Uygulamalı Mekatronik Mühendisliği ve Akıllı Sistemler Dergisi 2021, 1: XX-XX
DOI:10.7240/xxxxx

ROS Tabanlı Mobil Otonom Robot (AMR) Geliştirme

***Melih KORKMAZ*[1*] , İ*zzet Fatih* ŞENTÜRK*[2],**

[1]*Bursa Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Akıllı Sistemler Mühendisliği, Bursa, Türkiye*
[2]*Bursa Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Bilgisayar Mühendisliği, Bursa, Türkiye*

ÖZET

ROS (Robot Operating System) oluşturduğu standartta robotları kontrol etmeyi sağlayan, robotu çevresel bileşenlerle haberleştiren publish-subscribe mesajlaşma modeline sahip açık kaynak kodlu bir işletim sahibidir. Gün geçtikçe desteklediği ürün, paket ve proje sayısı artan ROS, simülasyon araçları ile dilden bağımsız ve modüler bir yapıya sahip olduğundan robot dünyasında standart konumunu almaktadır. Bu çalışmada ROS tabanlı AMR (Autonomous Mobile Robot) kullanılarak simülasyon ortamında SLAM (Simultaneous Localization and Mapping) ve navigasyon fonksiyonları gerçekleştirilmiştir. Simülasyon araçları olarak ROS'un sağlamış olduğu Gazebo, Rviz ve rqt kullanılmıştır. Lidar sensörü ve SLAM fonksiyonu sayesinde konumlama ve haritalama işlemi yapılmıştır. Çıkartılan harita ile çalıştırılan navigasyon fonksiyonu sayesinde AMR olduğu konumdan verilen hedefe dinamik, güvenli ve maliyeti düşük yol planlaması yaparak hareket eder. Ayrıca geliştirilecek arayüz ile AMR'ler manuel olarak kullanılabileceği gibi konumlar istasyon şeklinde kaydedilerek, reçete oluşturulup hızlı bir şekilde AMR'ler süreçlere eklenebilir. Doğal navigasyon, düşük maliyet, hızlı kurulum, merkezi yönetim sistemi ile çoklu robot kontrol imkanları dolayısıyla AMR'ler tercih edilir.

Anahtar Kelimeler: ROS, AMR, SLAM, Navigasyon

I. GİRİŞ

Mobil robotların kullanımı, başta endüstriyel alanlar olmak üzere hizmet, sağlık, savunma vb. sektörlerde giderek daha da yaygınlaşmaktadır [1] [2] [3]. Son zamanlarda yapay zekanın gelişmesiyle mobil robotların otonom şekilde hareket edebilmesini sağlayan sistemler üzerine odaklanılmıştır [4]. Mobil otonom robotlar (Autonomous Mobile Robot - AMR) ortamda hiçbir belirteç olmadan konumunu ve engelleri algılayıp dinamik olarak hedefine gitmesini sağlayan doğal navigasyonu sayesinde dinamik, güvenli ve maliyeti düşük yol planlamasına ve otonom sürüşe sahiptir. Bu çalışmada önce sistemde kullanılan araçlar olarak ROS, Robot ve Lidar sensöründen bahsedilmiştir. SLAM ve navigasyon fonksiyonlarından bahsedilerek simülasyon ortamında testler yapılmıştır. Gazeboda tasarlanan dünya ile birlikte simüle edilen robot kullanılarak harita çıkartılmıştır. Çıkartılan harita ile navigasyon fonksiyonu çalıştırılmış olup robotun verilen hedefe çevresindeki sabit ve değişken engelleri de dikkate alarak yol planlaması yapıp hedefe dinamik olarak vardığı gözlemlenmiştir. Daha sonra ROS ve AMR'nin geleceğinden bahsedilmiştir.

II. ARAÇLAR

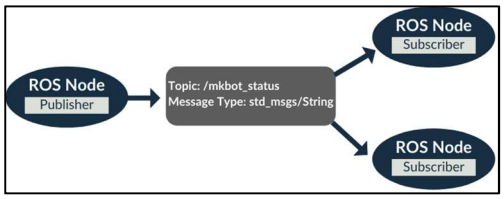
II.1. ROS

ROS robotik platformlar için BSD (Berkeley Yazılım Dağıtımı) lisanslı ve açık kaynaklı bir yazılımdır. Ticari ve araştırma amaçlı kullanım için ücretsizdir. ROS ilk olarak 2007 yılında Stanford Yapay Zeka Laboratuvarı (SAIL) tarafından Stanford AI Robot projesinin desteğiyle geliştirildi. 2008'den itibaren 20'den fazla kurumun işbirliği yaptığı bir robotik araştırma enstitüsü olan Willow Garage'da geliştirmesi devam etmektedir. Birçok araştırma kurumu ve firma donanım ekleyerek ve kod örneklerini paylaşarak ROS'ta projeler geliştirmeye başladılar. Bu çerçevede her geçen gün içerdiği proje sayısı ve desteklediği cihaz sayısı artmaktadır [5].

