal çözümlerin geliştirilmesi büyük bir önem kazanmıştır. Bu nedenle, çeşitli algoritmalar ve yöntemler geliştirilmiş ve uygulanmıştır.[1][2]

Başlangıçta, robotların endüstriyel ortamlarda tekrarlayan görevleri yerine getirmesi, özel yazılımlar ve denetleyicilerle sınırlıydı. Ancak daha sonra, robotların mobil yetenek kazanması, sensörlerin insan duyularına benzer hassasiyetle çalışması ve bilgi iletişim özelliklerinin gelişmesi sayesinde, robotlar tasarlanan ortamla ve insanlarla etkileşimde bulunabilme kapasitesine sahip oldu. Bu durum, sistemlerin giderek karmaşık hale gelmesine yol açtı.[3]

Karmaşıklaşan sistemlerin, ortam algılama, görev tanımlama, hareket etme, enerji sağlama gibi işleri aynı anda gerçekleştirebilme yeteneğinin beklenen bir norm haline gelmesiyle birlikte, birçok algoritma bu amaçla geliştirildi.[4][5]

Robot İşletim Sistemi (ROS), araştırmacılar ve geliştiricilerin robotik uygulamalarda kullanabileceği açık kaynaklı, programlama dili bağımsız bir meta işletim sistemidir. Bu sistem, robot ve robot bileşenlerinin kontrolünü sağlar ve robotun sensörler aracılığıyla dış dünyadan aldığı verileri, örneğin ses veya görüntü, tekrar robota komut olarak iletmek için kullanılır.

ROS, birbirleriyle iletişim kurabilen ve işlem yapabilen birimlerden oluşur. İletişim kurmak için bire-çok abone modeli ve TCP/IP protokolü kullanarak bağımsız modüllere "düğüm" (node) adı verilir. ROS, yayınlama/abone olma mantığına dayanır ve bilgisayar ile robot arasındaki iletişimi, topic'ler, service'ler ve action'lar üzerinden gerçekleştirir.[6]

Gazebo, üç boyutlu dinamik gerçek dünya simülasyon yazılımı olarak tasarlanmış ve açık veya kapalı mekanlar için kullanılabilen bir platformdur. 2002 yılında Linux ortamında açık kaynak olarak başlatılan Gazebo, 2009 yılında ROS ve PR2 ile entegre edilmiştir. Fizik, sensörler, sahneleme gibi işlevleri gelişmiş ve net bir şekilde sunar.

Gazebo, temel olarak istemci-sunucu ilişkisiyle çalışır. Gazebo sunucusu fiziksel işlemleri gerçekleştirirken, Gazebo istemcisi kullanıcının etkileşimini ve simülasyonun görselleştirilmesini sağlar. Bu sayede Gazebo ortamında robotlar, gerçek dünyadaki gibi hızlı bir şekilde test edilebilir ve sanal olarak çevreleriyle etkileşime girebilir..[7]

TurtleBot, düşük maliyetli ve açık kaynaklı bir yazılıma sahip kişisel bir robot kiti olarak bilinir. Öte yandan, TurtleBot3, ROS tabanlı bir mobil robottur ve eğitim, araştırma, hobi ve ürün prototipleme gibi birçok alanda kullanılmaktadır. Özellikle SLAM ve Navigasyon uygulamalarında sıkça tercih edilen bir robot kiti olarak öne çıkar. TurtleBot3'ün Gazebo üzerinde hazır bir modeli bulunmaktadır, bu da robotun hareketlerini simülasyon üzerinden gerçekleştirmek için kullanılabilmesine olanak tanır.[8]

3
Fizibilite

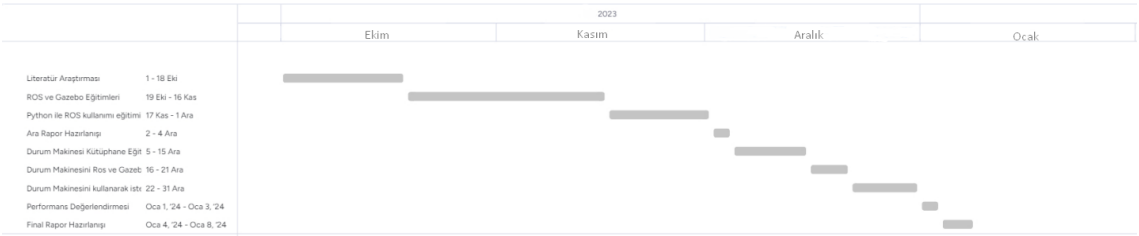
3.1 Teknik Fizibilite

1. Proje ekibi, ROS ve Gazebo simülasyon ortamlarını etkili bir şekilde kullanabilmek için gerekli teknik bilgiye sahiptir.

2. Seçilen durum makineleri kütüphanesi, projenin gereksinimlerini karşılamak için uygun ve mevcuttur.

3.2 İş Gücü ve Zaman Planlaması

3.2.1 Gant Diyagramı



Şekil 3.1 Gant Diyagramı

3.3 Yasal Fizibilite

1. Proje, tüm yasal düzenlemelere uygun olarak yürütülecektir.

2. ROS ve Gazebo kullanımı ile ilgili lisans koşulları ve kullanım şartları dikkate alınacaktır.

3. Projenin fikri mülkiyet hakları ve kullanılan yazılımların lisanslama detayları göz önünde bulundurularak yasal koruma sağlanacaktır.

3.4 Ekonomik Fizibilite

Yazılım gelistirmelerinde kullanılan dil ve kütüphaneler ücretsizdir.

Tablo 3.1 Maliyet Tablosu

Açıklama	**Fiyat**
ROS ve Gazebo'nun bilgisayarlara kurulumu	0
Kullanılan Kütüphaneler	0

4
Sistem Analizi

4.1 Hedef

Projenin temel hedefi, ROS ve Gazebo kullanarak durum makinelerini entegre ederek robotların karmaşık görevleri daha etkili bir şekilde yerine getirmektir. Bu hedef, robot programlama süreçlerini modülerleştirmeyi ve kontrolü daha anlaşılır ve esnek hale getirmeyi amaçlamaktadır. Proje sonuçları, robot programlamasındaki durum makinelerinin avantajlarını vurgulamak ve gelecekteki projeler için bir referans noktası oluşturmak amacıyla kullanılacaktır.

4.2 Bilgi Kaynakları

ROS ve Gazebo'nun resmi dokümantasyonları, projede başvurulacak temel bilgi kaynakları arasında yer alır. Bu dokümantasyonlar, sistemlerin nasıl kurulacağı, yapılandırılacağı ve kullanılacağı konusunda detaylı bilgiler sunar. Seçilen durum makineleri kütüphanesinin resmi belgeleri, durum makineleri modelleme ve entegrasyon süreçleriyle ilgili önemli bilgiler içerir. Ayrıca, kullanılacak robot modelinin URDF dosyası ve Gazebo konfigürasyonlarına dair dokümantasyon, projenin başarıyla ilerlemesi için kritik bir öneme sahiptir.

4.3 Gereksinim

4.3.1 Teknik Gereksinimler

1. ROS ve Gazebo'nun en son sürümleri.
2. Proje ekibi üyelerinin ROS ve Gazebo kullanımı konusunda yeterli bilgi ve beceri.
3. Durum makinelerini entegre etmek için uygun bir ROS paketi.
4. Geliştirilecek durum makinelerini test etmek için uygun bir robot modeli.

4.3.2 Donanım Gereksinimleri

1. Gazebo simülasyon ortamını çalıştırmak için yeterli grafik işleme gücüne sahip bir bilgisayar.

4.3.3 Eğitim Gereksinimler

1. Proje ekibi üyelerinin ROS ve Gazebo eğitimleri alması.
2. Durum makineleri kütüphanesi kullanımı ile ilgili eğitimler.

4.4 Sorular

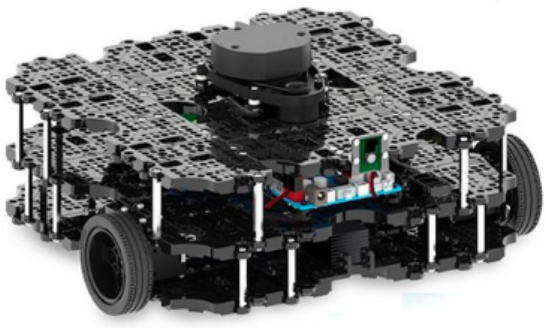
1. Durum makinelerini kullanarak gerçekleştirilecek görevlerde performans nasıl ölçülecek?
2. Geliştirilen sistem, geleneksel programlama yöntemleri ile karşılaştırıldığında hangi avantajlara sahip olacak?
3. Durum makineleri kullanımının sistem üzerindeki etkileri nasıl değerlendirilecek?
4. Robot modeli üzerinde durum makineleriyle gerçekleştirilen simülasyonlar ne kadar gerçekçi sonuçlar verecek?
5. Projede ortaya çıkan hatalar ve zorluklar nasıl ele alınacak ve çözümlenecek?