ROS Ubuntu Linux işletim sisteminde çalışmakta ve dil olarak Python, C++ ve Java desteklemektedir. ROS'un Melodic ve Noetic gibi güncel sürümleri mevcuttur [6].

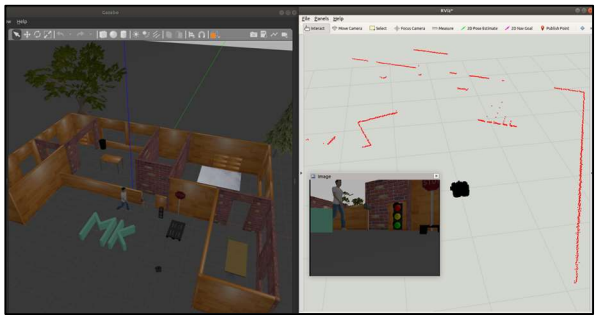
Şekil 1'de gösterildiği gibi gibi ROS sistem olarak publish-subscribe (yayınlama-abone) mesajlaşma modelini kullanarak birbirleri ile iletişim kuran düğümlerin (node) meydana getirdiği bir mimaridir. Modüler bir yapıda olduğundan paketlerin ve hataların yönetilmesinde kolaylık sağlar.

***Sorumlu yazar/Corresponding Author:** Melih KORKMAZ, Tel: +90 XXX XXX XX XX, e-posta: mk.melihkorkmaz@gmail.com
Gönderilme/Submitted: 12/05/2021, **Revised:** 12/07/2021, **Kabul/Accepted:** XXXXXXX



Şekil 1: ROS Mesajlaşma Modeli

ROS'un görselleştirme ve yönetim için sağlamış olduğu araçlardan ilki Gazebo, robotun ve çevresel objelerin koyulabildiği sanal dünya sağlamaktadır. Rviz ise robotun giriş ve çıkışlarının (robot modeli, robot konumu, sensör verileri, harita, sabit ve dinamik engeller, rota) simüle edildiği platformdur. Yönetim aracı olan Rqt ise parametrelerin değiştirilmesi, mesajların okunması ve yayınlanması, verilerin grafikte çizdirilmesi ve kaydedilmesini sağlar [7].

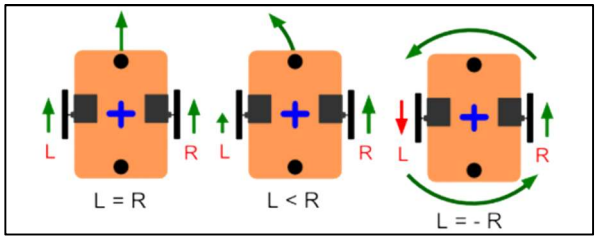


Şekil 2: Gazebo ve Rviz

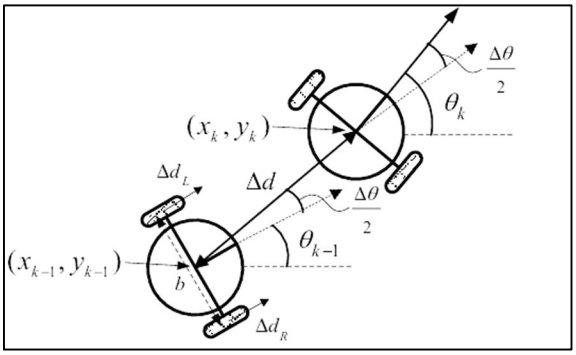
Şekil 2'de soldaki görselde Gazebo'da oluşturulan dünya ve robot gösterilmiştir. Sağdaki görselde ise Rviz'deki robotun çıktıları olarak robot model, kamera ve sensör verileri gösterilmiştir.

II.2. Robot

Kobuki firması tarafından üretilmiş olan Turtlebot mobil robotu kullanılarak simülasyon ortamında testler yapılmıştır. Sürüş yöntemi olarak kullanılan diferansiyel sürüş yönteminde Şekil 3'te gösterildiği gibi tekerleklerin hız farkından robotun sürüşü, yön farkından ise kendi etrafında dönmesi sağlanır [8]. Şekil 4'te gösterildiği gibi odometri verisi zamana bağımlı olarak robotun hareketini karakterize eden veridir.

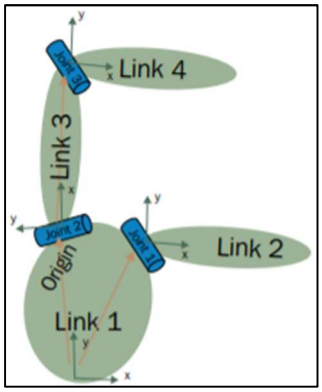


Şekil 3: Diferansiyel Sürüş Yöntemi [9]

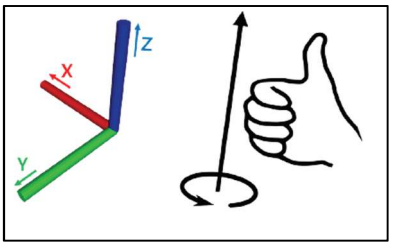


Şekil 4: Odometri Verisi [10]

Robotun tasarımı 3 boyutlu tasarım programı olan Solidworks ile yapıldıktan sonra ROS'un sw_urdf_exporter paketi [11] ile URDF (Unified Robot Description Format) dosyasına çevrilir. Ayrıca robot model için 3 boyutlu dosya formatı (stl) olarak mesh dosyaları alınır. URDF dosyasında robotun tüm kinematiği mevcuttur ve ROS'ta bu transform (tf) olarak belirtilmektedir.



Şekil 5: Joint ve Link Gösterimi (tf) [12]



Şekil 6: ROS tf Eksenleri [12]

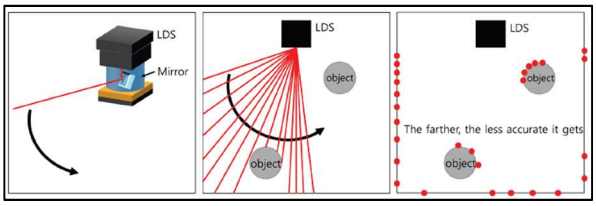
Şekil 5'te gösterildiği gibi URDF dosyasındaki tf değerleri robotun tüm sabit ve hareketli eksenlerini, eksenlerin arasındaki uzaklık ve açı değerlerini, hareketli eksenlerin nasıl hareket edeceğini ve hareketi ile ilgili değerleri, hangi eksenlerin birbirine bağlı hareket edeceği gibi birçok bilgiyi içerir. Tf'lerin x y z eksenleri sağ el kuralına göre Şekil 6'da gösterildiği gibidir. Otonom sürüş esnasında hız değerleri ve sensör

verileri tf değerlerine göre hesaplanacağından bu değerlerin hassas olması önem arz etmektedir [12]. Robota ait bütün tf'ler Şekil 17'de Tf ağacı olarak gösterilmiştir.

II.3. Lidar Sensörü

Robotun çevreyi algılamasında ışık algılama ve ölçme sensörü (LIDAR-Laser Imaging Detection and Ranging) – Laser Distance Sensor (LDS) - kullanılmıştır. Lidar, çevresindeki nesnelerin konumlarını algılamak için kullanılan radar benzeri bir sensördür. Lidar sensörleri yüksek performans ve hızla gerçek zamanlı veri toplama avantajına sahiptir, bu nedenle mesafe ölçümü ile ilgili geniş bir uygulama yelpazesine sahiptir. Lidar sensörü, nesnelerin ve insanların algılanması için robotik alanında, SLAM algoritmalarında ve gerçek zamanlı veri topladığı için insansız araçlarda yaygın olarak kullanılmaktadır.

Lidar sensörünün temelinde lazer, yansıtıcı bir ayna ve motor mevcuttur. Şekil 7'de solda lidar sensörü ve içindeki ayna gösterilmektedir. Motor aynayı döndürür, sensör lazerin dönüş süresini dalga boyundaki fark ile ölçer. Ortadaki şekilde gösterildiği gibi lidar etrafını yatay bir düzlemde tarar. Sağ şekilde de gösterildiği gibi uzaklık arttıkça hassasiyet azalmaktadır. Cam, plastik şişe gibi malzemeler lazer ışığını dağıttığından sağlıklı veri elde edilememektedir. Sadece yatay bir düzlemi taradığından 2D veri olduğu unutulmamalıdır [13].