5
Sistem Tasarımı

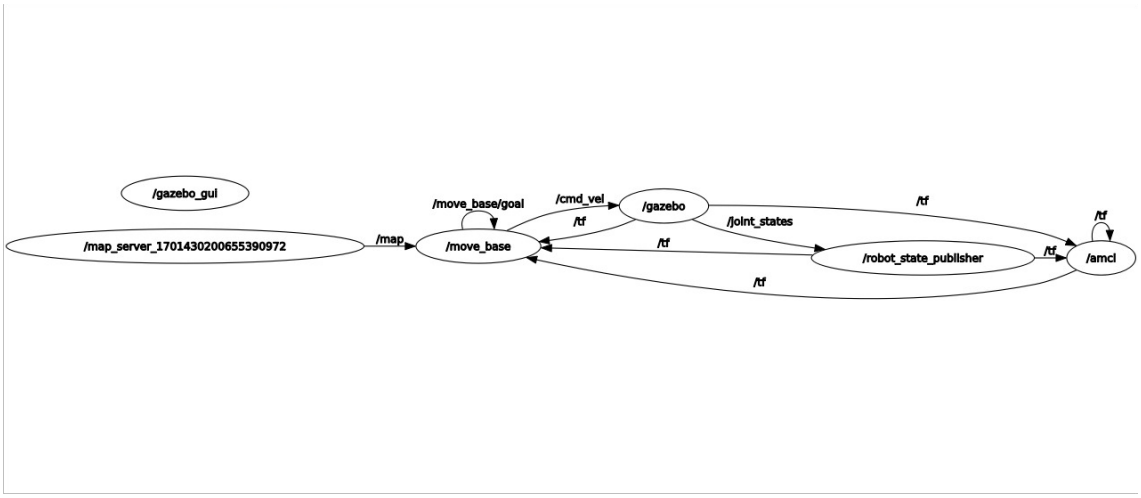
Bu projede geliştirilen sistem, etkileşimli bir şekilde çalışabilen modüler bir yapı üzerine inşa edilmiştir, ROS ve Gazebo'nun birlikte entegre çalışmasını sağlamaktadır. ROS, bu projede ana kontrol ve iletişim aracı olarak kullanılmıştır. Projedeki ROS çalışma alanı, kullanılacak ROS paketlerini ve modülleri içermektedir. Bu paketler arasındaki etkileşimi sağlamak için ROS konuları ve hizmetleri kullanılarak bir iletişim ağı oluşturulmuştur.

Proje, sensör verilerinin ve kontrol komutlarının sistem içinde nasıl dolaştığını belirleyen bir veri akışı ve kontrol mekanizması içermektedir. ROS konuları, simülasyon ortamında sensör verilerini ve kontrol komutlarını paylaşmak için kullanılmıştır.