Şekil 7: Lidar Sensörü ile Mesafe Ölçümü [13]

Şekil 8'de LaserScan mesaj türündeki lidar verisi gösterilmiştir. LaserScan verisi sensörün hangi tf'te olduğu (frame_id), kaç derece tarama yaptığı (angle_min/angle_max), min ve max tarama mesafesi (range_min/range_max), tarama frekansı ve çözünürlüğü ile birlikte her noktanın uzaklığını metre cinsinden içeren diziyi içermektedir.

```
header:
  seq: 263
  stamp:
    secs: 1624124262
    nsecs: 324845830
  frame_id: "laser"
angle_min: -3.1241390705108643
angle_max: 3.1415927410125732
angle_increment: 0.01745329238474369
time_increment: 0.0003783728461712599
scan_time: 0.13583585619926453
range_min: 0.15000000596046448
range_max: 12.0
ranges: [0.29499998688697815, 0.30399999022483826, 0.3109999895095825, 0.
617, 1.503000020980835, 1.4889999628067017, 1.4809999465942383, 1.4620000
06999945640564, 1.402999997138977, 1.3990000486373901, 1.5119999647140503
3131104, 1.4170000553131104, 1.409999966621399, 1.4049999713897705, 1.399
```

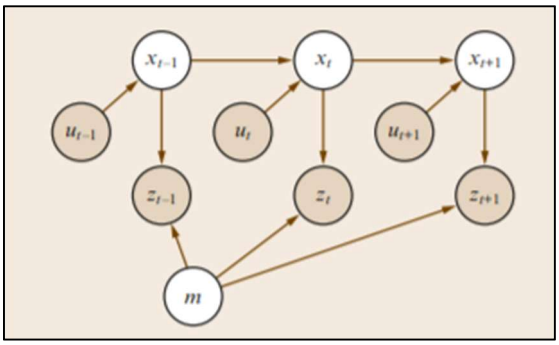
Şekil 8: Lidar (LaserScan) Verisi

III. SLAM

SLAM (Simultaneous Localization and Mapping), robotun üzerindeki sensörler ile robotun pozisyonunu tahmin ederken bilinmeyen bir çevreyi keşfetmek ve haritalamak anlamına gelir. SLAM algoritması olasılık terminolojisinde Denklem 1'deki gibi tanımlanır. Şekil 9'da gösterildiği gibi zamana bağımlı olarak ardışık işlem yaparak harita verisini çıkarır [14].

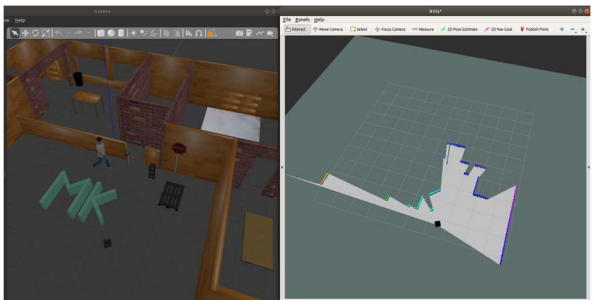
t: $zaman$
x_t: t $anındaki$ $robotun$ $konumu$
u_t: t $anındaki$ $odometri$ $verisi$
z_t: t $anındaki$ $lidar$ $sensöründen$ ölçülen $değer$
m: $harita$

$$\mathrm{p}(x_t, \mathrm{m}|z_t, u_t) \quad (1)$$

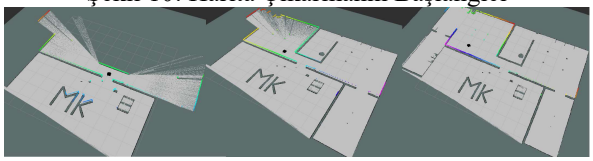


Şekil 9: SLAM Grafiksel Model [14]

Robot Gazebo ortamına eklendikten sonra, Gazebo'nun bize sağlamış olduğu odometri, lidar ve tf verileri ile haritalama işlemini gerçekleştiren gmapping [15] algoritması çalıştırılır. İlk çalıştığında Şekil 10'daki gibi lidar sensörünün ilk gördüğü yerlerin haritasını çıkarır. Robotu ideal bir hızda sürerek Şekil 11'deki gibi adım adım harita genişletilir. Buradaki en önemli ölçüt lidar sensörünün ölçüm mesafesidir.



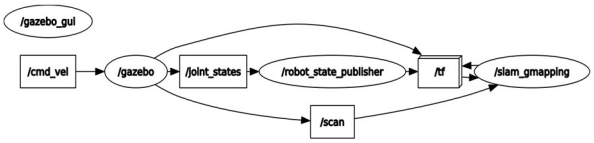
Şekil 10: Harita Çıkarmanın Başlangıcı



Şekil 11: Haritanın Çıkartılmasının Adımları

Daha sonra çıkartılan harita navigasyonda kullanılmak üzere uygun formatta kaydedilir. Şekil 12'de pgm formatındaki haritanın görseli ve haritayı yüklemesi için ROS tarafından kullanılacak olan YAML (YAML Ain't Markup Language) formatı gösterilmiştir. Şekil 13'te ise SLAM fonksiyonu çalıştığında hangi node'ların hangi mesajları dinleyip yayınladığını belirten NodeGraph'i verilmiştir.

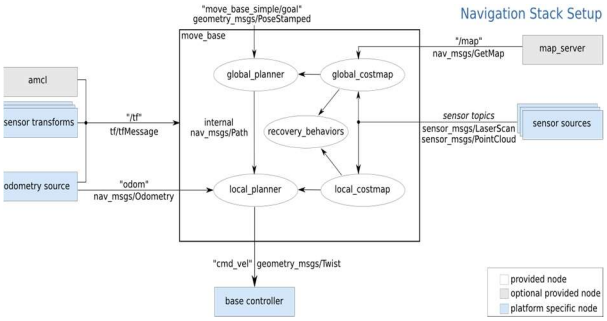
Şekil 12: Harita Görseli ve Parametre Dosyası



Şekil 13: SLAM Fonksiyonu (NodeGraph)

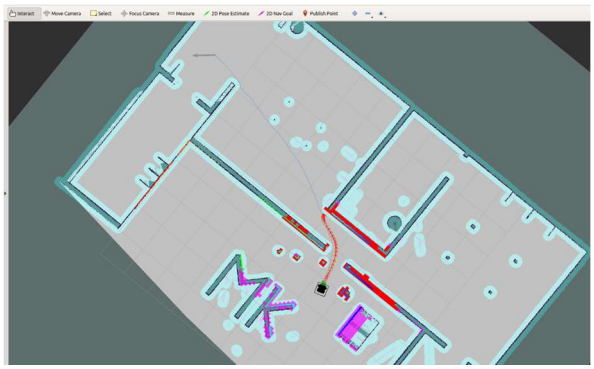
IV. NAVİGASYON

Navigasyon fonksiyonu daha önceden çıkartılan haritanın yüklenmesi ve lidar verisinin kullanılmasıyla konumlama işlemini yaparak otonom bir şekilde hedef noktaya dinamik olarak gitmesidir. Navigasyon fonksiyonu olarak ROS'un Şekil 14'teki grafiksel modele sahip move_base paketi kullanılmıştır. Konumlama işlemini lidar verisi ve haritadaki sabit değerleri (map_server) kullanarak AMCL (Adaptive Monte Carlo Localization) algoritması yapar. Sabit ve hareketli engelleri simüle eden 2 adet maliyet haritası (costmap) mevcuttur. Global costmap global yol planlaması için kullanılır. Global yol planlaması harita üzerindeki hedefe uzun vadeli yol planını yapar. Local costmap ise robotu merkezine alıp onunla hareket eden küçük bir haritadır ve local yol planlaması için kullanılır. Robotun hız değerleri local costmapteki engellere göre ve local yol planlamasındaki pozisyonlara göre çıkar. Engelleri algılamada 2D olarak lidar verisi, 3D olarak nokta yığınlarından oluşan point cloud (pcl) verisi kullanılabilir [16].



Şekil 14: ROS Navigation Grafiksel Model [16]

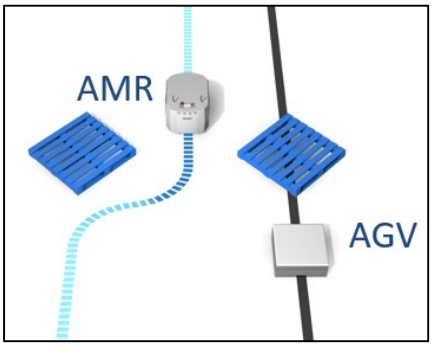
Şekil 15'te navigasyon fonksiyonunun çalıştırılmış hali mevcuttur. Robot gri ok ile gösterilmiş hedefe sabit ve hareketli engelleri dikkate alarak otonom bir şekilde gitmektedir. Haritada pembe ile local costmap, turkuaz ile global costmap, kırmızı ile local path gösterilmiştir. Local path'in takip ettiği global path ise mavi renk ile gösterilmiştir.