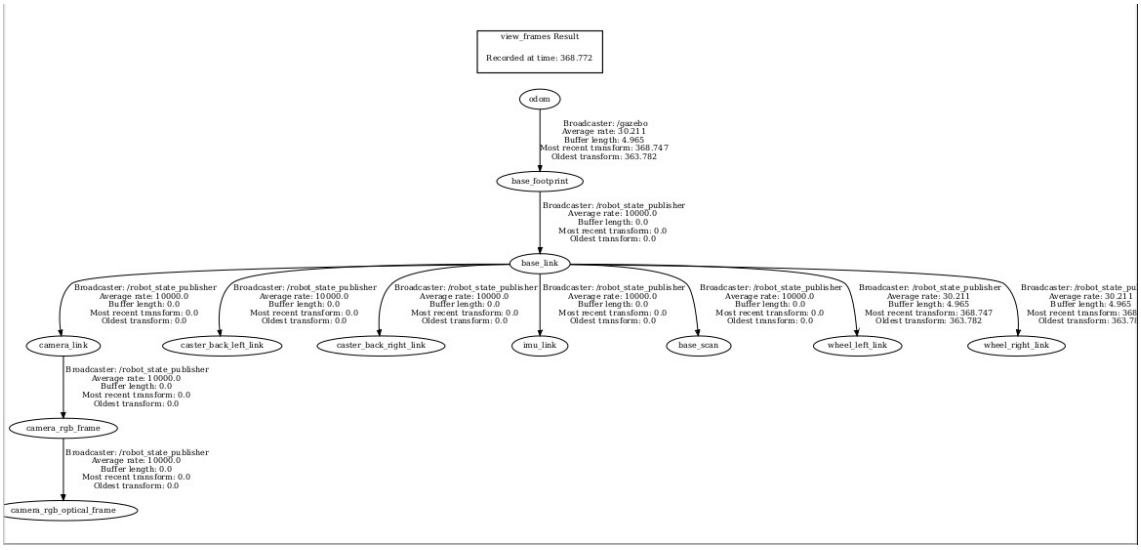
5.1 Diagramlar



Şekil 5.1 Simülasyon ortamında kullanılacak robot ve modeli



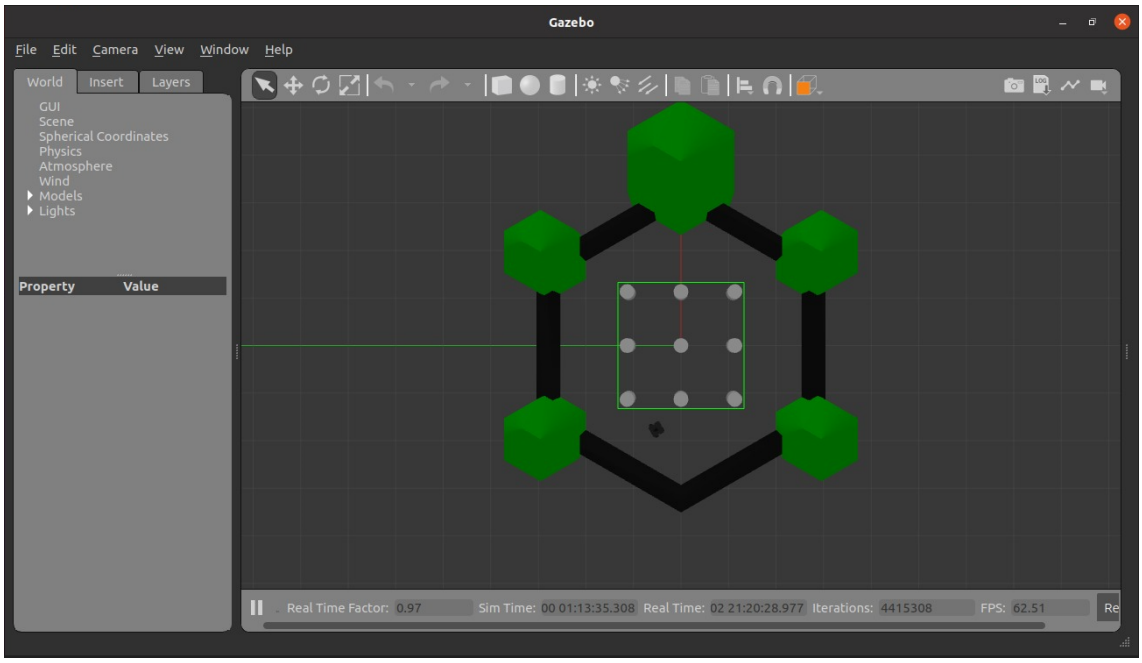
Şekil 5.2 Hareket eden bir robotun rqt-graph diyagrami



Şekil 5.3 Odom parametresi üzerinden yapılan yayınları gösteren TF2 tree

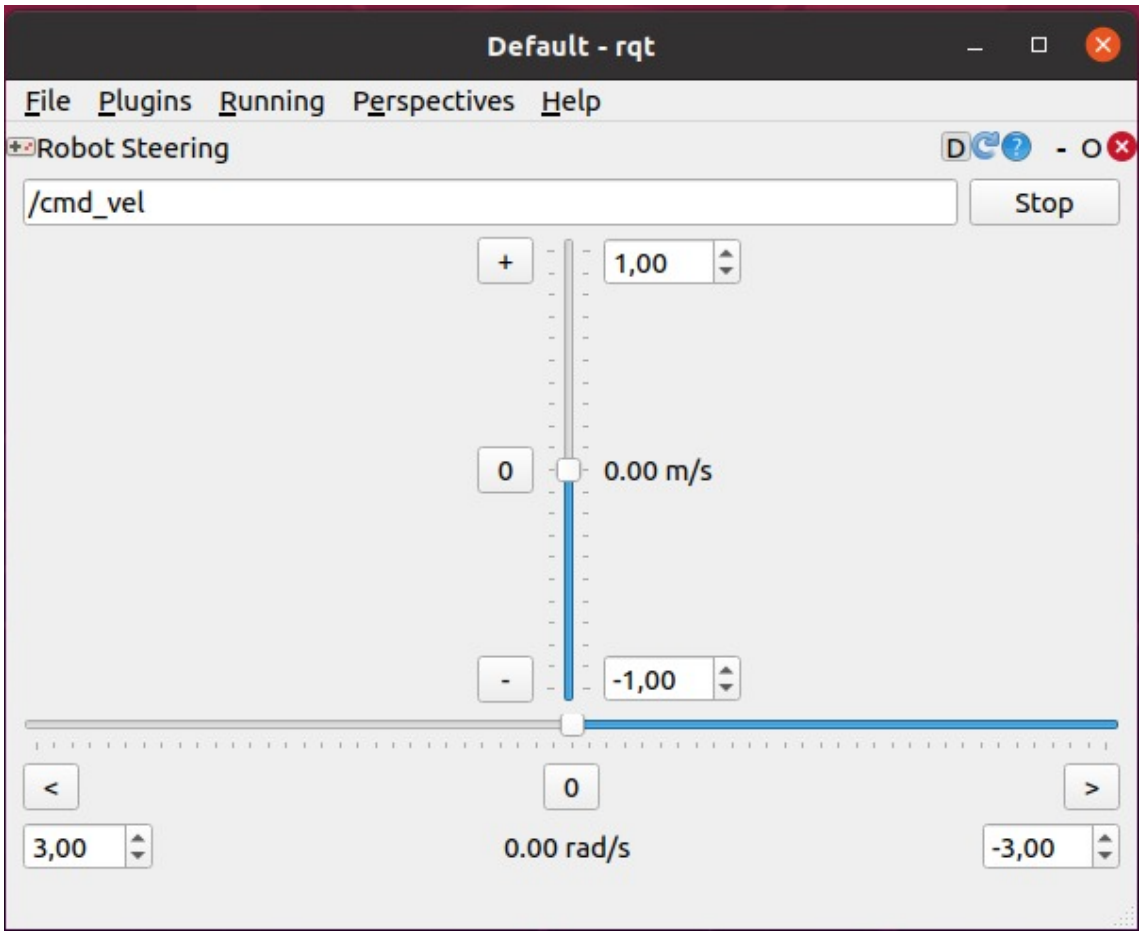
6
Uygulama

Tasarımın tamamlanmasının ardından, kullanılacak robot simülasyon ortamına aktarıldı ve örnek bir harita üzerinde çalışmalara başlandı. Uygulamalar, ROS çekirdeğinin, ortamın ve robotu hareket ettiren kumandanın bir arada çalıştırılması ile mümkün hale geldi. Ancak, launch dosyaları bu adımları bizim adımıza aynı anda gerçekleştirmektedir. Robotumuz bir yere konumlandırıldığında, gidebileceği yerleri