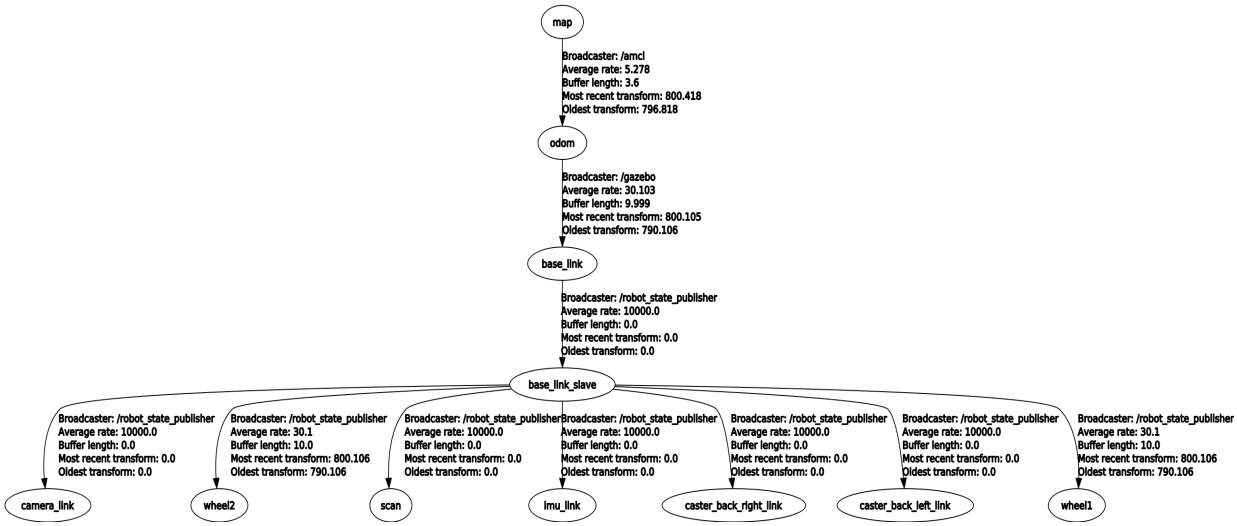
Şekil 15: Robotun Navigasyonda Hedefe Gitmesi

Şekil 17'de AMCL'in yayınladığı map tf'i görülmektedir. Şekil 18'de Navigasyon fonksiyonunun NodeGraph'i verilmiştir.

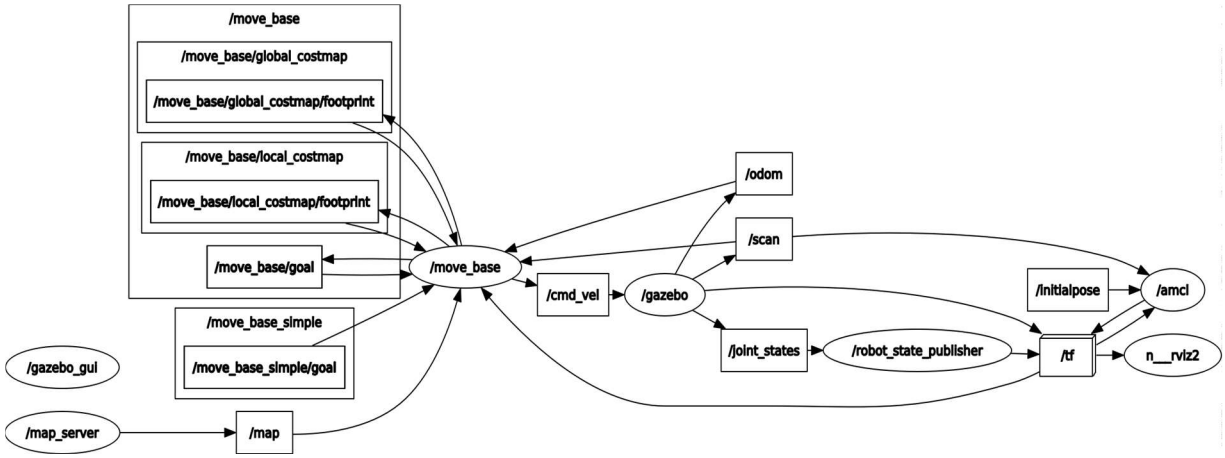
AMR'leri AGV (Automated Guided Vehicle) lerden ayıran en önemli farklar Şekil 16'da gösterildiği gibi ortamda herhangi bir belirteç olmadan otonom sürüşe sahip olması ve engel ile karşılaştığında yolunu dinamik olarak değiştirip hedefe gidebilmesidir.