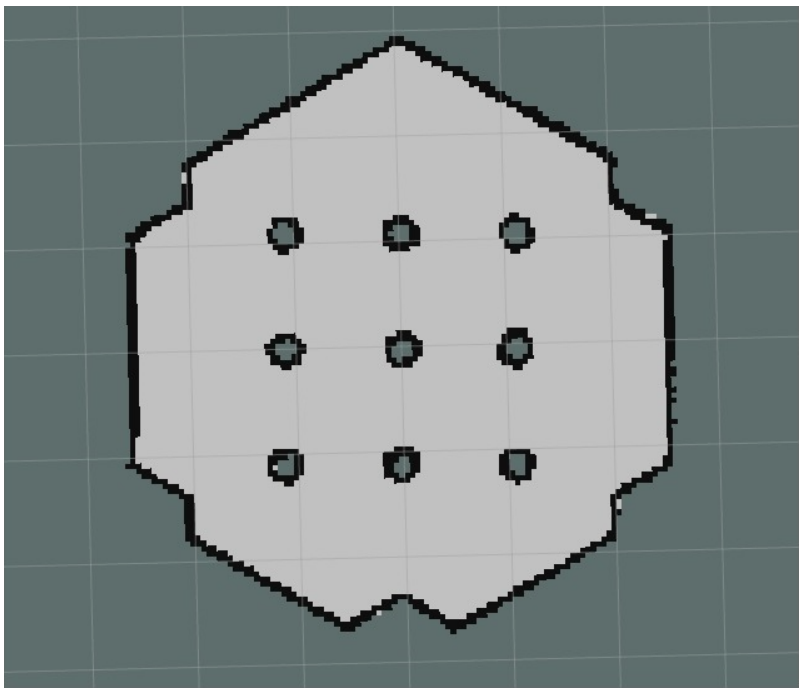


Şekil 6.1 Robotumuzun, simülasyon ortamında bir harita üzerinde görünüşü

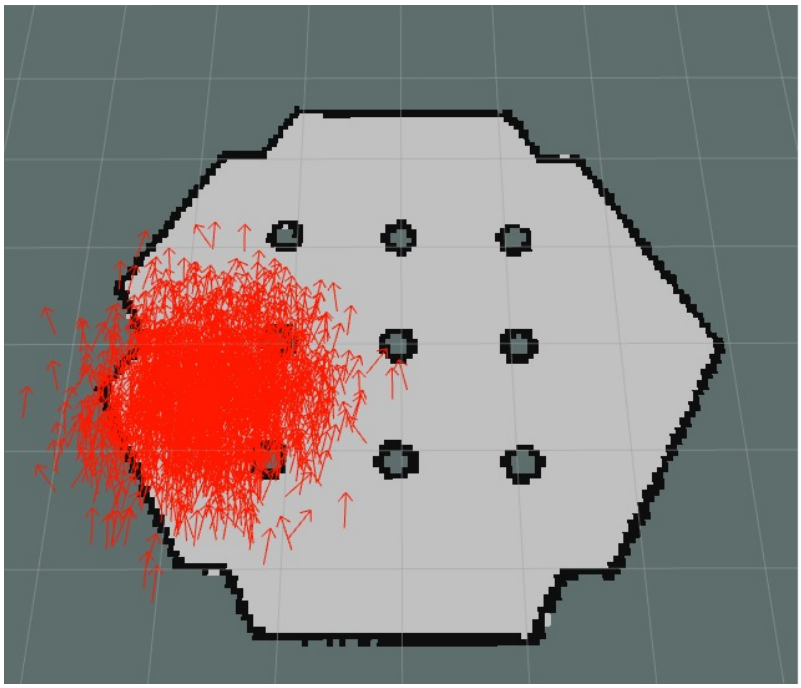
olasılıksal olarak hesaplamakta, harekete başladığında bu tahminler iyileşmekte ve doğru hale gelmektedir.



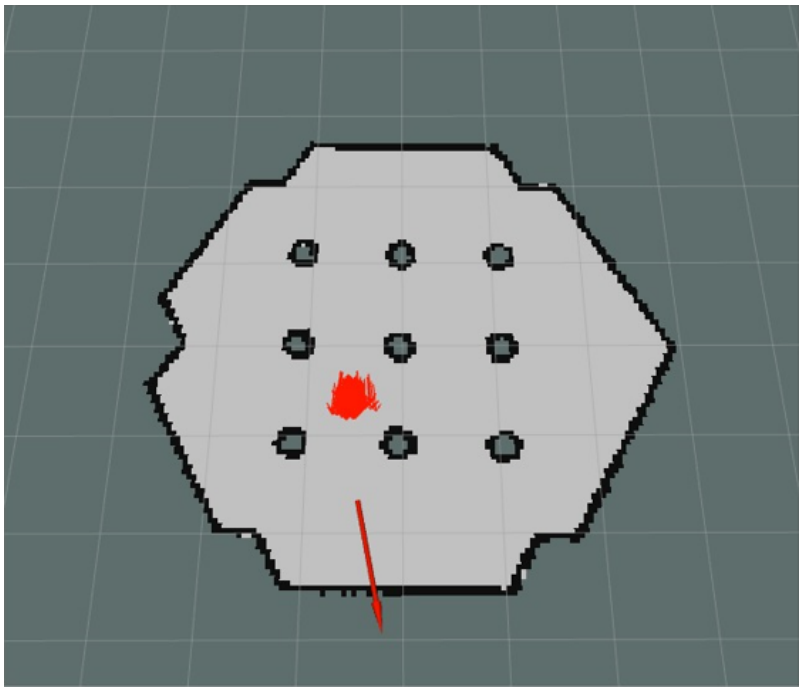
Şekil 6.2 Robotun açısal ve çizgisel hızını ayarlayabileceğimiz kumandamız



Şekil 6.3 Haritamızın robotun tahminlerini görüp, hedef vermeyi sağladığımız Rviz ortamında görüntüsü



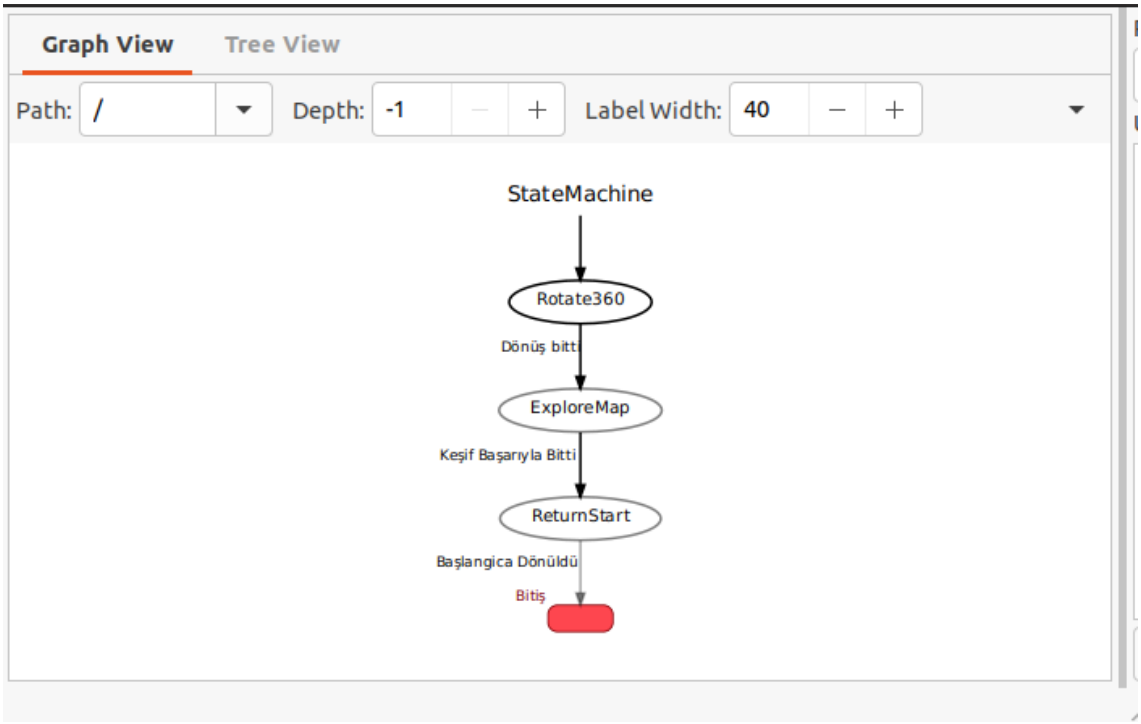
Şekil 6.4 Robotumuzun ilk konumlandırılmasındaki tahminleri



Şekil 6.5 Robotun hareket ettikçe iyileşen tahminleri ve hedefin veriliş şekli

7
Deneysel Sonuçlar

Görev senaryosu olarak belirlediğimiz 3 görev şu şekilde: Etrafında Dön, Haritayı Keşfet, Başlangıca Geri Dön.