Şekil 16: AMR ve AGV Farkı [17]

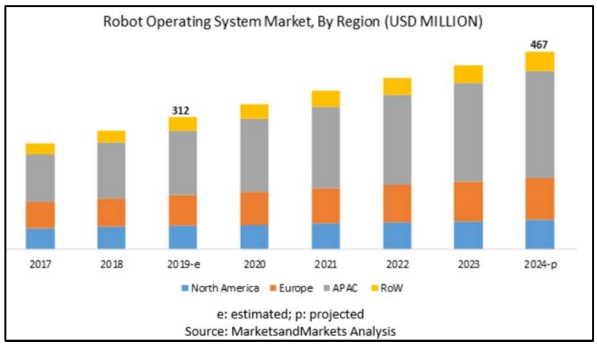


Şekil 17: Tf Tree



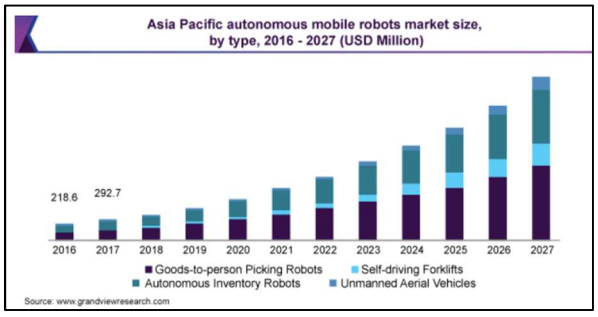
Şekil 18: Navigasyon Fonksiyonu (NodeGraph

V. ROS ve AMR'nin GELECEĞİ



Şekil 19: ROS Market Payı (2017-2024) [18]

2019 Ağustos'unda yayınlanan Market Araştırma Raporu'na göre Şekil 19'da gösterildiği gibi 2019'da 312 milyon dolar olan ROS pazarının 2024'te 467 milyon dolara yükseleceği öngörülmüştür. ROS pazarının büyümesini sağlayan etkenlere örnek olarak endüstri otomasyonu için artan Ar-Ge fonları, işbirliğine dayalı modüler robotların uygulamalarının artması, hizmet olarak robotik modeline artan talep ve düşük maliyetli endüstriyel robotların tercih edilmesi verilebilir. En yüksek büyüme oranına sahip bölgenin Çin, Japonya, Güney Kore'den dolayı Asya Pasifik (APAC) olacağı tahmin edilmektedir [18].



Şekil 20: AMR Market Payı (2016-2027) [19]

2021 Ocak ayında yayınlanan Market Analiz Raporu'na göre Şekil 20'de gösterildiği gibi Asya Pasifikte AMR pazarı 2019'da 1.9 milyar dolar ve 2020'de 2.36 milyar dolar olup, 2027' de ise 8.29 milyar dolara ulaşacağı tahmin edilmektedir. AMR'ler, tesis içinde hammadde ve mamul malların taşınması gibi çeşitli işlemleri gerçekleştirmek için depo yönetim yazılımları ile entegre edilmiş kamera, sensör ve tesis haritalarından yararlanır. AMR'ler ürün hasarını önleme, işçilik maliyetlerini düşürme, üretkenliği arttırma ve süreçleri otomatikleştirme gibi sayısız faydası nedeniyle oldukça tercih edilmektedir [19]**.**

VI. SONUÇLAR

ROS platformu hızla gelişmekte olup ve desteklediği marka ve proje sayısı giderek artmaktadır. ROS'ta robot ve dünyanın simüle edildiği Gazebo, çıktılarının simüle edildiği Rviz ortamı kullanılır. SLAM algoritmaları sayesinde istenilen her ortamda hızlıca harita çıkartılabilir. Ek olarak çıkartılan harita robota bağımlı olmadığından istenilen her robotta kullanılabilme esnekliğini sağlar.