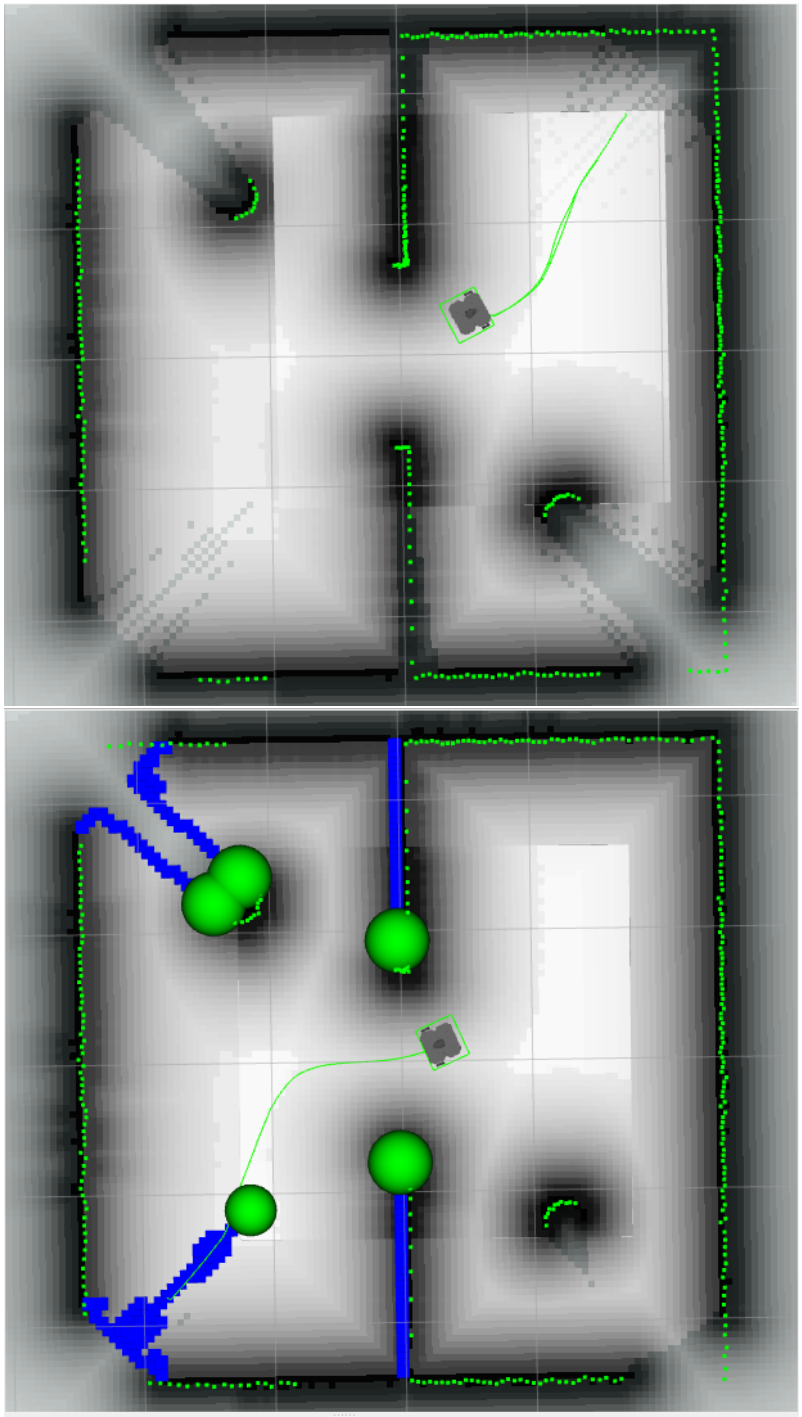
Şekil 7.1 Durum makinesi üzerinde durumların görünüşü

Henüz daha etrafında dönme üzerinde çalışırken robot hareketlerini aynı komutlar altında bazen gerçekleştirirken bazen de gerçekleştirmedi. Bu durum projenin daha ilk aşamasında olmamızdan dolayı ilerlemeyi güç hale getirdi.

Bir sonraki aşamaya geçtiğimizde keşif için gerekli launch dosyası çalışırken durum makinemiz bunun farkında olamadı ve program çalışmasını sürdürdü, fakat bu sorunu gerekli yayıncılardan ilgili mesajların dinlenmesiyle hallettik.

Her ne kadar gerekli mesajları dinlesek de robotumuz bazen başlangıca geri dönme durumuna henüz keşif tamamlanmadan dönüyordu, bazen de dönüyor fakat daha

sonra keşfe devam ediyor ve programı keşfi bitirdiği yerde durduruyordu. Bunu da çözmek adına keşfi bitirdiğinden emin olmadan dönmemesi için bitirdikten sonra gerekli dinlemeleri yapmasını sağladık ve robota bir miktar süre tanıdık.



Şekil 7.2 Keşif sırasında robot

Tüm bunları yaptığımızda robot başarıyla konduğu ortamda, keşfini tamamlıyor ve daha sonra başladığı noktaya geri dönüyor.

8
Performans Analizi

Robotu 3 farklı harita üzerinde ve çeşitli yerlerde başlattık bu durum keşif süresinin büyük haritada artmasıyla, küçük haritada ise azalmasıyla sonuçlandı. Bu süre farklarını hataları tespit ederken ve simülasyonları sonlandırırken kullandık.

- Eğer aradığımız şey, simülasyon sırasında bir mesajın hatasını bulmak veya bir durumu analizi etmek ise BÜYÜK HARİTA
- Eğer aradığımız şey, simülasyon sona erdiğinde karşılaşacağımız bir durum ve çıktıysa KÜÇÜK HARİTA

Bunun dışında durum makinesinin başındaki durumları kontrol ederken, ortasındaki durumu ve sondaki durumu kontrol ederken harcadığımız süre birbirlerinden farklılık gösterdi, hatta birleştirmeye çalışırken bu durum bir hayli arttı çünkü eğer son durumumuzda bir hata bulunduysa, ilk iki durumun başarıyla tamamlandığını her defasında tekrar tekrar görmek zorunda kaldık. Tablo 8.1, 8.2 ve 8.3 de küçük,orta büyük haritada gerçekleşen Etrafında Dönme, Keşif ve Başlangıca Dönme durumlarında harcanan süreler verilmiştir.

Tablo 8.1 Etrafında Dönme

Harita Boyutu	**Harcanan Süre(sn)**
Küçük Harita	12,56
Orta Harita	12,56
Büyük Harita	12,56

Tablo 8.2 Keşif

Harita Boyutu	**Harcanan Süre(sn)**
Küçük Harita	37,66
Orta Harita	33,74
Büyük Harita	227,33

Tablo 8.3 Başlangıca Dönme

Harita Boyutu	**Harcanan Süre(sn)**
Küçük Harita	22.46
Orta Harita	36.40
Büyük Harita	39.84

Tablolarda da görüldüğü gibi etrafında dönme sürelerinde bir değişiklik gözlenmez iken kalan tüm değerlerde değişiklik göstermiştir ve bu değişiklik başlangıca dönmede, bitiş noktasının başlangıç noktasına uzaklığına bağlı bir orandayken; keşifte ise, haritanın büyüklüğüyle orantılıdır. Ancak istisnai bir şekilde haritanın köşeli bir yapıda olmaması, ve rahat keşfedilebilir olmasından dolayı küçük haritadan daha kısa sürede keşfedilmiştir.

9
Sonuç

Bu projede, Turtlebot3 robotu üzerinde durum makineleri kullanılarak modüler bir görev tamamlama simülasyonu geliştirildi. Python programlama dili ve ROS (Robot Operating System) kullanılarak yazılan yazılım, Turtlebot3 üzerindeki sensörlerin kontrolünü sağlamak, robotu etrafında döndürmek, harita keşfi gerçekleştirmek ve başlangıç noktasına geri dönmek gibi temel görevleri başarıyla yerine getirdi. Bu görevlerin gerçekleştirilmesinde ROS'taki publisher ve subscriber yapısı sıkça kullanıldı.

Projenin başarılı bir şekilde tamamlanması, durum makineleri ve ROS platformunun robotik uygulamalardaki güçlü potansiyelini vurgulamaktadır. Elde edilen sonuçlar, modüler ve esnek robot kontrol sistemlerinin geliştirilmesi açısından ilgi çekicidir ve gelecekteki benzer projeler için bir temel oluşturabilir.

Referanslar

[1] G. G. ve İ. TÜRKOĞLU, "Robot sistemlerinde kullanılan algoritmalar," no. 1, pp. 17–31, 2019.

[2] F. B. E. İstanbul: İstanbul Teknik Üniversitesi. "Ortam tarama için robotlarla duyarga ağı konumlandırma." (2009).

[3] L. S and M. H, "Receding horizon particle swarm optimisation-based formation control with collision avoidance for non-holonomic mobile robots.," no. 9, pp. 2075–2083, 2015.

[4] D. Q. Wang T and P. P., "A multi-robot system based on a hybrid communication approach.," no. 1, pp. 91–100, 2013.

[5] F. B. E. İstanbul: İstanbul Teknik Üniversitesi. "14 serberstlik dereceli iki ayaklı bir robotun dinamik yürüme hareketinin kontrolü." (2008).

[6] "Ros website." (), [Online]. Available: `https://www.ros.org`.

[7] "Robotis website." (), [Online]. Available: `https://emanual.robotis.com/docs/en/platform/turtlebot3/ove%20rview/`.

[8] "Turtlebot website." (), [Online]. Available: `https://www.turtlebot.com/`.

Özgeçmiş

BİRİNCİ ÜYE

İsim-Soyisim: Muhammed Kasap
Doğum Tarihi ve Yeri: 12.07.2002, Ordu
E-mail: muhammed.kasap@std.yildiz.edu.tr
Telefon: 0507 155 00 96
Staj Tecrübeleri:

Proje Sistem Bilgileri

Sistem ve Yazılım: Ubuntu İşletim Sistemi, Python3
Gerekli RAM: 2GB
Gerekli Disk: 2GB