Çıkartılan harita kullanılarak navigasyon fonksiyonu çalıştırıldığında doğal navigasyon sayesinde ortamda hiçbir belirteç olmadan robot konumlama işlemini yapabilmektedir. Hareketli engelleri de dikkate alarak dinamik, maliyeti en düşük yol planlamasını çıkartıp hedefe ulaşması AMR'leri ön plana çıkarmaktadır.

Robotun fiziksel özelliklerine, kullanılan sensörlerine, bulunduğu ortama göre yazılımda ve ilgili parametrelerde değişiklik yapılarak daha ideal çalışma durumu sağlanabilir. Geliştirilebilecek arayüzlü merkezi yönetim sistemi ile birden fazla robot görev mantığında devreye alınabilir.

VII. REFERANSLAR

[1] F. C. Christian Tamantini, Francesco Scotto di Luzio and and L. Z. Giuseppe Pascarella, Felice Eugenio Agrò, "A Robotic Health-Care Assistant for the COVID-19 Emergency," *IEEE Robot. Autom. Mag.*, no. March, pp. 71–81, 2021.

[2] G. Fragapane, D. Ivanov, M. Peron, F. Sgarbossa, and J. O. Strandhagen, "Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics," *Ann. Oper. Res.*, 2020, doi: 10.1007/s10479-020-03526-7.

[3] J. Bačík *et al.*, "Phollower—the universal autonomous mobile robot for industry and civil environments with COVID-19 germicide addon meeting safety requirements," *Appl. Sci.*, vol. 10, no. 21, pp. 1–16, 2020, doi: 10.3390/app10217682.

[4] G. Fragapane, R. de Koster, F. Sgarbossa, and J. O. Strandhagen, "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," *Eur. J. Oper. Res.*, no. xxxx, 2021, doi: 10.1016/j.ejor.2021.01.019.

[5] A. Martinez and E. Fernández, *Learning ROS for Robotics Programming*. 2013.

[6] "Documentation - ROS Wiki." http://wiki.ros.org/ (accessed Apr. 18, 2021).

[7] R. Baxter, N. Hastings, A. Law, and E. J. . Glass, *Book-Effective Robotics Programming with ROS*, vol. 39, no. 5. 2008.

[8] C. Fairchild and T. L. Harman, *ROS Robotics By ExamFairchild, C., & Harman, T. L. (2016). ROS Robotics By Example.ple*. 2016.

[9] M. Granát, "Mobile robot control and guidance using computer vision," *BRNO Univ. Technol.*, 2018.

[10] K. Lee, C. Jung, and W. Chung, "Accurate calibration of kinematic parameters for two wheel differential mobile robots," *J. Mech. Sci. Technol.*, vol. 25, no. 6, pp. 1603–1611, 2011, doi: 10.1007/s12206-011-0334-y.

[11] "sw_urdf_exporter - ROS Wiki." http://wiki.ros.org/sw_urdf_exporter (accessed Jun. 19, 2021).

[12] L. Joseph, *Mastering ROS for Robotics Programming*, vol. 64, no. 6. 2015.

[13] Y. Pyo, H. Cho, J. Ryuwoon, and T. Lim, *Robot Programming From The Basic Concept To Practical Programming and Robot Application*. 2017.

[14] K. Bruno, Siciliano Oussama, *Handbook Springer of Robotics*. 2008.

[15] "OpenSLAM.org." https://openslam-org.github.io/gmapping.html (accessed May 08, 2021).

[16] R. L. Guimarães, A. S. de Oliveira, J. A. Fabro, T. Becker, and V. A. Brenner, "ROS navigation: Concepts and tutorial," *Stud. Comput. Intell.*, vol. 625, no. February, pp. 121–160, 2016, doi: 10.1007/978-3-319-26054-9_6.

[17] "LD-250 | OMRON, Türkiye." https://industrial.omron.com.tr/tr/products/ld-250 (accessed Jun. 20, 2021).

[18] "Robot Operating System Market by Types | ROS Market - 2024 | MarketsandMarkets." https://www.marketsandmarkets.com/Market-Reports/robot-operating-system-market-19870751.html (accessed Apr. 23, 2021).

[19] "Autonomous Mobile Robots Market Size Report, 2020-2027." https://www.grandviewresearch.com/industry-analysis/autonomous-mobile-robots-market (accessed Apr. 23, 2021).